SPIRAL: Symbolic LLM Planning via Grounded and Reflective Search

Yifan Zhang[1,2†], **Giridhar Ganapavarapu**[2], **Srideepika Jayaraman**[2], **Bhavna Agrawal**[2], **Dhaval Patel**[2], **Achille Fokoue**[2]

[1]Vanderbilt University, Nashville, TN, USA
[2]IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
{yifan.zhang.2}@vanderbilt.edu, {giridhar.ganapavarapu, j.srideepika}@ibm.com
{bhavna, pateldha, achille}@us.ibm.com

Abstract

Large Language Models (LLMs) often falter at complex planning tasks that require exploration and self-correction, as their linear reasoning process struggles to recover from early mistakes. While search algorithms like Monte Carlo Tree Search (MCTS) can explore alternatives, they are often ineffective when guided by sparse rewards and fail to leverage the rich semantic capabilities of LLMs. We introduce SPIRAL (Symbolic LLM Planning via Grounded and Reflective Search), a novel framework that embeds a cognitive architecture of three specialized LLM agents into an MCTS loop. SPIRAL's key contribution is its integrated planning pipeline where a Planner proposes creative next steps, a Simulator grounds the search by predicting realistic outcomes, and a Critic provides dense reward signals through reflection. This synergy transforms MCTS from a brute-force search into a guided, self-correcting reasoning process. On the DailyLifeAPIs and HuggingFace datasets, SPIRAL consistently outperforms the default Chain-of-Thought planning method and other state-of-the-art agents. More importantly, it substantially surpasses other state-of-the-art agents; for example, SPIRAL achieves 83.6% overall accuracy on DailyLifeAPIs, an improvement of over 16 percentage points against the next-best search framework, while also demonstrating superior token efficiency. Our work demonstrates that structuring LLM reasoning as a guided, reflective, and grounded search process yields more robust and efficient autonomous planners. The source code, full appendices, and all experimental data are available for reproducibility at the official project repository.

Code — https://github.com/IBM/SPIRAL

Introduction

The recent evolution of Large Language Models (LLMs) has marked a paradigm shift from pure text generation towards the development of autonomous agents capable of complex, goal-oriented behavior (Zhao et al. 2023). A key capability of these agents is their ability to formulate symbolic plans and interact with external tools to accomplish tasks (Schick et al. 2023). Foundational approaches for this planning process often rely on prompting the LLM to interleave reasoning "thoughts" with "actions" in a linear sequence (Wei et al. 2022; Yao et al. 2023b). However, the reliability of this autoregressive generation is a critical bottleneck; the plans produced are often brittle, where a single logical error can derail the entire process without a mechanism for structured deliberation or backtracking. Consequently, enhancing the robustness of LLM agents necessitates a move beyond simple linear prompting towards more sophisticated planning paradigms.

To address this challenge, integrating tree search frameworks has shown potential for adding structured exploration to the planning process (Yao et al. 2023a; Zhou et al. 2024). While MCTS (Silver et al. 2017) is a particularly compelling choice, applying it effectively presents significant challenges. Standard implementations often treat the LLM as a black-box policy with sparse rewards, leading to inefficient exploration (Zhang et al. 2024). Furthermore, without a grounding world model, agents may explore syntactically valid but practically nonsensical paths (Hao et al. 2023). This highlights a critical gap for a framework that synergizes MCTS's exploratory power with LLM semantics, guided by dense, semantic-aware feedback.

To overcome these limitations, we propose **SPIRAL** (Symbolic LLM Planning via Grounded and Reflective Search), a framework designed to address the strategic failures of linear planners. The motivation for our approach is illustrated in Figure 1. Faced with a simple conditional planning task, a standard "vanilla" agent follows a plausible but incorrect plan, failing to respect the user's constraints. In contrast, SPIRAL successfully navigates the task by employing a more deliberative process. It reframes LLM-driven MCTS by introducing a cognitive architecture of three specialized LLM agents that work in concert: a *Planner* to propose actions, a *Critic* to reflect on their strategic soundness, and a *Simulator* to ground the plan in plausible outcomes. The synergy between these agents transforms the search process, allowing SPIRAL to intelligently prune flawed reasoning paths and converge on robust solutions.

SPIRAL is built on the key insight that LLM-based planning can be made more robust by structuring it as a grounded and reflective search process. We achieve this by embedding three core principles within the MMCTS framework: 1) Decomposition, treating planning as a cognitive task fulfilled by specialized agentic roles; 2) Grounding, where a learned world model grounds exploration in plausible consequences;

[†]Work done during an AI research internship at IBM T.J. Watson Research Center.

Figure 1: A comparison of a Vanilla Agent and the SPIRAL Agent on a conditional planning task. The Vanilla Agent follows a linear but flawed plan, while SPIRAL uses its grounded and reflective search to correctly handle the weather-based constraint before acting.

and 3) Reflection, using a dense reward signal that evaluates strategic merit to replace sparse terminal rewards. To our knowledge, SPIRAL is the first framework to holistically integrate these principles within a formal search algorithm for LLM agents. Our approach substantially outperforms other state-of-the-art agents; on the DailyLifeAPIs of TaskBench, SPIRAL achieves 83.6% accuracy, surpassing the next-best search framework by over 16 percentage points.

Our main contributions are fourfold: 1) We propose **SPIRAL**, a novel framework that integrates a synergistic tri-agent cognitive architecture into Monte Carlo Tree Search to enable structured and dynamic planning. 2) We introduce its core mechanisms: a *Planner* to generate actions, a *Critic* to provide dense reflective feedback, and a *Simulator* to ground the search. Together, these components address the common MCTS challenges of sparse rewards and ungrounded exploration. 3) We evaluate SPIRAL on complex tool-use benchmarks, showing that it substantially outperforms both default planners and state-of-the-art frameworks like ReAct and LATS. 4) Finally, we demonstrate SPIRAL's superior resource efficiency and validate our architecture through ablation studies that confirm the critical role of each component.

