SGA-MCTS: Decoupling Planning from Execution via Training-Free Atomic Experience Retrieval

Xin Xie[1]*, **Dongyun Xue**[2]*, **Wuguannan Yao**[1], **Mingxiao Feng**[2], **Wengang Zhou**[2], **Xiang Qi**[1], **Houqiang Li**[2], **Peng Zhang**[1]†
[1]Ant Digital Technologies, Ant Group
[2]Hefei Comprehensive National Science Center, Hefei, China
{xinyuan.xx, yaowuguannan.ywgn, qixiang.qx, minghua.zp}@antgroup.com,
{andyxue, fengmx}@mail.ustc.edu.cn, {zhwg, lihq}@ustc.edu.cn

Abstract

LLM-powered systems require complex multi-step decision-making abilities to solve real-world tasks, yet current planning approaches face a trade-off between the high latency of inference-time search and the limited generalization of supervised fine-tuning. To address this limitation, we introduce **SGA-MCTS**, a framework that casts LLM planning as non-parametric retrieval. Offline, we leverage Monte Carlo Tree Search (MCTS) to explore the solution space and distill high-fidelity trajectories into State-Goal-Action (SGA) atoms. These atoms are de-lexicalized primitives that abstract concrete entities into symbolic slots, preserving reusable causal logic while discarding domain-specific noise. Online, a retrieval-augmented agent employs a hybrid symbolic-semantic mechanism to fetch relevant SGAs and re-ground them into the current context as soft reasoning hints. Empirical results on complex benchmarks demonstrate that this paradigm enables frozen, open-weights models to match the performance of SOTA systems (e.g., GPT-5) without task-specific fine-tuning. By effectively amortizing the heavy computational cost of search, SGA-MCTS achieves System 2 reasoning depth at System 1 inference speeds, rendering autonomous planning both scalable and real-time feasible.

1 Introduction

LLM-based autonomous agents increasingly leverage external tools to solve complex, multi-step problems (Schick et al., 2023; Qin et al., 2023; Wang et al., 2024; Qu et al., 2025; Feng et al., 2025), extending capabilities beyond plain text generation. This transformation has enabled sophisticated capabilities, from booking flights to analyzing scientific data, by grounding language in executable environments. However, as task complexity grows—demanding long-horizon planning, multi-step dependencies, and dynamic error recovery—agents face an increasingly severe dilemma. On one hand, they can employ inference-time search methods (Yao et al., 2023a; Zhou et al., 2023; Snell et al., 2024) to achieve deep, strategic reasoning, but at the cost of high latency that renders them impractical for interactive applications. On the other hand, they can embed reasoning patterns into model parameters via supervised fine-tuning, but this suffers from "parametric rigidity": any adaptation to new tool schemas or domain logic demands expensive retraining and risks catastrophic forgetting (Masterman et al., 2024; Chen et al., 2025; Schick et al., 2023). This constraint has created a critical bottleneck, forcing practitioners to choose between depth and deployability.

To address this challenge, we formulate learning as non-parametric experience curation rather than implicit weight adaptation. Our framework is built on the insight that complex reasoning, despite surface variability, structurally decomposes into recurring, logic-invariant atoms. For instance, a refund operation adheres to the same abstract protocol regardless of the user ID; similarly, a multi-hop query follows an identical retrieval-verification loop irrespective of the target entities. While prior memory-based agents attempt to exploit this repetition via monolithic trajectory retrieval (Shinn et al., 2023; Zhao et al., 2024; Zhang et al., 2025c), they suffer from contextual rigidity: slight deviations in entity values or schemas often render retrieved plans brittle or irrelevant. Our approach overcomes this limitation by distilling execution traces into State-Goal-Action (SGA) atoms—symbolic primitives that isolate causal logic from surface details. By de-lexicalizing concrete entities into typed slots (e.g. <ID>), we transform raw episodes into a composable algebra of reasoning skills, enabling robust transfer across novel tasks and unseen tool ecosys-

* Equal contribution.
† Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 1176–1193
July 2-7, 2026

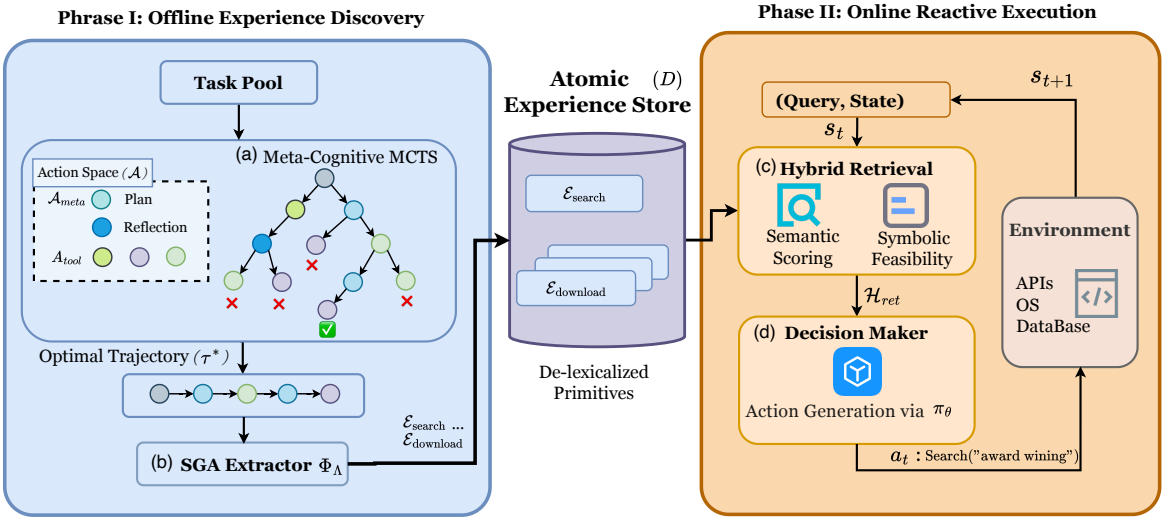

Figure 1: SGA-MCTS Framework Architecture. (a) MCTS exploration discovers optimal reasoning paths. (b) Valid trajectories are distilled into de-lexicalized SGA atoms. (c) Relevant SGAs are retrieved as soft hints based on the current state. (d) The Decision Maker grounds these hints to generate the final action a_t.

tems.

We implement this insight via a two-phase architecture aligned with the dual-process theory of cognition, distinguishing between deliberative planning (System 2) and reactive execution (System 1). In the offline discovery phase, we employ MCTS as a data generator rather than a direct policy optimizer. By augmenting the search space with meta-cognitive operators for goal decomposition and error recovery, we extensively explore the state space to identify optimal reasoning trajectories. These trajectories are subsequently condensed into the SGA store via schema-guided abstraction. In the online execution phase, the agent operates as a retrieval-augmented generator. Instead of conducting online search with expensive token cost, it utilizes a hybrid symbolic-semantic retrieval mechanism to fetch relevant SGAs, which are then integrated into the current context as soft reasoning hints. This framework effectively amortizes the computational cost of planning, enabling the agent to approximate the strategic depth of search-based methods with the inference latency of standard generation.

