ToolTree: Deliberate Tool Selection for LLM Agents via Monte Carlo Tree Search

Anonymous ACL submission

Abstract

Large Language Model (LLM) agents are increasingly applied to complex, multi-step tasks that require interaction with diverse external tools across domains such as mathematics, vision, and knowledge retrieval. However, current frameworks typically rely on greedy, reactive tool selection strategies that lack foresight and fail to account for inter-tool dependencies. In this paper, we present ToolTree, a generalizable agent framework that integrates a plug-and-play Monte Carlo Tree Search (MCTS) module for deliberate tool selection. ToolTree explores possible tool usage trajectories using a dual-stage LLM evaluation mechanism that enables the agent to make informed, adaptive decisions over extended tool-use sequences. To ensure broad applicability, we introduce standardized "tool library" that encapsulate domain-specific models, enabling seamless orchestration across multiple domains. Empirical evaluations across 15 tasks demonstrate that ToolTree consistently improves downstream performance, achieving an average gain of over 5% compared to state-of-the-art agent systems.

1 Introduction

Recent advancements in Large Language Models (LLMs) (Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023) have propelled the emergence of autonomous LLM agents capable of tackling complex tasks across domains such as software engineering (Yang et al., 2024), web interaction (Zhou et al., 2023), computer use (Xie et al., 2024) and multimodal understanding (Wu et al., 2023). These agents leverage LLMs' capabilities in planning (Hao et al., 2023), reasoning (Wei et al., 2022), and knowledge representation (Gu et al., 2024), and have shown impressive performance in open-ended, multi-step problem-solving scenarios. As these agents become more sophisticated, the integration of external tools has become a crucial mechanism for extending their functional capabilities beyond the limits of their pre-trained knowledge.

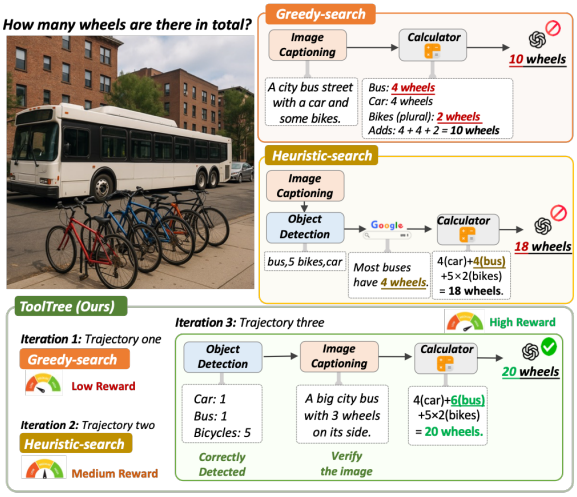


Figure 1: Comparison of ToolTree with greedy search and linear heuristic search. A greedy search with caption, calculator pipeline hallucinates default counts and answers 10, while a linear heuristic search with caption, detector, web lookup and calculator chain still fails with answer 18. ToolTree chooses the optimal tool trajectory and answer correctly with 20.

A core requirement for effective LLM agents lies in their ability to interact with external tools to retrieve up-to-date information, executing computations, or invoking specialized models (Schick et al., 2023; Qin et al., 2023). However, simply enabling tool use is not enough. Effective orchestration that determines which tools to use, in what order, and how their outputs influence subsequent decisions is critical. This orchestration becomes especially challenging when tools exhibit inter dependencies or when task success relies on the cumulative outcome of a tool sequence. As illustrated in Figure 1, even a seemingly simple counting-wheel question requiring multi-tool interaction can lead to incorrect outcomes when agents adopt reactive or insufficiently planned strategies. Thus, developing agents that can robustly reason over such sequential