Preliminaries

SPIRAL's core idea is to frame the complex task of LLM-based planning as a search process that is both grounded and reflective. To achieve this, the paper defines tool-use planning as a sequential decision-making problem within a Markov Decision Process (MDP) framework. The approach then uses the Monte Carlo Tree Search (MCTS) algorithm as its foundational search method.

LLM-based Planning as a Search Problem

We model the task of generating a multi-step tool-use plan as a search problem through a state space, formally defined as a sequential decision-making process by the tuple $(\mathcal{S}, \mathcal{A}, T, R)$. The *state space* $\mathcal{S}$ consists of states $s_t \in \mathcal{S}$, where each state is the history of actions and observations generated so far, often called a chain: $s_t = (a_0, o_1, \ldots, a_{t-1}, o_t)$. The *action space* $\mathcal{A}$ is the union of available tool calls, $\mathcal{A}_{\text{tool}}$, and a terminal action, $\mathcal{A}_{\text{term}} = \{\texttt{finish}\}$. The *transition function* $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ maps a state-action pair to a new state. Given a state s_t and an action a_t from a policy $\pi(a_t|s_t)$, we approximate this transition with a world model, $\mathcal{W}(o_{t+1}|s_t, a_t)$, which generates a plausible observation o_{t+1}. The subsequent state is then $s_{t+1} = s_t \oplus (a_t, o_{t+1})$. Finally, the *reward function* $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ provides feedback. A key challenge in this domain is that the reward signal is sparse; it is zero for all intermediate steps and non-zero only upon termination, where $R(s_T, a_T)$ evaluates the final plan's success.

The overall objective is to find an optimal policy, π^*, which generates a sequence of actions $(a_0, a_1, \ldots, a_T)$ that maximizes the expected cumulative reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} R(s_t, a_t)\right] \tag{1}$$

Monte Carlo Tree Search

MCTS is a heuristic search algorithm particularly well-suited for navigating the large decision spaces inherent in planning problems. The algorithm iteratively builds a search tree to balance the exploration of new paths with the exploitation of known promising ones. Each node n in the tree represents a state s_n and stores two key statistics: its total accumulated reward, or *value* v_n, and its *visit count* c_n.

A canonical MCTS iteration consists of four steps. The process begins with a *selection* phase, where the algorithm traverses the tree from the root by recursively selecting the child node i of a parent node p that maximizes the Upper Confidence Bound for Trees (UCT) score:

$$\text{UCT}_i = \frac{v_i}{c_i} + C\sqrt{\frac{\ln(c_p)}{c_i}} \tag{2}$$

where C is an exploration constant. Once a leaf node is reached, the *expansion* phase adds a new child node by taking a valid, unexplored action. From this new node, a *simulation* (or rollout) is performed, typically using random actions until a terminal state is reached, to produce an estimated value $\hat{v}$. Finally, in the *backpropagation* phase, this value is propagated up the tree to the root. For each ancestor node n_i on the path, the statistics are updated as $c_i \leftarrow c_i + 1$ and $v_i \leftarrow v_i + \hat{v}$. This update process ensures that the knowledge gained from the rollout is distributed to all relevant parent nodes in the search path. Over many iterations, this allows the UCT selection policy to become more accurate, effectively guiding the search towards the most promising regions of the decision space. Our work adapts this traditional MCTS framework by replacing its core components with specialized LLM agents, as detailed in the next section.

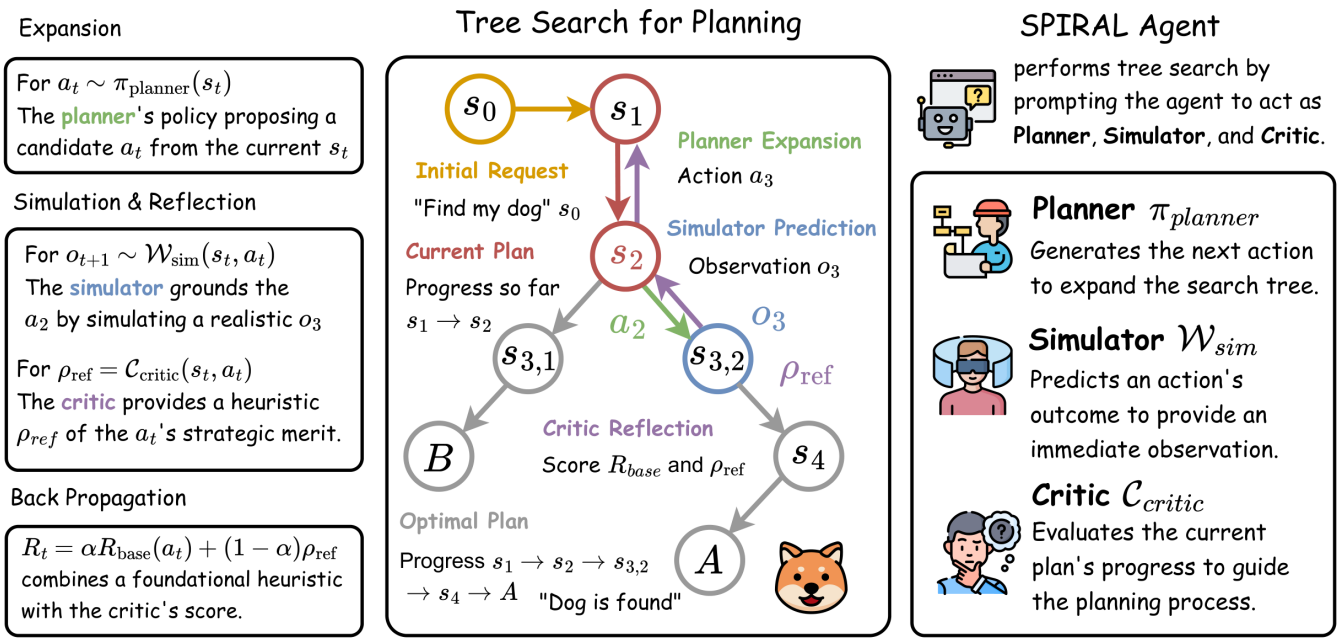

Figure 2: An overview of the SPIRAL framework, where a tri-LLM symbolic architecture drives the MCTS loop. **(1) Expansion:** The Planner proposes a new action (a_2) to expand the search from the current state (s_2). **(2) Simulation & Reflection:** The Simulator provides a grounded observation (o_3) for the action, while the Critic generates a strategic reflection score (ρ_{ref}) evaluating the action's merit. **(3) Backpropagation:** A composite reward, calculated using the Critic's score, is then propagated up the tree to update node values and intelligently guide future selections.