In contrast to RAG systems that retrieve static facts or memory agents bound by rigid trajectory replay (Zhao et al., 2024; Ouyang et al., 2025; Zhang et al., 2025a), SGA-MCTS facilitates the dynamic recombination of abstract reasoning patterns. This flexibility provides two critical advantages: zero-shot generalization to unseen tools via logic re-grounding, and adaptive reasoning depth controlled by context richness. On complex benchmarks requiring multi-hop dependency resolution, this approach yields substantial gains: a standard open-weights 8B with non-thinking mode achieves a 44.79% success rate—a 13.86% absolute improvement over its zero-shot baseline—approaching the performance of proprietary systems like GPT-5 without a single parameter update. Furthermore, by condensing raw MCTS explorations by 6.9× into reusable atoms, we demonstrate that the heavy computational cost of deep reasoning can be effectively amortized.

Our contributions are:

1. **Amortized Efficiency:** We decouple strategic planning from execution. This eliminates inference-time search overhead and reduces token consumption by ∼2,080 tokens per task (76% reduction) compared to reasoning-heavy baselines, granting the agent System 2 depth at System 1 cost.

2. **De-lexicalized SGA Abstraction:** We introduce State-Goal-Action atoms that distill raw trajectories into symbolic primitives. This representation achieves a 6.9x compression rate and enables robust zero-shot generalization to unseen toolsets by extracting reusable causal logic from domain specific trajectories.

3. **Parameter-Efficient Generalization:** We show that intelligent retrieval bridges the gap between model sizes. SGA-MCTS facilitates an 8B model to achieve a +13.86% gain, ef-

fectively outperforming a 32B baseline in a completely training-free manner.

2 Related Works

LLM Agents and Tool Use. LLMs have evolved from passive generators to proactive agents via external tools (Schick et al., 2023; Qin et al., 2023; Wölflein et al., 2025; Li et al., 2025, 2024). While reasoning-action interleaving (e.g., ReAct (Yao et al., 2022)) improves grounding, greedy strategies often fail in long-horizon tasks due to error propagation. Unlike supervised fine-tuning approaches (Zhu et al., 2025; Masterman et al., 2024; Liu et al., 2024; Lin et al., 2024) that suffer from "parametric rigidity," SGA-MCTS adopts a non-parametric, training-free paradigm via retrievable experience, enabling zero-shot adaptation without models' parametric updates.

Planning and Inference Search. To overcome greedy shortsighted, "test-time scaling" methods (Yao et al., 2023b; Muennighoff et al., 2025; Zhou et al., 2023; Erdogan et al., 2025) introduce deliberate search. However, they incur high latency. SGA-MCTS address this by shifting computationally intensive search to an offline discovery phase (System 2), allowing the online agent to execute a lightweight, retrieval-augmented policy (System 1) with negligible latency.

Agent Memory. Memory mechanisms evolve agents into lifelong learners (Zhang et al., 2024; Fang et al., 2025; Yan et al., 2025; Zhang et al., 2025b; Kang et al., 2025). While recent frameworks store structured trajectories (Zhang et al., 2025a; Ouyang et al., 2025), they often rely on holistic trajectory retrieval, leading to contextual rigidity (Zhang et al., 2025c). SGA-MCTS overcomes this via de-lexicalized atomization, distilling trajectories into abstract $(State, Goal) \rightarrow Action$ primitives to enable compositional generalization.

3 Methodology

We propose SGA-MCTS, a framework that decouples planning from execution via training-free atomic experience retrieval. As shown in Figure 1, it operates in two phases: (1) Offline Experience Discovery, where MCTS mines optimal reasoning paths and distills them into de-lexicalized State-Goal-Action atoms; and (2) Online Reactive Execution, where the agent retrieves these atoms as soft hints to solve tasks efficiently. We treat the offline discovery phase as a non-parametric learning process, where atomic SGAs serve as explicit policy proxies.

3.1 Problem Formulation

We formulate tool-use planning as a Goal-Conditioned Markov Decision Process (MDP), $\mathcal{M} = \langle \mathcal{S}, \mathcal{G}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. At step t, the agent observes state s_t, aims for goal g, executes action a_t, and receives reward r_t.

Structured State & Abstraction. Unlike flat token sequences, we define a structured state $s_t = \langle h_t, \mathcal{K}_t, g_t \rangle$, comprising the execution history h_t, a symbolic known-info tracker $\mathcal{K}_t$, and the current atomic sub-goal g_t. We employ a State Abstraction Function $\Phi : \mathcal{S} \rightarrow \hat{\mathcal{S}}$ that de-lexicalizes specific entities in $\mathcal{K}_t$ into typed slots (e.g., `<ID>`), enabling retrieval across disjoint domains.

Hybrid Action Space. The action space $\mathcal{A} = \mathcal{A}_{tool} \cup \mathcal{A}_{meta}$ unifies external API calls ($\mathcal{A}_{tool}$) with internal reasoning operators ($\mathcal{A}_{meta}$). The latter includes `Plan` for decomposition and `Reflect` for error handling, allowing the agent to modify its logical trajectory without further interacting with the external environment.

3.2 Objective and Reward

We treat the atomic experience store $\mathcal{D}$ as a non-parametric policy support. Our objective is to optimize $\mathcal{D}$ to maximize the expected return $\mathbb{E}_{\tau \sim \pi_{\mathcal{D}}}[R(\tau)]$. To balance correctness and efficiency, we design a gated reward function:

$$R(\tau) = \mathbb{1}_{\text{succ}}(\tau) \cdot \left((1 - \lambda) + \lambda \cdot \frac{1}{1 + |\tau|} \right) \quad (1)$$

where $\mathbb{1}_{\text{succ}}(\tau) \in \{0, 1\}$ is the binary success indicator, and $|\tau|$ denotes the length of the trajectory. $\lambda \in [0, 1]$ is a hyperparameter balancing the base reward for correctness and the bonus for efficiency. This multiplicative formulation ensures that failed trajectories always yield zero reward, strictly biasing the subsequent MCTS exploration toward successful reasoning paths with minimal steps.

3.3 Phase I: Offline Experience Acquisition

To circumvent the reasoning limitations of greedy policies, we employ Monte Carlo Tree Search (MCTS) as a high-fidelity data generator. This phase aims to explore the state space extensively and distill optimal trajectories into a generalized reusable format.