Feature	Ours	General LLM Agent Framework		Tool Augmented LLM System			LLM Agent Tree Search		
		GPT-Functions (OpenAI, 2024)	OctoTools (Lu et al., 2025)	HuggingGPT (Shen et al., 2023)	ToolChain* (Zhuang et al., 2024)	ToolPlanner (Liu et al., 2025)	REACT (Yao et al., 2023b)	Reflexion (Shinn et al., 2023)	LATS (Zhou et al., 2024)
Tool Calling	✓	✓	✓	✓	✓	✓	✓	✓	✓
Planning	✓	✓	✓	✗	✓	✓	✓	✓	✓
Deliberate Tool Selection	✓	✗	✗	✗	✓	✓	✗	✗	✓
Tool Verification	✓	✗	✓	✗	✓	✓	✗	✓	✓
Tool Refinement	✓	✗	✓	✗	✓	✓	✗	✓	✗
Tool Pruning	✓	✗	✗	✗	✗	✗	✗	✗	✓
Skill Dimension									
Math	✓	✓	✓	✗	✓	✗	✓	✓	✓
External Knowledge	✓	✓	✓	✓	✓	✓	✓	✓	✓
Medical	✓	✓	✓	✗	✗	✗	✗	✗	✗
General Vision	✓	✓	✓	✓	✗	✗	✗	✗	✗
Document	✓	✓	✗	✓	✓	✗	✗	✗	✗

Table 1: A comparison of **ToolTree** with notable LLM agent frameworks, tool-augmented LLM systems and LLM agent tree search. Our method shows significant advantages in tool integration with diverse coverage of skills.

dependencies remains a key open problem.

Existing research on enhancing LLM agents with tool-use capabilities can be broadly categorized into three main streams. 1) General-purpose agent frameworks, such as GPT-Functions (OpenAI, 2024), Langchain (LangChain, 2024) and OctoTools (Lu et al., 2025), provide foundational capabilities by allowing LLMs to invoke external APIs or tools. However, these systems largely rely on greedy tool selection that choose a tool appearing locally optimal at each step without strategic planning. This lack of foresight often results in brittle performance when early decisions prove suboptimal or when later tool choices depend on earlier outcomes. 2) Tool augmented LLMs, including HuggingGPT (Shen et al., 2023), ToolChain* (Zhuang et al., 2024) and ToolPlanner (Liu et al., 2025), has made strides in more structured planning and deliberate tool selection. Nevertheless, these systems often do not explicitly integrate robust mechanisms for ongoing verification of intermediate tool outputs or iterative refinement of these outputs once found to be imprecise or incomplete, carrying forward unexamined results into subsequent steps. 3) LLM Agent Tree Search approaches, exemplified by Tree-of-Thought (Yao et al., 2023a), Reflexion (Shinn et al., 2023), and LATS (Zhou et al., 2024), have introduced search paradigms to explore sequences of actions or thoughts. However, the "tools" in these tree-search agents are usually limited to one or two generic primitives such as web retriever or calculator, limiting the evaluations to stay within text-centric domains. Scaling search to a multi-modal, multi-domain tool ecosystem, therefore, remains an open challenge, one that our framework tackles head-on.

To overcome these challenges, we introduce ToolTree, a general-purpose agent framework that integrates a plug-and-play Monte Carlo Tree Search (MCTS) module for deliberate tool selection. Unlike conventional greedy methods, our MCTS module explores multiple tool trajectories to identify effective multi-step execution plans. Central to this is a novel dual-stage LLM-guided evaluation: a pre-execution model predicts the utility of a tool before it is invoked, while a post-execution model assesses its actual contribution based on observed outcomes. This feedback loop enables the agent to refine its strategy iteratively, incorporating foresight and hindsight into tool selection. Additionally, we standardize tool interfaces through a domain-specific "tool library", allowing seamless orchestration across diverse tools and domains. Empirical evaluations across 15 downstream tasks demonstrate consistent improvements over strong baselines, with average gains exceeding 5%.

The main contributions of this paper can be summarized as follows:

- We propose ToolTree, a generalizable LLM agent framework featuring deliberate tool orchestration and standardized "tool library" that encapsulates domain-specific models for diverse tasks.
- ToolTree implement a novel plug-and-play MCTS-based tool selection module, which leverages both pre-evaluation and post-evaluation from the environment to guide strategic multi-step tool planning, moving beyond traditional greedy selection approaches.
- Extensive evaluation across 15 datasets demonstrate consistent downstream task improvements exceeding 5% on average compared to state-of-the-art agent frameworks.