The SPIRAL Framework

To overcome the limitations of sparse rewards and unguided exploration, we introduce the SPIRAL framework (Figure 2). Our approach integrates a multi-agent cognitive architecture into MCTS, decomposing planning into specialized roles for proposing, grounding, and evaluating actions. We detail this architecture, the reward shaping mechanism, and the search algorithm below.

SPIRAL's Cognitive Architecture

SPIRAL's design decomposes planning into specialized cognitive roles, efficiently instantiating a *Planner*, *Simulator*, and *Critic* within a single LLM (Figure 2). This creates a robust, self-correcting process that leverages in-context learning, allowing SPIRAL to adapt to new problems without fine-tuning.

Agent Roles and Functionality

Each agent in SPIRAL's cognitive architecture is responsible for a specific function within the MCTS search, corresponding to the roles and prompts outlined in Figure 2.

The Planner (π_{planner}). The Planner drives the **Expansion** phase of the search and acts as the creative strategist. Given the current problem description and the action-observation history, the Planner's function is to generate the next candidate action to expand the search tree. As the exploration policy of the framework, it is designed to propose diverse and contextually relevant next steps, providing the raw material for the search process.

The Simulator ($\mathcal{W}_{\text{sim}}$). The Simulator functions as a learned world model to ground the search process during the **Simulation & Reflection** phase. When the Planner proposes an action, the Simulator's role is to predict a plausible, natural language observation that would result from executing that action. This grounds the search in realistic consequences, allowing the agent to plan based on the likely outcomes of its actions rather than operating in a vacuum of pure reason.

The Critic ($\mathcal{C}_{\text{critic}}$). The Critic serves as the logical evaluator during the **Simulation & Reflection** phase, providing dense, strategic feedback. It assesses the strategic merit of the Planner's action, producing a quantitative reflection score, ρ_{ref}. This score is combined with a foundational heuristic, R_{base}, to form a composite reward, R_t. This reward is used during backpropagation to guide the search toward strategically sound plans, directly addressing the MCTS challenge of sparse rewards. This use of a dense, semantic score for backpropagation aligns with recent work on using semantic feedback to guide reasoning, e.g., TextGrad and CodeGrad (Yuksekgonul et al. 2024; Zhang et al. 2025).

Reflection-Driven Reward Shaping

To address sparse rewards, SPIRAL introduces Reflection-Driven Reward Shaping, generating a dense, semantic-aware signal R_t for use during **Backpropagation**. Unlike outcome-based rewards, R_t is calculated immediately for each expanded node as $R_t = \alpha R_{\text{base}}(a_t) + (1-\alpha)\rho_{\text{ref}}$, blending a validity heuristic R_{base} with the Critic's strategic score

Approach	TU	ME	SF	SR	DU
CoT (Wei et al. 2022)	×	✓	×	×	×
ReAct (Yao et al. 2023b)	✓	×	✓	×	×
Reflexion (Shinn et al. 2023)	✓	×	✓	×	×
MCTS (Świechowski et al. 2023)	✓	✓	×	×	×
ToT (Yao et al. 2023a)	×	✓	×	✓	✓
RAFA (Liu et al. 2024b)	✓	✓	✓	✓	×
LATS (Zhou et al. 2024)	✓	✓	✓	✓	×
SPIRAL (Ours)	✓	✓	✓	✓	✓

Table 1: Comparison of agent frameworks. Columns: Tool Use (TU), Multi-path Exploration (ME), Step-wise Feedback (SF), Strategic Reflection (SR), and Dynamic Policy Update (DU). SPIRAL unifies these capabilities.

ρ_{ref} via hyperparameter $\alpha \in [0, 1]$. Crucially, this formulation converts qualitative semantic evaluations into a numerical scalar, allowing the algorithm to directly apply standard MCTS backpropagation updates to the value (v) and visit count (c) statistics of ancestor nodes. This guides the search toward paths that are both valid and strategically sound, significantly improving efficiency.

Grounded and Reflective Tree Search

The SPIRAL framework culminates in the Grounded and Reflective Tree Search algorithm, which integrates the cognitive architecture and the reward shaping mechanism into the four canonical stages of MCTS. The full process for each iteration is detailed below.

Selection. The algorithm begins by traversing the existing search tree from the root. At each level, it recursively selects the child node that maximizes the Upper Confidence Bound for Trees (UCT) score (Equation 2), balancing exploitation of known high-value paths with exploration of less-visited ones. This process continues until a leaf node n_L is reached, identifying a promising frontier for expanding the search.

Expansion. Upon reaching a leaf node n_L, the *Planner* (π_{planner}) leverages the accumulated interaction history to generate a single candidate action a_L. This proposal creates a new child node, extending the search frontier with a diverse and contextually relevant step that logically progresses the plan.

Simulation & Reflection. This stage replaces the expensive, noisy random rollout of MCTS with a deterministic one-step lookahead that is grounded and semantically evaluated. First, the *Simulator* ($\mathcal{W}_{\text{sim}}$) grounds the action a_L by generating a realistic observation o_{L+1} to form a new state s_{L+1}. Concurrently, the *Critic* ($\mathcal{C}_{\text{critic}}$) performs reflection, evaluating the action's strategic merit to produce the reflective score ρ_{ref}.