3.3.1 Meta-Cognitive Search

We construct a search tree where nodes represent structured states s and edges represent actions. Standard MCTS often struggles with the large branching factor of open-ended tool use. To mitigate this, we augment the action space $\mathcal{A}$ with meta-cognitive operators that function as heuristic pruners during the expansion phase:

- Plan ($\mathcal{A}_{\text{plan}}$): Enforces a topological ordering on sub-goals. By decomposing the global goal g into atomic steps $\{g_1, g_2, \dots\}$, it constrains the search to relevant tool subsets, effectively pruning logically invalid branches.
- Reflect ($\mathcal{A}_{\text{reflect}}$): Triggered upon execution failure. It analyzes error feedback to generate counterfactual pivots, enabling the search to escape local optima that trap standard greedy samplers.

State-Goal Deduplicated Exploration. A major advantage of using MCTS as our rollout operator lies in its inherent structural deduplication at the $(State, Goal)$ level. Unlike linear path sampling rollout operators (e.g., ReAct), which redundantly explores the same state across independent trajectories, MCTS aggregates visitation statistics at each decision node, effectively viewing the search space as a directed graph.

MCTS efficiently approximates the optimal policy $(s_t, g_t) \to a_t^*$. Guided by the Upper Confidence Bound (UCB) criterion, the search prioritizes high-potential branches while pruning suboptimal subtrees. The accumulated Q-values—derived from leaf-node evaluations via the gated reward function (Eq. 1)—serve as a quality assurance filter. This mechanism ensures that the search converges specifically on actions that are verifiably correct, rather than merely plausible. This selectivity is critical for our atomic experience store: we discard raw, noisy trajectories in favor of canonical SGA triplets derived solely from high-confidence paths. Consequently, each stored entry represents a verified transition, ensuring the store remains compact and highly distinctive."

3.3.2 Atomic SGA Extraction via Schema-Guided Abstraction

Upon the convergence of MCTS, the search tree provides a set of high-reward trajectories τ^*. To transform these monolithic sequences into reusable knowledge, we decompose each successful transition into a State-Goal-Action (SGA) triplet. Storing raw transitions (e.g., searching specifically for "award-winning" films) leads to "lexical overfitting." To bridge this gap, we introduce a Schema-Guided Abstraction Function Φ_Λ that distills raw execution data into generalized reasoning "atoms."

For each optimal transition (s_t, g_t, a_t^*) identified by MCTS, we generate an atomic experience $\mathcal{E}_i$:

$$\mathcal{E}_i = \Phi_\Lambda(s_t, g_t, a_t^*) = \langle \hat{S}, \hat{G}, \hat{A} \rangle \tag{2}$$

The abstraction process is governed by three core components:

1. State Abstraction ($\hat{S}$): Instead of preserving the full history h_t, $\hat{S}$ encapsulates a semantic summary of the context and a symbolic schema $\hat{S}_{sym}$. The latter identifies the entity types currently verified in $\mathcal{K}_t$ (e.g., `<MOVIE_QUERY>` or `<ID>`), which serves as a prerequisite for the action.
2. Goal Abstraction ($\hat{G}$): The sub-goal g_t (e.g., "Retrieve streaming link") is preserved as the functional intent. This allows the agent to retrieve the atom based on its "intended utility" rather than surface text matching.
3. Action De-lexicalization ($\hat{A}$): We employ a LLM to identify a selective mask to parameters: arguments matching the domain schema (e.g., `query`) are replaced by typed slots (`<QUERY>`), while control literals essential for API behavior are preserved. This hybrid structure ensures the agent generalizes to novel data while retaining the execution logic discovered via MCTS.

3.4 Phase II: Online Reactive Execution

In the online phase, the agent transitions into a reactive executor (System 1), operating under strict latency constraints. We forego computationally expensive search in favor of a lightweight retrieve-inject-generate pipeline. We treat the retrieved experiences not as rigid commands or templates to be filled, but as soft reasoning hints that provide a logical prior for the agent's generative process.

3.4.1 Hybrid Symbolic-Semantic Retrieval

Standard dense retrieval often fails in tool-use scenarios because it prioritizes surface-level semantic similarity while ignoring the hard constraints of

Backbone	Method	StableTB *(Complex)*	ToolHop *(Dependency)*	BFCL v3 *(Multi-turn)*	Avg.
Proprietary	GPT-5	70.20	43.52	51.68	55.13
Qwen3-8B	ReAct	11.50 ±0.80	36.94 ±3.35	44.35 ±3.65	30.93
	LangMem	19.30 ±2.60	39.72 ±0.56	46.07 ±1.19	35.03
	SGA (Ours)	**43.80** ±2.10	**41.87** ±0.71	**48.70** ±1.40	**44.79**
Qwen3-14B	ReAct	21.70 ±2.90	36.38 ±0.16	47.47 ±1.86	35.18
	LangMem	28.50 ±2.80	36.21 ±0.67	48.22 ±1.13	37.64
	SGA (Ours)	**40.20** ±4.70	**46.63** ±1.21	**52.33** ±2.21	**46.39**
Qwen3-32B	ReAct	18.20 ±1.80	48.70 ±0.51	53.53 ±1.85	40.14
	LangMem	24.70 ±2.50	48.24 ±0.67	52.27 ±1.13	41.73
	SGA (Ours)	**48.20** ±2.00	**50.87** ±0.37	**54.20** ±1.20	**51.09**

Table 1: Main Results: Success Rate (%) comparison across different backbones. We compare SGA-MCTS (highlighted in blue) against baselines. The table is formatted to span a wider area for better readability.

execution logic. To address this, we propose a dual-factor scoring mechanism that evaluates candidate experiences across two orthogonal dimensions: semantic relevance and symbolic feasibility.

At each timestep t, the agent constructs a query vector $\mathbf{q}_t$ and identifies the set of currently available symbolic slots $\Lambda_t = \text{Keys}(\mathcal{K}_t)$ updated via an LLM-based state tracker. The unified relevance score for a candidate experience $\mathcal{E}_i$ in the store $\mathcal{D}$ is:

$$\text{Score}(\mathcal{E}_i \mid \mathbf{q}_t, \Lambda_t) = (1-\beta) \cdot \underbrace{\cos(\mathbf{q}_t, \mathbf{e}_i)}_{\text{Semantic Relevance}} + \beta \cdot \underbrace{\frac{|\Lambda_t \cap \hat{S}^i_{sym}|}{|\hat{S}^i_{sym}| + \epsilon}}_{\text{Symbolic Feasibility}} \tag{3}$$

where $\beta \in [0, 1]$ modulates the balance between intent alignment and execution grounding. Symbolic feasibility acts as a logical gate, penalizing atoms whose prerequisite parameters are missing from the current state, thereby filtering out unexecutable "hallucinated" plans.