2 Related Work

LLM-based Autonomous Agent Recent advancements in LLM-based autonomous agents have focused on augmenting LLMs with external modules to enhance planning (Hao et al., 2023; Liu et al., 2023), reasoning (Wei et al., 2022; Wang et al., 2022), memory (Zhong et al., 2024) and tool-use (Schick et al., 2023; Shen et al., 2023). These agents generally fall into two categories: General and Domain-Specific frameworks. General frameworks, like AutoGPT (AutoGPT, 2024), Langchain (LangChain, 2024), MetaGPT (Hong et al., 2023), while offering broad applicability with versatile, domain-agnostic foundational architecture, tend to lack depth for complex, multi-step tasks and specialize in tool pruning per task domain. Domain-specific frameworks for fields such as math (Poesia et al., 2024; Xiong et al., 2024), vision (Wu et al., 2023), research (OpenAI, 2025; Google, 2025), medical (Li et al., 2024; Tang et al., 2024). At the same time, achieving significant progress for a single field tends to over-optimized in one domain with narrow tool coverage without generalizability. Our framework distinguishes by incorporating a domain-specialized Tool-Card Library, enabling both wide-ranging multi-disciplinary task support and fine-grained tool specialization, alongside an explicit planner and executor that interacts with the environment for robust multi-step reasoning.

Tool Selection for LLM Agents Dynamic tool selection is crucial for complex tasks that require the use of sequential tools (Qu et al., 2025). In order to mitigate such a problem, prompt-based methods leverage LLMs with their strong world knowledge priors (Hao et al., 2023; Gu et al., 2024) as a planner to select tools using in-content learning techniques, such as chain-of-thought (Wei et al., 2022) or ReAct (Yao et al., 2023b) schema (Shen et al., 2023; Paranjape et al., 2023; Lu et al., 2025). Even though flexible, these approaches often make greedy, single-step choices without adequate looking-ahead or backtracking, potentially leading to hallucinated or incorrect actions (Qin et al., 2023; Liu et al., 2024). Alternatively, training-based methods fine-tune models or add specific heads for tool invocation (Schick et al., 2023; Yang et al., 2023), incurring significant computational and data annotation costs. In contrast, our method employs a training-free, tree-based module for hierarchical exploration and ranking of multi-step tool sequences, featuring both forward-looking evaluation and backwards verification.

Augmenting LLM Agent with Tree Search To address the limitations of reactive LLM agents in complex tasks requiring lookahead (Gu et al., 2024), augmenting them with tree search provides a deliberate planning layer. Various search algorithms, such as greedy search (Yao et al., 2023b), A* Search (Zhuang et al., 2024), Beam Search (Xie et al., 2023), MCTS (Zhou et al., 2024; Hao et al., 2023), BFS/DFS (Yao et al., 2023a), Best-first search (Koh et al., 2024) have been integrated at inference time. However, this method often lacks sufficient tool invocation diversity for broad domain generalization. Our approach addresses this with explicit tree search for tool selection, contrasting with LLM-internal reasoning prevalent in prior methods, and further incorporates dual environmental feedback for robust verification and plan refinement.

3 Methodology

In this paper, we introduce **ToolTree**, a training-free LLM-agent framework that performs deliberate multi-tool orchestration for diverse domain-specific tasks as visualized in Figure 2. Unlike previous frameworks that suffer from greedy or shallow planning of tool selection within a limited domain, ToolTree systematically addresses these issues through a four-component architecture: 1) a *Domain-Specialized Tool Library* for versatile access to a broad spectrum of external tools; 2) a lightweight *Planner* that sketches a coarse-grained route by filtering tools relevant to the user query; 3) *MCTS-based Tool Selector* that iteratively refines this route into a concrete, step-by-step tool implementation plan steered by dual feedback from LLM scores; 4) *Answer Generator* that assembles the optimal tool trajectory with their output into a final, fluent response.