Backpropagation. This final phase uses Reflection-Driven Reward Shaping to calculate a composite reward (R_t) for the new node using the semantic formulation defined earlier. This reward is then backpropagated up the path to the root, updating the value (v) and visit count (c) of each ancestor. Propagating a reward based on strategic merit, rather than a noisy outcome, makes the tree's stored values more reliable indicators of a plan's intrinsic quality, leading to a more intelligent and efficient search.

As summarized in Table 1, SPIRAL synthesizes capabilities prior frameworks only partially address, integrating multi-path exploration with tool use. It uniquely enriches this search with two forms of dense, internal feedback: step-wise grounding from the Simulator and strategic guidance from the Critic. The backpropagation of this reflective feedback enables a dynamic policy update at each step, yielding a more robust, deliberative, and adaptive planner. The complete pseudocode is available in Appendix A.

Experimental Setup

We comprehensively evaluate SPIRAL's performance, efficiency, and robustness through three main analyses: comparing against standard Chain-of-Thought (CoT) baselines, benchmarking against state-of-the-art agent frameworks, and conducting a detailed ablation study to validate our design. Collectively, these experiments are designed to isolate the impact of our grounded, reflective search mechanism on both solution quality and computational cost.

Datasets and Tasks

Our evaluation uses two benchmarks for complex, multi-step tool utilization from the TaskBench suite (Shen et al. 2024): 'dailylifeapis' and 'huggingface'. To ensure a robust and reproducible evaluation, all experiments are conducted using five fixed random seeds, and we report the *mean and standard deviation* across these runs. Appendix C details the data sampling and pre-processing methodology used in the experiments.

Implementation and Evaluation

All agents are implemented using a suite of state-of-the-art models: DeepSeek-V2.5, Llama 3.3 70B, Llama 4 Maverick 17B, Phi 4 14B, and Qwen 2.5 72B. Model interactions were conducted via a dedicated internal research cluster (see Appendix D for details) and orchestrated using the LangChain library. Our SPIRAL agent uses a Planner temperature of 0.1, an MCTS budget of 50 iterations, an exploration constant $C = 1.5$, and a reward shaping hyperparameter $\alpha = 0.5$. We include a brief sensitivity analysis for key hyperparameters, such as the MCTS budget and the reward shaping coefficient α, in Appendix B, which confirms the robustness of our chosen configuration. The baseline frameworks, whose capabilities are compared in Table 1, are configured for a strong and fair comparison, with full details available in our experiment scripts.

Evaluation Metrics. Our primary evaluation metric is Success Rate, which measures the percentage of tasks correctly solved. To provide a more nuanced analysis, we report this metric across three categories: Overall Accuracy, Simple Task Accuracy on single-step plans, and Complex Task Accuracy on multi-step plans. To analyze resource costs, we also report two efficiency metrics: Token Efficiency, defined as the success rate per 10,000 tokens consumed, and API Call Efficiency, defined as the success rate per LLM call made during the planning process.

Model	Method	DailyLifeAPIs			HuggingFace		
		Simple Acc. (%)	Complex Acc. (%)	Overall Acc. (%)	Simple Acc. (%)	Complex Acc. (%)	Overall Acc. (%)
DeepSeek-V2.5	CoT (k=1)	85.12 ±6.42	59.22 ±5.86	66.60 ±4.90	93.84 ±0.97	67.92 ±1.97	75.77 ±1.57
	CoT (k=3)	86.04 ±4.09	60.91 ±3.83	67.90 ±3.87	94.98 ±1.35	71.94 ±3.06	79.34 ±2.42
	CoT (k=5)	84.81 ±2.06	62.41 ±5.34	68.83 ±4.19	94.44 ±1.34	71.16 ±1.46	78.61 ±1.40
	SPIRAL	**94.33** ±3.95	**89.89** ±3.57	**91.24** ±2.65	**98.89** ±0.91	**95.81** ±1.23	**96.84** ±0.65
Llama 3.3 70B	CoT (k=1)	94.97 ±1.85	94.68 ±2.67	94.87 ±1.80	93.10 ±0.71	92.18 ±1.67	92.48 ±1.29
	CoT (k=3)	95.26 ±1.73	94.82 ±1.58	94.88 ±1.36	93.85 ±1.40	92.29 ±1.41	92.79 ±1.24
	CoT (k=5)	95.54 ±3.63	94.06 ±2.08	94.38 ±1.88	93.77 ±0.97	93.75 ±0.67	93.75 ±0.68
	SPIRAL	**95.79** ±2.44	**98.82** ±0.82	**98.35** ±0.83	**99.08** ±0.90	**96.76** ±1.25	**97.44** ±0.89
Llama 4 Maverick 17B	CoT (k=1)	84.43 ±6.19	47.59 ±4.73	57.95 ±5.59	91.81 ±1.15	67.78 ±1.80	76.43 ±0.97
	CoT (k=3)	87.72 ±5.09	50.21 ±5.35	60.60 ±4.16	91.67 ±2.40	66.24 ±1.82	75.65 ±1.31
	CoT (k=5)	89.21 ±3.95	48.86 ±6.53	60.00 ±5.31	90.88 ±2.17	68.34 ±0.83	77.09 ±0.76
	SPIRAL	**93.64** ±2.96	**79.53** ±4.44	**83.31** ±4.11	**93.63** ±1.92	**92.94** ±0.56	**93.04** ±0.89
Phi 4 14B	CoT (k=1)	89.51 ±3.34	83.88 ±2.54	85.67 ±2.67	95.27 ±1.20	90.38 ±0.75	92.08 ±0.26
	CoT (k=3)	89.81 ±4.86	85.19 ±4.00	86.24 ±3.61	96.86 ±1.55	90.72 ±1.59	92.84 ±0.95
	CoT (k=5)	91.63 ±1.65	84.27 ±4.38	86.45 ±3.44	**97.49** ±1.06	91.72 ±2.10	93.71 ±1.13
	SPIRAL	**95.48** ±3.63	**90.53** ±2.25	**91.57** ±2.44	96.67 ±1.83	**94.95** ±0.82	**95.48** ±0.86
Qwen 2.5 72B	CoT (k=1)	90.45 ±4.30	89.25 ±2.10	89.70 ±2.03	90.51 ±2.33	85.05 ±1.85	86.78 ±1.58
	CoT (k=3)	91.01 ±3.33	89.02 ±1.80	89.64 ±1.14	91.45 ±3.79	86.57 ±1.80	88.16 ±1.11
	CoT (k=5)	92.10 ±3.39	88.47 ±1.21	89.50 ±1.37	91.50 ±2.61	87.09 ±2.00	88.47 ±1.65
	SPIRAL	**94.09** ±3.94	**100.00** ±0.00	**97.69** ±1.23	**97.73** ±1.06	**98.52** ±0.49	**97.08** ±0.54