3.4.2 Action Generation via the Decision Maker

The Decision Maker functions as a generative synthesizer that grounds abstract reasoning patterns into executable actions. Instead of enforcing rigid templates, we inject the top-k retrieved SGA triplets into the agent's context as soft logical priors. Leveraging the model's in-context learning capabilities, the Decision Maker performs implicit instantiation: it autonomously maps the symbolic slots in the de-lexicalized atoms (e.g., `<QUERY>`) to concrete entities derived from the execution history h_t. This allows for adaptation where retrieved logic guides, rather than constrains, the generation process based on real-time observation o_t.

The decision process is formally modeled as a conditional generation task:

$$a_t \sim \pi_\theta(a_t \mid h_t, \mathcal{E}_{\text{ret}}) \tag{4}$$

This "Reasoning-as-Retrieval" paradigm enables the agent to exhibit the strategic depth of offline search at the latency of greedy generation, effectively bypassing the fragility of manual slot-filling or rule-based execution.

4 Experiments

To validate the effectiveness and efficiency of SGA-MCTS, we conducted extensive evaluations on three diverse benchmarks covering complex tool chaining, embodied decision-making, and multi-turn state tracking.

4.1 Experimental Setup

Datasets. To evaluate our framework's capability to generalize from offline exploration to online execution, we utilize three datasets with specific split strategies for experience store construction (Offline) and evaluation (Online):

- StableToolbench (Guo et al., 2024): We adopt a cross-difficulty transfer setting. We utilize the G2 Instruction subset (intermediate complexity) for offline experience discovery and evaluate on the G3 Instruction subset (complex/hard). This tests the agent's ability to extrapolate logic from simpler scenarios to long-horizon tasks.

- ToolHop(Ye et al., 2025): A query-driven benchmark for multi-hop tool use containing 995 complex queries. We randomly sample 50% of the data for offline exploration to build the SGA store. The remaining 50% are reserved for evaluation, ensuring the agent must handle unseen query-tool dependencies.

- BFCL v3 (Multi-turn Base) (Patil et al., 2025): A challenging subset of the Berkeley Function Calling Leaderboard focusing on multi-turn dialogue state tracking. We sample only 25% of the episodes for offline atom extraction and evaluate on the remaining 75%. This low-resource setting stress-tests the efficiency of our symbolic state tracking mechanism.

Baselines. We compare SGA-MCTS against the following representative paradigms:

- ReAct (Zero-shot) (Yao et al., 2022): The most widely used prompting-based baseline. It relies solely on the model's internal parametric knowledge to interleave reasoning and tool calls. This baseline serves to demonstrate the "baseline" capability of the Qwen3 backbone without any external experience guidance.

- LangMem (LangChain, 2025): A representative long-term memory baseline. It extracts and stores key information from interactions to enable future retrieval. Specifically, we adopt its episodic memory implementation, which allows the agent to continuously improve by learning from past experiences. This baseline serves to evaluate the benefits of retrieval-based memory guidance.

Consistent with the training-free nature of our approach, we evaluate the Qwen3 family (8B, 14B, and 32B) (Yang et al., 2025) in standard non-thinking mode to isolate gains strictly from our retrieval mechanism. Conversely, we employ GPT-5 in thinking mode as a high-ceiling reference to measure the extent to which pure retrieval can bridge the gap to proprietary reasoning models.

Superiority in Complex Planning. SGA-MCTS achieves consistent and substantial improvements across all datasets. On average, our method boosts the performance of Qwen3-8B by 13.86% absolute (from 30.93% to 44.79%). This advantage is particularly pronounced on StableToolBench, the most challenging benchmark involving complex instruction following, where SGA-MCTS achieves a relative improvement of nearly 400% over the ReAct baseline (43.80% vs. 11.50%).

Structured Abstraction vs. Holistic Memory. While LangMem improves over ReAct by retrieving past trajectories (35.03% avg), it still lags significantly behind SGA-MCTS (44.79% avg). The performance gap is widest on StableToolBench (19.30% for LangMem vs. 43.80% for SGA). This disparity supports our hypothesis regarding "contextual rigidity": LangMem's retrieval of raw trajectories often fails when specific entity values or constraints shift in new tasks. In contrast, SGA's de-lexicalized abstraction isolates reusable causal logic from domain noise, enabling robust generalization even when the surface form of the task changes drastically.

Parameter-Efficient Planning. We observe that external memory can serve as a potent alternative to purely parametric scaling. Notably, the Qwen3-8B agent with SGA achieves performance comparable to, and in some cases exceeding, the much larger Qwen3-32B baseline. This suggests that for logic-intensive tasks, retrieving curated reasoning patterns offers a resource-efficient path to high performance, reducing the dependency on massive model size.

Closing the Gap with Proprietary SOTA. SGA-MCTS allows open-weights models to approximate closed-source models. The Qwen3-32B + SGA agent achieves an average success rate of 51.09%, significantly narrowing the gap with GPT-5 (55.13%). On the BFCL v3 benchmark, our method actually outperforms GPT-5 (54.20% vs. 51.68%), demonstrating that specialized, retrieval-augmented planning can exceed the capabilities of general-purpose frontier models in structured tool-use scenarios.

4.2 Efficiency and Resilience

Beyond success rates, Table 2 evaluates the computational cost and stability on the hardest tasks.

Amortized Cost and Inference Efficiency. The efficiency gains of SGA-MCTS extend beyond simple token savings to a fundamental shift in computational allocation. As detailed in Table 2, our method reduces token consumption by 76% ($\sim$2,080 tokens per task) compared to the *ReAct-Thinking* baseline. Traditional "inference-time scaling" approaches (e.g., CoT or online search) incur

a linear "reasoning tax" for every query, requiring extensive token generation to traverse the solution space. In contrast, SGA-MCTS amortizes this cost into the offline phase. By converting complex reasoning paths into retrievable static assets, the online agent effectively "memorizes" the strategic depth of MCTS. This allows it to achieve System 2-level decision quality with the latency profile of a shallow, greedy executor, making high-intelligence planning viable for latency-sensitive deployment.

Resilience to Reasoning Depth and Drift. Long-horizon planning is notoriously brittle due to the cascading error propagation inherent in autoregressive generation. Table 2 quantifies this degradation: baseline performance collapses precipitously as task complexity increases, dropping to a mere 15.38% on chains exceeding 4 hops. SGA-MCTS, however, exhibits remarkable resilience, maintaining a robust success rate of 61.54% in these deep-dependency scenarios. This stability stems from the function of retrieved SGA atoms as logic checkpoints. Instead of relying solely on a volatile context window that drifts over time, the agent regrounds its logic at every step using validated, de-lexicalized schemas. This mechanism effectively resets the "reasoning uncertainty" at each hop, preventing the hallucination drift that typically derails long-chain execution.

Method	Success Rate (%)		Avg. Tokens
	Easy	Hard	
ReAct (Baseline)	16.67	7.69	260.54
ReAct - Thinking	31.25	15.38	2712.75
SGA-MCTS (Ours)	**43.75**	**61.54**	**630.28**
Imp. vs. React-Thinking	+12.50	**+46.16**	**-2082.47**

Table 2: Efficiency on StableToolBench G3. SGA-MCTS achieves significantly higher success rates on hard tasks while maintaining efficient token usage compared to reasoning-heavy baselines.