3.1 Domain-Specialized Tool library

ToolTree interacts with a diverse set of external tools through its extensible Domain-Specialized Tool library, denoted as $\mathcal{T}_{\text{lib}}$. Each tool $t \in \mathcal{T}_{\text{lib}}$ is represented by a structured tool card C_t with explanatory metadata using JSON format to provide standardized information for further utilization. A card begins with a *"name"* plus a brief *"description"* so human operators and error logs remain readable. It also has a categorical *"domain tag"* drawn from vision, math, medical, external-

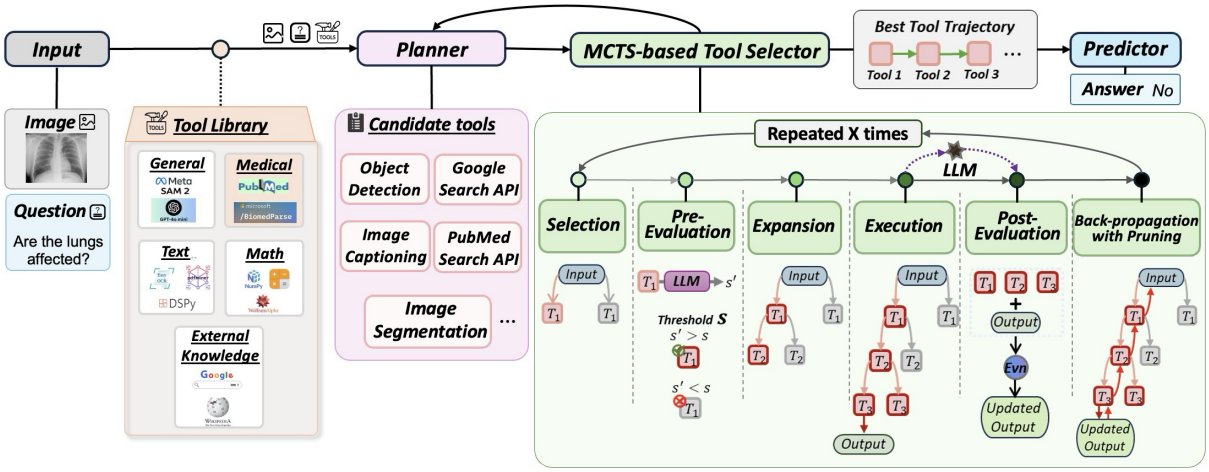


Figure 2: Architecture of **ToolTree**. An input query is processed sequentially by: (1) the *Domain-Specialized Tool Library*; (2) a lightweight *Planner* for coarse-grained tool filtering; (3) an *MCTS-based Tool Selector* that refines tool sequences via iterative, dual LLM-guided search, including selection, pre-evaluation, expansion, execution, post-evaluation, and backward-prorogation; and (4) an *Answer Generator/Predictor* to produce the final output.

knowledge, or text/document so that the planner discards whole groups that are clearly irrelevant to the user query. The card also carries a *"formal signature"*, listing every callable parameter together with admissible value ranges, and paired *"input schema"* and *"output schema"* so the LLM can reason about type compatibility when chaining tools. Finally, each card stores three short *"examples"* that serve as in-context demonstrations for the planner. The current library comprises thirty tool cards: five for vision, four for math, seven for external knowledge, six for medical reasoning, and eight for text-centric document tasks. Full details can be found in Appendix B.3.

This unified schema ensures that heterogeneous services ranging from a symbolic-algebra solver to a radiology image classifier can be ranked, compared, and invoked through exactly the same planning interface. Adding a new capability is therefore as simple as registering another card in $\mathcal{T}_{\text{lib}}$.