Table 2: Performance comparison of CoT and our proposed SPIRAL method. We report the mean ± standard deviation over 5 runs with fixed seeds. The results for our **SPIRAL** method are highlighted unless a baseline performs better. **Simple Acc.**: Simple Task Accuracy (%), **Complex Acc.**: Complex Task Accuracy (%), **Overall Acc.**: Overall Accuracy (%). For CoT, k refers to the number of self-consistency levels used. Full hyperparameter details for all methods are listed in Appendix B.

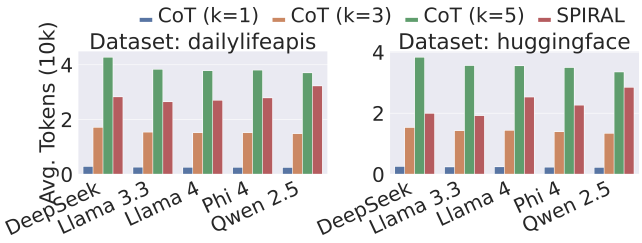

Figure 3: Comparison of average token usage per task across different models and methods. Our SPIRAL method consistently reduces token costs compared to CoT baselines on both datasets.

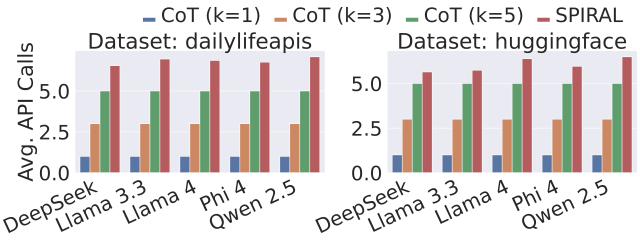

Figure 4: Comparison of average API call usage per task across different models and methods. SPIRAL requires more API calls, reflecting its more complex reasoning process.

Results and Analysis

This section empirically evaluates SPIRAL through comparisons against CoT baselines and state-of-the-art frameworks, followed by an ablation study. Together, these analyses confirm SPIRAL's robustness in complex planning and validate the necessity of its cognitive architecture.

Baseline Performance Analysis

We begin by comparing SPIRAL against CoT baselines (Table 2), which SPIRAL consistently outperforms, with the performance gap most pronounced on *complex tasks* requiring backtracking. For instance, on the DailyLifeAPIs benchmark with Llama 4 Maverick 17B, SPIRAL's accuracy on complex tasks (79.53%) is over 29 percentage points higher than the best CoT variant (50.21% for k=3). This advantage stems from SPIRAL's MCTS-based exploration, which navigates conditional logic and recovers from mistakes where linear CoT plans fail.

Cost-Benefit Analysis. While superior in accuracy, we also analyze SPIRAL's operational cost (Figure 3 and Figure 4). As expected from its deliberative design, SPIRAL uses more API (LLM) calls than CoT methods. However, this increased interaction leads to a more focused search, making it remarkably token-efficient; it consistently consumes fewer total tokens than the more effective CoT baselines (k=3 and k=5). This highlights a key trade-off: SPI-

Model	Method	DailyLifeAPIs			HuggingFace		
		Simp. Acc. (%)	Comp. Acc. (%)	Overall Acc. (%)	Simp. Acc. (%)	Comp. Acc. (%)	Overall Acc. (%)
Llama 4 Maverick 17B	ReAct (Yao et al. 2023b)	87.79 ±6.20	51.38 ±6.20	61.60 ±6.92	94.81 ±0.86	70.95 ±1.86	79.54 ±1.15
	RAFA (Liu et al. 2024b)	88.90 ±6.76	53.83 ±4.83	63.75 ±6.24	96.32 ±0.73	73.75 ±1.88	81.86 ±1.28
	ReAct+RAFA	90.19 ±4.13	54.94 ±4.55	64.90 ±5.32	97.47 ±0.60	75.56 ±1.90	83.43 ±1.18
	LATS (Zhou et al. 2024)	91.48 ±5.08	57.93 ±3.62	67.39 ±4.74	98.28 ±0.76	78.60 ±1.30	85.67 ±0.91
	SPIRAL	**95.77** ±2.94	**78.79** ±3.50	**83.61** ±3.10	**99.06** ±0.89	**88.38** ±8.76	**92.18** ±5.98
Phi 4 14B	ReAct (Yao et al. 2023b)	90.13 ±2.26	85.12 ±2.64	86.67 ±2.58	96.40 ±0.67	90.83 ±1.07	92.77 ±0.77
	RAFA (Liu et al. 2024b)	90.13 ±2.26	85.36 ±2.63	86.84 ±2.60	97.26 ±1.43	91.81 ±1.09	93.70 ±0.87
	ReAct+RAFA	91.36 ±0.66	85.36 ±2.63	87.17 ±2.20	97.38 ±1.44	92.31 ±1.11	94.07 ±0.73
	LATS (Zhou et al. 2024)	92.61 ±2.44	85.82 ±2.55	87.84 ±2.19	98.74 ±0.92	92.94 ±1.04	94.96 ±0.86
	SPIRAL	**97.81** ±2.20	**95.48** ±1.98	**96.17** ±1.61	**99.64** ±0.33	**99.00** ±0.26	**99.23** ±0.09
Qwen 2.5 72B	ReAct (Yao et al. 2023b)	94.34 ±4.15	94.86 ±1.49	94.77 ±0.74	98.93 ±1.03	96.94 ±0.69	97.57 ±0.55
	RAFA (Liu et al. 2024b)	96.20 ±3.03	95.57 ±2.02	95.78 ±1.35	99.60 ±0.61	97.84 ±0.57	98.40 ±0.51
	ReAct+RAFA	96.96 ±2.16	96.04 ±2.21	96.29 ±1.66	99.60 ±0.61	98.34 ±0.31	98.74 ±0.28
	LATS (Zhou et al. 2024)	96.96 ±2.16	96.26 ±1.88	96.46 ±1.51	99.60 ±0.61	98.59 ±0.56	98.92 ±0.31
	SPIRAL	**98.79** ±1.76	**100.00** ±0.00	**99.67** ±0.46	**99.86** ±0.31	**99.81** ±0.18	**99.83** ±0.18