5 Ablation and Diagnostic Study

To isolate the sources of improvement, we conduct a fine-grained analysis of the framework's constituents: the topology of offline discovery, the impact of de-lexicalized atoms, and the sensitivity of hybrid retrieval.

5.1 Quality of Offline Discovery

The effectiveness of our framework hinges on the MCTS to mine high-quality logic offline.

Backbone	Search Topology Metrics		Avg Nodes
	Branch. Factor	Avg Depth	
Qwen3-8B	1.25 $\pm$ 0.24	11.98 $\pm$ 3.81	28.9
Qwen3-14B	1.29 $\pm$ 0.30	13.50 $\pm$ 5.16	38.4
Qwen3-32B	1.30 $\pm$ 0.30	13.96 $\pm$ 4.95	40.4

Table 3: MCTS Exploration Statistics (BFCL v3). The low *Branch. Factor* ($\approx$ 1.3) and high *Depth* ($>$ 11) indicate efficient, deep reasoning.

Deep-but-Narrow Exploration. Table 3 reveals a "deep-but-narrow" search topology (branching $\approx$ 1.3, depth $>$ 11). This corroborates that our meta-cognitive operators successfully guide exploration through narrow logical corridors, avoiding shallow heuristics. By encapsulating these expensive trajectories into retrievable atoms, we effectively amortize search costs, strictly decoupling reasoning depth from online latency.

Dataset	Explored Actions	SGAs ($\lvert\mathcal{D}\rvert$)
StableToolBench	2388	213
ToolHop	10685	1560
BFCL v3	1290	226

Table 4: Statistics of the atomic experience store. We compare the volume of raw actions explored via MCTS during the offline phase against the final size of the deduplicated, de-lexicalized atomic experience store ($\mathcal{D}$). The reduction highlights the high reusability of the distilled reasoning atoms.

High-Density Compression. Our targeted exploration strategy yields significant data compaction. As shown in Table 4, we consolidate 10,685 raw actions from the ToolHop dataset into just 1,560 reusable atoms—a compression factor of $\sim 6.9\times$. This finding validates that diverse tasks share a common, low-dimensional basis of recurring causal logic. Instead of storing redundant execution traces, SGA-MCTS captures these essential patterns, thereby mitigating the combinatorial explosion of the state space.

5.2 Retrieval Sensitivity and Hybrid Scoring

Top-K Sensitivity. Figure 2 shows a clear positive trend: performance consistently improves as K increases. The success rate climbs steadily with more retrieved SGA atoms, indicating that the model effectively uses the richer context to ground its reasoning. There is no performance degradation

from $Top - k$ grows, suggesting that more reference examples provide stronger logical guidance.

Necessity of Symbolic Constraints. Ablating the symbolic term ($\beta = 0$) on Qwen3-8B yields a 1.5% drop, driven by *precondition hallucinations*–invoking tools without requisite arguments. As shown in Figure 2, this confirms symbolic feasibility acts as a critical validity gate.

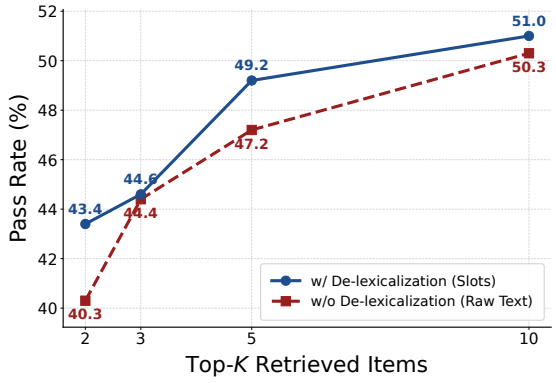

Figure 2: Impact of Retrieval Size (K) on StableToolBench. Unlike Raw Text (Red) which degrades due to noise, our De-lexicalized approach (Blue) maintains robust performance as K increases.

5.3 Generalization to Unseen Tools

We investigate whether SGA achieves robust generalization or merely relies on memorization by correlating performance gains with tool familiarity.

Metric: Tool Familiarity Score. To quantify the semantic distribution shift between the offline discovery toolset ($\mathcal{T}_{src}$) and the online evaluation toolset ($\mathcal{T}_{tgt}$), we introduce the Tool Familiarity Score ($\mathcal{S}_{\text{fam}}$). Unlike rigid overlap statistics, $\mathcal{S}_{\text{fam}}$ operates in the dense embedding space to measure the *semantic proximity* of the testing environment to the source domain. For each tool t in the target set, we identify its nearest neighbor in the source set and compute the average peak similarity:

$$\mathcal{S}_{\text{fam}} = \frac{1}{|\mathcal{T}_{tgt}|} \sum_{t \in \mathcal{T}_{tgt}} \max_{t' \in \mathcal{T}_{src}} \cos(\mathbf{e}_t, \mathbf{e}_{t'}) \quad (5)$$

where $\mathbf{e}_t$ denotes the dense embedding of the tool's functional description. Intuitively, $\mathcal{S}_{\text{fam}}$ serves as a continuous proxy for *domain novelty*: a score approaching 1.0 implies an in-distribution setting where the agent can rely on memory, whereas a lower score indicates a high-entropy OOD scenario, demanding the transfer of abstract reasoning logic to semantically distinct tools.

Results Analysis. Figure 3 illustrates the performance disparity between SGA-MCTS and LangMem(LangChain, 2025) across varying tool familiarity. On high-familiarity benchmarks (e.g., BFCL v3, $\mathcal{S}_{\text{fam}} \approx 0.99$), the gap is narrow, as LangMem's retrieval of raw, lexically-matched experience remains effective for seen tools. However, this gap widens significantly on OOD domains. On StableToolBench ($\mathcal{S}_{\text{fam}} \approx 0.57$), where LangMem expose the limitations due to contextual rigidity, SGA achieves a dominant lead (43.80% vs. 19.30% for 8B). This confirms that while raw memory suffices for reproduction, SGA's de-lexicalized abstraction is essential for OOD generalization, enabling the transfer of reasoning logic to different tool ecosystems.

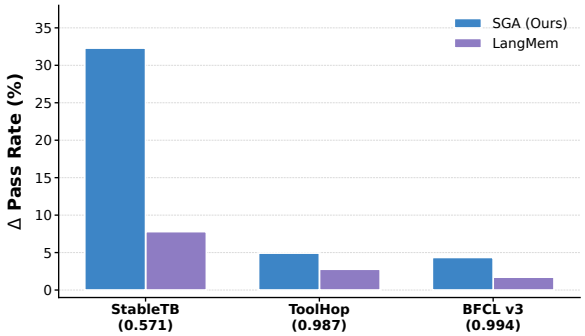

Figure 3: Δ Pass Rate vs. Dataset (Tool Familiarity). The inverse relationship demonstrates that SGA is most effective in OOD settings (low tool familiarity), showing its strong ability to generalize abstract reasoning logic to unseen tools.