3.2 Planner

Given a user query q and the full tool library $\mathcal{T}_{\text{lib}}$, the *Planner* produces a coarse, low-cost sketch that seeds the Monte-Carlo Tree Search (MCTS) loop. We first utilize Planner as a classifier to assign one or more domain labels to query q. Any card whose *"domain tag"* is not among these labels is removed. For survivors, we verify that their declared *"input schema"* is satisfiable in the current context and that their *"output schema"* can be consumed by at least one other remaining tool. The check is implemented with a deterministic regex over the schema strings. For the surviving set, the *planner* runs a lightweight ranking pass that returns a scalar utility estimate $p_t \in [0, 1]$ for each tool. We then retain only those tools whose score meets or exceeds the threshold τ, giving the set of filtered candidates $\mathcal{T}^*_{\text{cand}} = \{t \mid p_t \geq \tau\}$. All tools in $\mathcal{T}^*_{\text{cand}}$ are passed into the MCTS loop, forming the immediate action space for its subsequent fine-grained exploration.

3.3 MCTS-based Tool Selector

The *Tool Selector* is the primary engine for intelligent decision-making within the ToolTree framework. Starting from the coarse candidate set $\mathcal{T}^*_{\text{cand}}$ and context C_0 produced by the *Planner*, it performs an MCTS to construct an explicit, ordered sequence of tool invocations that maximizes a learned utility signal. Unlike greedy schemes that decide one step at a time, our search explores multiple future branches, evaluates each prospective call with lightweight *pre-execution* scoring, and then refines these estimates with *post-execution* feedback once a tool is actually run. This look-ahead/look-back loop allows the agent to recover from early mischoices, avoid dead-end tool combinations, and allocate its limited call budget to the most promising trajectories.

Problem Definition. We model the search process as exploring a finite tree. A node at depth t is a pair $s_t = \langle C_t,\ A_{1:t} \rangle$, where C_t is the accumulated context that represents the user query plus all intermediate tool outputs and $A_{1:t} = (a_1, \ldots, a_t)$ is the sequence of tool calls executed so far. From s_t the legal actions are the tools in $\mathcal{T}^*_{\text{cand}}$ whose

"input schema" is compatible with C_t; we denote this set $\mathcal{A}(s_t)$. Invoking a tool $a \in \mathcal{A}(s_t)$ produces a successor state s_{t+1} and an immediate *post-execution reward* $r_{\text{post}}(s_t, a) \in [0, 1]$ scored by an LLM judge. The objective is to find a trajectory $A_{1:T}$ that maximizes the cumulative reward $R = \sum_{t=0}^{T-1} r_{\text{post}}(s_t, a_{t+1})$ within a rollout budget of at most $R_{\max}$.

Within the search tree for every node (s, a), we update two statistics after each rollout: the visit count $N(s, a)$, which records how many times action a has been selected from state s, and the mean value estimate $Q(s, a)$, which stores the running average of the post-execution rewards observed at that node. Both are initialized to 0 when the node is first created.

Selection. Given the current search state s, the selector repeatedly traverses the tree until it reaches a leaf by choosing the child action that follows a *prior-augmented UCT* score to balance the exploitation and exploration:

$$UCT(s, a) = Q(s, a) + \lambda r_{\text{pre}}(s, a) \sqrt{\frac{\ln N(s)}{N(s, a)}} \quad (1)$$

The first term $Q(s, a)$ in $\text{UCT}(s, a)$ drives exploitation as it accumulates the *post evaluation* rewards obtained so far. The second term modulates exploration with multiplier $r_{\text{pre}}(s, a)$ with *pre evaluation* score as a fast prediction obtained before any tool call is made, acting as a Bayesian before steering early roll-outs toward likely fruitful paths.

Pre-Evaluation. When a state, action pair (s, a) is visited for the first time, we query an LLM to predict the prospective utility of calling a next, returning $r_{\text{pre}}(s, a) \in [0, 1]$. If $r_{\text{pre}}(s, a) < \tau_{\text{pre}}$, the branch is discarded. Otherwise, the value is cached for use in Eq. (1).

Expansion. At a leaf state s_t, we first compute the set of actions that are still unused as $\mathcal{A}_{\text{rem}}(s_t) = \mathcal{A}(s_t) \setminus \{a_1, \ldots, a_t\}$. For every candidate $a \in \mathcal{A}_{\text{rem}}(s_t)$ whose *pre-evaluation* score reaches the threshold $r_{\text{pre}}(s_t, a) \geq \tau_{\text{pre}}$, we create a child node (s_t, a), caching the corresponding r_{pre} value.