Table 3: Cascaded accuracy comparison on Llama 4, Phi 4, and Qwen 2.5. Each method was applied to the failures of a CoT (k=1) baseline. We report the mean ± standard deviation over 5 runs with fixed seeds. The best result for each metric is highlighted in **bold**. **Simp. Acc.**: Simple Task Acc. (%), **Comp. Acc.**: Complex Task Acc. (%), **Overall Acc.**: Overall Accuracy (%). Full hyperparameter details for all methods are listed in Appendix B.

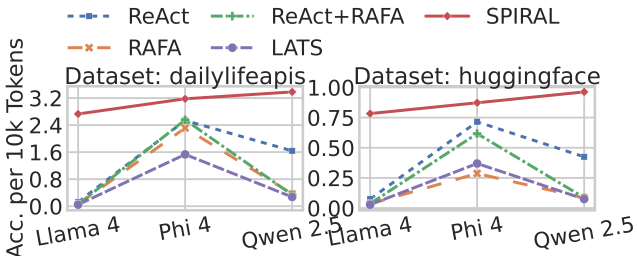

Figure 5: Token Efficiency of SOTA methods. This plot shows the final cascaded accuracy achieved per 10,000 tokens consumed. A higher value indicates greater efficiency.

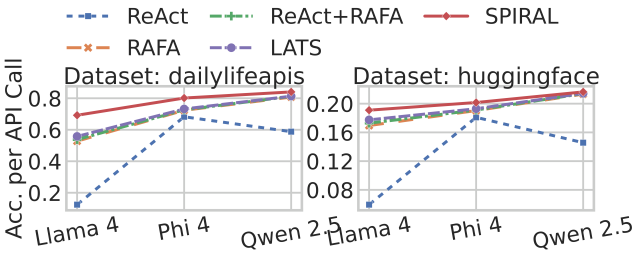

Figure 6: API Call Efficiency of SOTA methods. This plot shows the final cascaded accuracy achieved per API call, measuring the effectiveness of each interaction.

RAL's higher number of targeted calls is a more resource-effective strategy than generating fewer, but much longer and often redundant, reasoning paths. While this deliberative search incurs higher wall-clock latency (detailed in Appendix H), our analysis shows that each API call is used with greater strategic purpose, leading to the highest accuracy per interaction (Figure 4).

Comparison with State-of-the-Art Methods

We benchmark SPIRAL against state-of-the-art frameworks, including ReAct, RAFA, LATS, and Tree of Thoughts (ToT), on the challenging set of problems where a baseline CoT agent fails. This cascaded setup rigorously tests each framework's advanced problem-solving capabilities. The results, detailed in Table 3 and Table 4, demonstrate SPIRAL's superior performance. For instance, on the DailyLifeAPIs dataset using Llama 4 Maverick 17B, SPIRAL achieves an overall cascaded accuracy of 83.61%, significantly outperforming the next-best method, LATS (67.39%), and substantially surpassing ToT (61%).

Resource Efficiency. Beyond raw accuracy, SPIRAL demonstrates superior resource efficiency, achieving the highest accuracy per token and API call (Figure 5 and Figure 6). Although SPIRAL uses more API calls than simple CoT, its guided search ensures each interaction serves a greater strategic purpose than competing frameworks. This confirms that SPIRAL's performance stems from an efficient cognitive architecture rather than brute force.

Method	DailyLifeAPIs (%)	HuggingFace (%)
ToT (S=4, B=3, C=2)	60.99 ±4.11	82.50 ±0.87
ToT (S=5, B=4, C=3)	60.99 ±4.87	82.54 ±0.58
ToT (S=6, B=5, C=4)	60.83 ±4.12	82.21 ±1.48
SPIRAL	**80.83** ±3.48	**88.46** ±10.03

Table 4: Cascaded accuracy on Llama 4 Maverick 17B using residual methods. For ToT, configurations are (S=steps, B=breadth, C=candidates). Best results are in **bold**.

Model	Method	DailyLifeAPIs	HuggingFace
Llama 3.3 70B	MCTS (N=15)	87.60 ±1.31	89.68 ±1.43
	MCTS (N=30)	87.11 ±1.72	89.12 ±0.95
	MCTS (N=50)	86.28 ±1.71	89.44 ±1.03
	w/o Planner	94.71 ±1.90	87.48 ±1.42
	w/o Simulator	92.89 ±1.11	69.48 ±3.02
	w/o Validator	87.27 ±2.38	90.08 ±1.30
	w/ Uniform Rewards	88.27 ±1.79	89.12 ±1.06
	SPIRAL	**98.35** ±0.83	**97.44** ±0.89
Llama 4 Maverick 17B	MCTS (N=15)	79.84 ±4.66	73.80 ±1.39
	MCTS (N=30)	79.83 ±4.77	73.64 ±1.51
	MCTS (N=50)	80.16 ±4.21	73.44 ±2.85
	w/o Planner	81.82 ±2.41	91.08 ±1.46
	w/o Simulator	80.99 ±4.89	55.28 ±2.87
	w/o Validator	80.00 ±4.31	75.16 ±2.46
	w/ Uniform Rewards	79.01 ±5.31	74.44 ±2.06
	SPIRAL	**83.30** ±4.11	**93.04** ±0.89

Table 5: Ablation study of SPIRAL components on Llama 3.3 70B and Llama 4. For MCTS baselines, 'N'=search iterations. Best results are in **bold**. Full hyperparameter details for all methods are listed in Appendix B.