6 Conclusion

We presented SGA-MCTS, a framework that decouples deliberative planning from reactive execution. By amortizing the heavy computational cost of search into an offline phase, we address the limitations of parametric rigidity, recasting planning as the efficient retrieval of de-lexicalized atomic experiences. Our results demonstrate that this non-parametric approach allows frozen, small-scale models to approximate the reasoning depth of proprietary frontier systems without task-specific fine-tuning. By successfully embedding System 2 reasoning patterns into a retrievable System 1 format, SGA-MCTS offers a scalable path toward interpretable autonomy. We envision this 'Reasoning-as-Retrieval' paradigm as a promising direction for future research, particularly in enabling robust generalization across dynamic environments.

7 Limitations

Despite the efficiency gains, SGA-MCTS faces two primary constraints. First, the quality upper bound: the online agent's performance is strictly limited by the fidelity of the offline MCTS exploration. Inaccurate verification during the discovery phase leads to the storage of sub-optimal logic ("low-quality trajectories"), and extremely high-entropy domains may challenge the coverage of our finite experience store.

Second, initialization via seed questions. The construction of the atomic experience store is currently driven by a set of cold-start queries. While our approach efficiently mines optimal reasoning paths (*depth*) within these tasks, the categorical *breadth* of the store is naturally influenced by the diversity of the initial input distribution. Exploring mechanisms for autonomous task proposal (e.g., Active Learning) to expand coverage beyond the seed set remains a promising direction for future research.

References

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. 2025. Atlas: Agent tuning via learning critical steps. *arXiv preprint arXiv:2503.02197*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.(2024). *arXiv preprint arXiv:2401.08281*.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-act: Improving planning of agents for long-horizon tasks. *arXiv preprint arXiv:2503.09572*.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2025. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *arXiv preprint arXiv:2403.07714*.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. *arXiv preprint arXiv:2506.06326*.

LangChain. 2025. Langmem: Long-term memory for llm agents. https://github.com/langchain-ai/langmem. Accessed: 2025-06-01.

Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. 2024. Agent-oriented planning in multi-agent systems. *arXiv preprint arXiv:2410.02189*.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, et al. 2025. Deepagent: A general reasoning agent with scalable toolsets. *arXiv preprint arXiv:2510.21618*.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, et al. 2024. Hammer: Robust function-calling for on-device language models via function masking. *arXiv preprint arXiv:2410.04587*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, et al. 2024. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.

Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. 2024. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*.

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

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. 2025. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

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

Zhiruo Wang, Zhoujun Cheng, Hao Zhu, Daniel Fried, and Graham Neubig. 2024. What are tools anyway? a survey from the language model perspective. *arXiv preprint arXiv:2403.15452*.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. 2025. Llm agents making agent tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26092–26130.

BY Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. 2025. General agentic memory via deep research. *arXiv preprint arXiv:2511.18423*.

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

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

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, Tao Gui, Qi Zhang, Xuanjing Huang, and Jiecao Chen. 2025. ToolHop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2995–3021, Vienna, Austria. Association for Computational Linguistics.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025a. G-memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398*.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. 2025b. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. 2025c. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

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

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. 2025. Knowagent: Knowledge-augmented planning for llm-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3709–3732.

A Implementation Details

A.1 Model Configuration and Hyperparameters

To facilitate reproducibility, we detail the specific configurations used in the SGA-MCTS framework. We employ the Qwen3 model family (8B, 14B, and 32B) (Yang et al., 2025) as the backbone for all agentic components, encompassing both the offline Planner and the online Decision Maker.

To balance the trade-off between generative creativity and instruction-following adherence, we enforce a unified decoding strategy across all tasks. Specifically, we set the temperature to 0.6, with nucleus sampling parameters `top_p` $= 0.95$ and `top_k` $= 20$. Additionally, we utilize `min_p` $= 0$, consistent with the default configuration specified in the model's `generation_config.json`.

These sampling parameters are maintained consistently during the offline MCTS phase to ensure that the distilled experiences remain representative of the model's natural probability distribution. The specific hyperparameters governing the MCTS exploration method and the hybrid retrieval mechanism are systematically summarized in Table 5.

Module	Parameter	Value
Offline MCTS	Exploration Constant (c)	1.41
	Max Iterations (N)	50
	Max Depth	10
	Lambda (λ)	0.1
Generation	Temperature (T)	0.6
	Top-P / Top-K	0.95 / 20
	Min-P	0.0
	Max Context Window	32k
Retrieval	Semantic Weight (α)	0.7
	Symbolic Weight (β)	0.3
	Embedding Model	`bge-m3`
	Smoothing Term (ϵ)	1e-5
	Retrieved SGA Top-k	3

Table 5: Hyperparameters for the SGA-MCTS Framework.

A.2 Computational Infrastructure

All experiments, including the computationally intensive offline MCTS trajectory distillation and the online evaluation benchmarks, were conducted on a high-performance computing cluster. The infrastructure consists of 8 $\times$ NVIDIA A100 (80GB) GPUs. This substantial VRAM capacity is essential for the efficient parallel inference of the Qwen3-32B model.

For the atomic experience store, we leverage the FAISS library (CPU-optimized build) (Douze et al., 2024) to execute high-dimensional vector similarity searches with low latency. The BAAI/bge-m3(Chen et al., 2024) model serves as the primary embedding backbone, encoding semantic states into dense vector representations.

B Prompt Templates

This section presents the exact system prompts employed across the SGA-MCTS pipeline. These prompts act as the interface between our structured algorithms and the LLM's reasoning capabilities. In the templates below, 'variable' denotes dynamic slots populated at runtime based on the execution context.

B.1 SGA Extraction (De-lexicalization)

An extractor model utilizes the following prompt to analyze raw MCTS trajectories. Its primary function is to abstract concrete execution paths into generalized, de-lexicalized State-Goal-Action (SGA) triplets, as described in Section 3.3.2.

B.2 Reflection Generator

The Reflection Generator prompt serves as a critic within the MCTS loop. It evaluates whether the current trajectory is logically sound and grounded in tool outputs, ensuring high-quality data for the experience store.

B.3 Online Decision Maker

During the online phase, the Decision Maker uses the following prompt to arbitrate between retrieved experiences and the current context, functioning as the primary actuator of the system.

B.4 SGA Retriever Planner

The SGA Retriever Planner analyzes the current execution state to formulate precise queries for the experience store.

C Additional Experimental Results

C.1 Data Efficiency and Store Scaling

We further analyze the sensitivity of the agent's performance to the scale of the atomic experience store (N_{SGA}). Figure 4 plots the success rate as a function of the number of stored atoms.