Ablation Study

To validate our design, we conducted an ablation study, summarized in Table 5. The study confirms each component of the cognitive architecture is critical, with performance degrading most when the Simulator and Critic are removed. Disabling the Simulator ('w/o Simulator') causes a catastrophic drop in accuracy (e.g., from 97.44% to 69.48% on HuggingFace), underscoring the necessity of a grounded world model. Similarly, disabling the Critic's dense feedback ('w/ Uniform Rewards') significantly impairs performance, validating our reflection-driven approach. Removing the Planner and Validator also results in a noticeable performance decrease. Crucially, the SPIRAL framework outperforms a well-budgeted standard MCTS baseline, confirming the synergy between agents is the primary driver of success, as illustrated by the case studies and failure analysis in Appendix G.

Related Works

Our work is situated at the confluence of research in LLM reasoning and autonomous agent architectures. The rapid growth of this field is captured in several comprehensive surveys on LLMs and autonomous agents (Zhao et al. 2023; WANG et al. 2024).

Reasoning and Planning with Search

Recent literature has focused on evolving LLM reasoning from simple linear generation to more robust, structured problem-solving. Initial breakthroughs in eliciting step-by-step reasoning via prompting (Wei et al. 2022; Wang et al. 2023c) have been extended by advanced decomposition strategies that separate planning from execution (Wang et al. 2023b; Zhou et al. 2022; Press et al. 2023). However, the logical consistency and faithfulness of these generated plans remains a key challenge (Lanham et al. 2023; Creswell and Shanahan 2022; Dhuliawala et al. 2024), motivating a shift towards formal search algorithms. This is particularly crucial as recent work continues to investigate the specific failure points of LLMs on long-horizon and classical planning tasks (Kambhampati et al. 2024). While investigations have highlighted the planning deficiencies of LLMs (Valmeekam et al. 2023), a dominant theme has been the application of tree search to navigate the vast space of possibilities. Frameworks based on deliberate exploration, such as Tree of Thoughts (Yao et al. 2023a), and specifically MCTS (Świechowski et al. 2023; Silver et al. 2017), have become a cornerstone of modern agent planning (Zhang et al. 2024; Zhou et al. 2024; Song et al. 2023), often contrasting with classical formal planning methods (Li et al. 2024). This informs SPIRAL's choice of an MCTS-driven framework.

Architectures for Autonomous Agents

A major research thrust focuses on architecting LLMs into autonomous agents. A foundational theme is the move from static reasoning to dynamic interaction, where agents interleave thought and action to engage with an environment (Yao et al. 2023b), often specializing in domains like embodied control (Huang et al. 2022; Wang et al. 2023a) or web navigation (Nakano et al. 2021). The primary mechanism for this interaction is the use of external tools and APIs, a rich ecosystem with foundational work on enabling models to use tools (Schick et al. 2023; Patil et al. 2024), the development of tool-specific models (Qin et al. 2023), and findings that executable code actions improve reliability (Wang et al. 2024). For these agents to operate reliably, they require mechanisms to ground their plans in plausible reality, using the LLM as a world model (Hao et al. 2023) or employing specialized decoding strategies (Huang et al. 2023). Furthermore, agents are increasingly designed for self-correction through feedback, including verbal reinforcement (Shinn et al. 2023), iterative refinement (Madaan et al. 2023), and interactive critiquing (Gou et al. 2023; Liu et al. 2024b; Zhang and Leach 2025). The growing complexity of these architectures, including multi-agent systems (Wu et al. 2024; Zhang and Yang 2025; Qian et al. 2024; Cemri et al. 2025), has also necessitated the creation of comprehensive benchmarks for standardized evaluation (Shen et al. 2024; Liu et al. 2024a; Zhou et al. 2023; Yang et al. 2025; Li et al. 2023; Hendrycks et al. 2021; Jimenez et al. 2024).

Conclusion and Future Work

We introduced SPIRAL, embedding a tri-agent cognitive architecture (Planner, Critic, Simulator) within MCTS to unify grounded search with strategic reflection, addressing the limitations of linear reasoning and sparse rewards. Our evaluation demonstrates that this synergy of decomposition, grounding, and reflection substantially outperforms CoT and SOTA frameworks in accuracy and resource efficiency. Future work includes improving search efficiency (e.g., via pruning or distillation), enabling self-improvement, and extending the framework to complex domains like robotics. All artifacts are available at the project repository.[1]

[1]https://github.com/IBM/SPIRAL

Acknowledgments

This project was conducted at the IBM T.J. Watson Research Center[2] with support from IBM Research[3]. We also thank the IBM WatsonX team[4] for providing platform and compute resources, including access to WatsonX.ai[5] for managed model serving and governed experimentation. Platform support enabled reproducible, at-scale experiments. Looking ahead, we expect the methods introduced here to inform future WatsonX capabilities for governed, scalable automation in enterprise AI workflows.

References

Cemri, M.; Pan, M. Z.; Yang, S.; Agrawal, L. A.; Chopra, B.; Tiwari, R.; Keutzer, K.; Parameswaran, A.; Klein, D.; Ramchandran, K.; et al. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.

Creswell, A.; and Shanahan, M. 2022. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*.

Dhuliawala, S.; Komeili, M.; Xu, J.; Raileanu, R.; Li, X.; Celikyilmaz, A.; and Weston, J. 2024. Chain-of-Verification Reduces Hallucination in Large Language Models. In *Findings of the Association for Computational Linguistics ACL 2024*, 3563–3578.

Gou, Z.; Shao, Z.; Gong, Y.; Yang, Y.; Duan, N.; Chen, W.; et al. 2023. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. In *The Twelfth International Conference on Learning Representations*.