The results exhibit a logarithmic growth trajectory. Performance climbs rapidly from a robust baseline of 42.3% at $N = 2$ to a peak of 45.4%

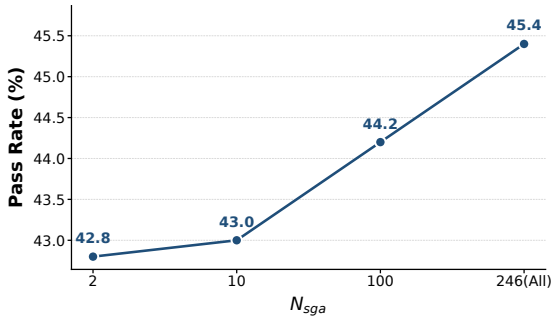

Figure 4: Impact of Experience Volume. Performance improves from 42.3% ($N = 2$) to 45.4% ($N = 246$). The high starting point highlights the data efficiency of SGA atoms, while the sustained growth demonstrates the value of broader coverage.

at $N = 246$. This saturation profile highlights the high information density of our SGA abstraction: a relatively small core of canonical atoms is sufficient to capture the universal reasoning logic of the domain. The subsequent marginal gains suggest that expanding the repository primarily helps in resolving long-tail edge cases rather than learning fundamental capabilities.

C.2 Qualitative Analysis of the Inference Workflow.

Table 10 illustrates the complete lifecycle of an atomic reasoning step.

As shown in the top block, the Stored SGA Atom serves as a generic logic template. It is delexicalized, containing only the schema of required information (e.g., required_slots: ["<YEAR>", "<GENRE>"]) and an abstract action definition, decoupled from specific entity values.

The bottom block details the Online Inference, which proceeds in a "Plan-then-Ground" manner:

- **Planning Phase**: Upon receiving the user query ("cartoons from '94"), the agent first analyzes the context to generate a semantic sub_goal and extracts concrete values into candidate_slots (mapping "cartoons" to animated and "'94" to 1994).

- **Retrieval and Grounding**: The system uses the generated goal and slots to query the experience store. Upon retrieving the matching SGA (th_filter_constraint_01), the agent instantiates the abstract action template with the concrete values, resulting in the final executable tool call.

This mechanism ensures that the agent follows proven reasoning patterns (from MCTS) while dynamically adapting to new data instances.

C.3 Tool Schema Specification

To understand the complexity of the environment, we present the JSON definitions of the meta-cognitive operators injected into the agent's action space. Table 11 details the schemas for `Plan` and `Reflection`. These operators are not external APIs but internal cognitive scaffolds designed to guide the MCTS exploration process.

C.4 Meta-Cognitive Operators in MCTS

To address the reviewer's inquiry regarding how `Plan` and `Reflect` guide the tree construction, Algorithm 1 illustrates their integration during the node expansion phase. These meta-actions are modeled as internal LLM calls that prune the vast tool space before actual execution.

Algorithm 1 Meta-Cognitive Node Expansion in MCTS

Require: Current state s_t, Global Goal g, Action Space $\mathcal{A} = \mathcal{A}_{tool} \cup \{\text{Plan}, \text{Reflect}\}$
if is_initial_state(s_t) **or** sub-goal_completed(s_t) **then**
  // Invoke Plan operator to decompose tasks
  $g_{next} \leftarrow$ LLM_Call(`Plan`, s_t, g)
  Prune $\mathcal{A}_{tool}$ to retain only tools relevant to g_{next}
end if
Sample and execute tool action $a_t \sim \pi(a \mid s_t, g_{next})$
Obtain observation o_t and calculate reward r_t (Eq. 1)
if $r_t == 0$ (Execution Failure) **then**
  // Invoke Reflect operator for error recovery
  $pivot\_strategy \leftarrow$ LLM_Call(`Reflect`, s_t, o_t)
  Backpropagate penalty and update UCB values
  Prioritize $pivot\_strategy$ in the next iteration
else
  Append (s_t, g_{next}, a_t) to valid trajectory τ
end if

System Prompt: SGA Extractor

ROLE
You are the **SGA Extractor** for an advanced autonomous agent. Your task is to analyze raw execution trajectories and distill them into generalized, atomic **State-Goal-Action (SGA)** patterns.

CONTEXT
The input contains a user request and a step-by-step trace of an agent using tools to solve it. Your goal is to extract **reusable logic** from this trace. These SGAs will be stored in a knowledge base to guide future agents. Therefore, the output must be **de-lexicalized** (stripped of specific values) and **templated**

2. SCHEMA DEFINITIONS

State (S)
Describes the agent's understanding of the situation before acting.

- **state_summary**: A generic, high-level summary of the agent's current situation. It should describe the problem to be solved, focusing on what information is needed and what information is already available, without mentioning specific values.

Goal (G)
Describes the agent's immediate intent for the current step.

- **goal**: The specific, high-level sub-goal the agent is trying to achieve. This should describe the question to be answered or the objective to be met at this stage, in a tool-agnostic way.
- Good: Find available flights matching the specified criteria.
- Good: Verify the current status of an order.

Action (A)
Describes the strategic category of the action taken to achieve the Goal.

- **action**: A generalized description of the type of action the agent performs. This should be abstract and never mention the specific tool name. It describes the how in a strategic sense.

INPUT FORMAT
A JSON object containing the `question` and the `trajectory` (a list of actions and results).

OUTPUT FORMAT
You must output a strictly valid JSON object adhering to the following structure:

RULES & CONSTRAINTS

1. **Atomicity**: If a trajectory has 3 steps (A $\rightarrow$ B $\rightarrow$ C), output **3 separate SGA triplets**, not one combined chain. Each step is an independent training example.

Table 6: The formal system prompt for the SGA Extraction. The rules ensure consistent extraction of reusable SGA patterns from execution traces.

System Prompt: MCTS Messages Evaluation Expert

ROLE
You are a **Messages Evaluation Expert** specializing in analyzing Tool Learning / Agentic workflows. Your objective is to audit the logical connection between tool outputs and the AI's final answer.

CONTEXT
You will be presented with a conversation trace involving an AI and various Tools. The trace may include:
1. **AI Messages:** Tool calls or final answers.
2. **Tool Messages:** The raw results returned from a tool.

EVALUATION CRITERIA
You must verify if the **Final Answer** is logically derived from the **Tool Execution Results**.
1. Completion Status
* If the final message contains an `<answer>...</answer>` block and *no* new tool calls, consider the task "Solved".
2. Grounding Verification (The Core Task)
Once the task is deemed "Solved," you must judge the **validity** based *strictly* on the provided traces.