Hao, S.; Gu, Y.; Ma, H.; Hong, J.; Wang, Z.; Wang, D.; and Hu, Z. 2023. Reasoning with Language Model is Planning with World Model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, 9118–9147. PMLR.

Huang, W.; Xia, F.; Shah, D.; Driess, D.; Zeng, A.; Lu, Y.; Florence, P.; Mordatch, I.; Levine, S.; Hausman, K.; et al. 2023. Grounded decoding: Guiding text generation with grounded models for embodied agents. *Advances in Neural Information Processing Systems*, 36: 59636–59661.

Jimenez, C. E.; Yang, J.; Wettig, A.; Yao, S.; Pei, K.; Press, O.; and Narasimhan, K. 2024. SWE-bench: Can Language Models Resolve Real-world Github Issues? In *The Twelfth International Conference on Learning Representations*.

[2]https://research.ibm.com/labs/yorktown-heights

[3]https://research.ibm.com

[4]https://www.ibm.com/watsonx

[5]https://www.ibm.com/products/watsonx-ai

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L.; and Murthy, A. 2024. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*.

Lanham, T.; Chen, A.; Radhakrishnan, A.; Steiner, B.; Denison, C.; Hernandez, D.; Li, D.; Durmus, E.; Hubinger, E.; Kernion, J.; et al. 2023. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*.

Li, M.; Zhao, Y.; Yu, B.; Song, F.; Li, H.; Yu, H.; Li, Z.; Huang, F.; and Li, Y. 2023. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Li, Z.; Hua, W.; Wang, H.; Zhu, H.; and Zhang, Y. 2024. Formal-llm: Integrating formal language and natural language for controllable llm-based agents. *arXiv preprint arXiv:2402.00798*.

Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; et al. 2024a. AgentBench: Evaluating LLMs as Agents. In *ICLR*.

Liu, Z.; Hu, H.; Zhang, S.; Guo, H.; Ke, S.; Liu, B.; and Wang, Z. 2024b. Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency. *arXiv preprint arXiv:2309.17382*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36: 46534–46594.

Nakano, R.; Hilton, J.; Balaji, S.; Wu, J.; Ouyang, L.; Kim, C.; Hesse, C.; Jain, S.; Kosaraju, V.; Saunders, W.; et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Patil, S. G.; Zhang, T.; Wang, X.; and Gonzalez, J. E. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565.

Press, O.; Zhang, M.; Min, S.; Schmidt, L.; Smith, N. A.; and Lewis, M. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Qian, C.; Liu, W.; Liu, H.; Chen, N.; Dang, Y.; Li, J.; Yang, C.; Chen, W.; Su, Y.; Cong, X.; et al. 2024. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 15174–15186.

Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language models can teach

themselves to use tools. *Advances in Neural Information Processing Systems*, 36: 68539–68551.

Shen, Y.; Song, K.; Tan, X.; Zhang, W.; Ren, K.; Yuan, S.; Lu, W.; Li, D.; and Zhuang, Y. 2024. Taskbench: Benchmarking large language models for task automation. *Advances in Neural Information Processing Systems*, 37: 4540–4574.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36: 8634–8652.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Song, C. H.; Wu, J.; Washington, C.; Sadler, B. M.; Chao, W.-L.; and Su, Y. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2998–3009.

Świechowski, M.; Godlewski, K.; Sawicki, B.; and Mańdziuk, J. 2023. Monte Carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3): 2497–2562.

Valmeekam, K.; Marquez, M.; Sreedharan, S.; and Kambhampati, S. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36: 75993–76005.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023a. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*.

WANG, L.; MA, C.; FENG, X.; ZHANG, Z.; YANG, H.; ZHANG, J.; CHEN, Z.; TANG, J.; CHEN, X.; LIN, Y.; et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345.

Wang, L.; Xu, W.; Lan, Y.; Hu, Z.; Lan, Y.; Lee, R. K.-W.; and Lim, E.-P. 2023b. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

Wang, X.; Chen, Y.; Yuan, L.; Zhang, Y.; Li, Y.; Peng, H.; and Ji, H. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q. V.; Chi, E. H.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023c. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; and Zhou, D. 2022. Chain-of-thought prompting elicits reasoning in large language models. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 24824–24837. Curran Associates, Inc.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Yang, X.; Liu, Z.; Huang, C.; Zhang, J.; Zhang, T.; Zhang, Y.; and Lei, W. 2025. ELABORATION: A Comprehensive Benchmark on Human-LLM Competitive Programming. *arXiv preprint arXiv:2505.16667*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yuksekgonul, M.; Bianchi, F.; Boen, J.; Liu, S.; Huang, Z.; Guestrin, C.; and Zou, J. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Zhang, D.; Zhoubian, S.; Hu, Z.; Yue, Y.; Dong, Y.; and Tang, J. 2024. ReST-MCTS* LLM self-training via process reward guided tree search. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 64735–64772.

Zhang, Y.; and Leach, K. 2025. Leveraging Human Insights for Enhanced LLM-based Code Repair. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, 1536–1537.

Zhang, Y.; and Yang, X. 2025. CONSTRUCTA: Automating Commercial Construction Schedules in Fabrication Facilities with Large Language Models. *arXiv preprint arXiv:2502.12066*.

Zhang, Y.; Zhang, Y.; Leach, K.; and Huang, Y. 2025. CodeGrad: Integrating Multi-Step Verification with Gradient-Based LLM Refinement. *arXiv preprint arXiv:2508.10059*.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223*.

Zhou, A.; Yan, K.; Shlapentokh-Rothman, M.; Wang, H.; and Wang, Y.-X. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, 62138–62160.

Zhou, D.; Schärli, N.; Hou, L.; Wei, J.; Scales, N.; Wang, X.; Schuurmans, D.; Cui, C.; Bousquet, O.; Le, Q. V.; et al. 2022. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. 2023. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *The Twelfth International Conference on Learning Representations*.