- **High Score Criteria (Grounded):** The final answer is directly derived from the information provided in the `Tool: result` messages. The logic is traceable.
- **Low Score Criteria (Hallucinated/Ungrounded):** The final answer ignores or contradicts tool outputs, or appears to be generated solely from pre-trained knowledge.

SCORING RUBRIC
* **Pass / High Score:** The AI successfully used the tool data to construct the answer.
* **Fail / Low Score:** The AI generated an answer "by itself" without relying on the tool trace evidence.

EXAMPLE SCENARIOS
Scenario A (High Score): Tool returns `{"temp": "15C"}`. AI answers `<answer> 15C </answer>`. Verdict: Fully supported.
Scenario B (Low Score): Tool returns `{"temp": "15C"}`. AI answers `<answer> 30 </answer>`. Verdict: Not directly from tool result.
Scenario C (Low Score): Tool returns `Error`. AI answers "The user is John Doe". Verdict: Severe hallucination.

TASK
Analyze the provided trace and provide your evaluation.

{{format_prompt}}

Table 7: The formal system prompt for the Reflection Generator. This module acts as a critic to ensure that the agent's final output is explicitly grounded in the observation history rather than parametric memory.

System Prompt: Decision Maker

ROLE
You are a professional agent in an autonomous agent system. Your role is to decide which tool to call and what parameters to use.

CONTEXTUAL INPUTS
- Question: {{question}}
- Retrieved Experiences: {{experiences_text}}

OPERATIONAL CONSTRAINTS
* ACTION_MANDATORY: You MUST invoke at least one tool. Do not just respond with text unless you can answer the question based on the tool response.
* NO_HISTORY_DUPLICATES: You MUST NOT repeat a tool call with the exact same parameters used previously in this conversation history.
* NO_USER_CLARIFICATION: Do not ask questions back to the user. You must infer needed information and proceed with an attempt.
* ERROR_RECOVERY: If a previous tool call failed, DO NOT retry it immediately. Change arguments significantly based on the feedback.
* STEP_BY_STEP: ALWAYS attempt to decompose the task and solve it sequentially using the available tools.

Table 8: The formal system prompt for the Decision Maker. The constraints are designed to minimize hallucination and ensure logical tool-chaining in zero-shot scenarios.

System Prompt: SGA Retriever Planner

ROLE
You are the **SGA Retriever Planner** of an autonomous agent system. Your role is to analyze the current situation, extract state information, and prepare queries for SGA experience retrieval.

CONTEXTUAL INPUTS
- User Request: {{question}}
- Execution History: {{history_str}}
- Current World Model (Known Info): {{current_known}}

TASK
1. **Update Known Info**: Extract new facts from execution history and merge with existing knowledge
2. **State Analysis**: Generate abstract state summary suitable for semantic retrieval of similar past experiences
3. **Goal Definition**: Identify the immediate next goal to achieve
4. **Slot Extraction**: List available symbolic slots (e.g., <CITY>, <DATE>, <ID>)

OUTPUT REQUIREMENTS
Provide structured output with these fields:

- **thought**: Your reasoning process and current situation analysis
- **updated_known_info**: Dictionary of new facts from history (can be empty)
- **state_summary**: Abstract state description for SGA retrieval (avoid specific values, focus on patterns)
- **available_slots**: List of symbolic slot tags available in current context
- **next_goal**: Immediate actionable goal for the next step

EXAMPLES
For "What's weather in Beijing?":

- state_summary: "User requests weather information for a specific location"
- available_slots: ["<LOCATION>"]
- next_goal: "Get weather data for specified location"

LANGUAGE
English by default # NOTE
You are NOT responsible for task completion decisions - focus on state analysis and goal formulation.

Table 9: The formal system prompt for the SGA Retriever Planner. The structured format ensures consistent state analysis and retrieval query preparation.

Stored SGA Atom (De-lexicalized Logic)

```
{
  "sga_id": "th_filter_constraint_01",
  "entry": {
    "state": {
      "description": "Requires filtering entity candidates via temporal and genre constraints.",
      "required_slots": [ "<YEAR>", "<GENRE>" ]
    },
    "goal": "Retrieve candidate list matching constraints.",
    "action": "wiki_search"
  }
}
```

Online Inference Workflow (Planning → Retrieval → Grounding)

```
{
  "user_query": "Find me some cartoons from '94.",
  "phase_1_planning": {
    "thought": "I need to find films. The user specifies a year and a genre.",
    "generated_sub_goal": "Retrieve candidate list matching constraints.",
    "extracted_slots": {
      "<YEAR>": "1994",
      "<GENRE>": "animated"
    }
  },
  "phase_2_grounding": {
    "retrieved_sga": "th_filter_constraint_01",
    "reasoning": "The retrieved SGA aligns with my sub-goal. I will fill its abstract action
template with my extracted slots.",
    "final_tool_call": {
      "tool": "wiki_search",
      "arguments": (year=1994, genre=animated)
    }
  }
}
```

Table 10: **Qualitative Example of the Retrieval-Augmented Planning Workflow.** The process operates in two stages: (1) The **Stored SGA Atom** (top) represents a frozen, de-lexicalized reasoning primitive residing in the experience store $\mathcal{D}$. Its `required_slots` define the schema needed for activation. (2) During the **Online Inference Workflow** (bottom), the agent first performs *Planning* to generate a sub-goal and extract candidate slot values (e.g., mapping "'94" to <YEAR>). This structured intent triggers the retrieval of the matching SGA. Finally, in the *Grounding* phase, the agent instantiates the abstract logic with concrete values to execute the precise tool call.

JSON Schema: Plan Operator

```
{
  "name": "plan",
  "description": "Decomposes complex objectives into actionable steps. Acts as 'Memory' to
anchor the search branch.",
  "parameters": {
    "type": "object",
    "properties": {
      "task_plan": {
        "type": "string",
        "description": "Structured breakdown: 1. Analysis; 2. Strategy; 3. Execution steps."
      },
      "priority_focus": {
        "type": "string",
        "description": "The single most critical aspect to prioritize in the immediate next
step."
      }
    },
    "required": ["task_plan"]
  }
}
```

JSON Schema: Reflection Operator (`reflection`)

```
{
  "name": "reflection",
  "description": "Critically evaluates the execution trace to identify logical flaws and
brainstorm pivots.",
  "parameters": {
    "type": "object",
    "properties": {
      "current_context": { "type": "string", "description": "Summary of observed state." },
      "critique": { "type": "string", "description": "Identification of potential flaws or edge
cases." },
      "alternative_ideas": { "type": "string", "description": "Proposed recovery paths if
current branch fails." }
    },
    "required": ["current_context", "critique", "alternative_ideas"]
  }
}
```

Table 11: **Meta-cognitive operator specifications derived from implementation.** The `task_decomposition` (top) enforces a three-stage structural prior (Analysis, Strategy, Execution), while `reflection` (bottom) implements a mandatory multi-perspective critique to bypass local optima during MCTS exploration.