Execution. The selected child action is executed against the real tool API, producing an output o_{t+1} that is appended to the context to form C_{t+1}. We cache output that is identical so that calls are not repeated inside a rollout budget.

Post-Evaluation. After the tool has been executed, we request a second LLM to assess the triple $\langle C_t, a, o_{t+1} \rangle$ and return a post-execution score $r_{\text{post}}(s_t, a) \in [0, 1]$ that reflects the utility of the new output. If $r_{\text{post}}(s_t, a) < \tau_{\text{post}}$, we discard the responding node. Otherwise, the node remains expandable in later roll-outs.

Backward Propagation. The obtained post-evaluation score $r_{\text{post}}(s_t, a)$ is propagated along the path from the new child back to the root. For every edge (s, a) on that path, we update $N(s, a) \leftarrow N(s, a) + 1$ and $Q(s, a) \leftarrow Q(s, a) + \frac{r_{\text{post}}(s_t, a) - Q(s, a)}{N(s, a)}$, thereby refining the exploitation term in Eq. (1) with the latest empirical evidence.

Tree Pruning. The two thresholds τ_{pre} and τ_{post} work together via pre-evaluation and post-evaluation to limit search overhead: the former filters out obviously irrelevant actions before any real call is made, while the latter cuts off branches that deliver disappointing results after execution.

Termination. Search stops when either (i) the allotted number of rollouts $R_{\max}$ is reached, or (ii) the best cumulative Q value has not improved by more than ϵ over the past k rollouts. Upon termination, the action sequence attached to the root child with the highest Q value is returned to the *Answer Generator* as the final ordered tool-execution plan.

3.4 Answer Generator/Predictor

Given the optimal trajectory $A_{1:T}$ returned by the tool selector, the *Answer Generator* converts the accumulated context C_T that contains the user query, intermediate reasoning, and the outputs of all executed tools into a natural-language form reply.

4 Experiment

Datasets. We evaluate our ToolTree framework using a diverse collection of 15 datasets spanning five distinct specialized domains: general, external knowledge, medical, math, and text/document. The general visual understanding domain includes main datasets such as VQAv2 (Goyal et al., 2017), GQA (Hudson and Manning, 2019), and ScienceQA (Lu et al., 2022). For the knowledge-based question answering domain, we utilize OK-VQA (Marino et al., 2019), A-OKVQA (Schwenk et al., 2022), and WebQ (Berant et al., 2013). The medical question answering domain is represented by MedQA (Jin et al., 2021), VQA-Rad (Lau et al., 2018), and PathVQA (He et al., 2020). In the mathematical reasoning domain, we include MATH (Hendrycks et al., 2021), Game of 24 (nlile, 2025), and Mathvista (Lu et al., 2024). Finally, text/document

Domain	Dataset	GPT4o-mini				GPT4o			
		Few-Shot	HuggingGPT	OctoTools	**ToolTree (Ours)**	Few-Shot	HuggingGPT	OctoTools	**ToolTree (Ours)**
General Visual	VQAv2	68.82	60.17	69.28	**74.47**	73.22	67.77	74.18	**76.43**
	GQA	63.80	65.13	66.14	**71.54**	66.84	60.33	68.58	**74.44**
	SQA	76.50	70.82	78.29	**84.28**	82.15	78.45	84.13	**87.33**
Medical	MedQA	79.14	84.33	86.18	**91.13**	83.20	86.73	92.17	**93.88**
	VQA-Rad	48.10	55.14	60.10	**63.27**	54.47	58.88	66.42	**74.12**
	PathVQA	24.90	40.72	43.13	**47.12**	26.20	37.82	46.17	**50.86**
External Knowledge	OK-VQA	48.46	44.19	50.17	**55.38**	53.62	50.12	53.42	**59.27**
	A-OKVQA	60.28	55.81	62.15	**70.54**	65.91	60.33	68.33	**73.48**
	WebQ	50.20	56.24	61.12	**64.28**	56.41	58.18	63.44	**67.94**
Math	MATH	53.26	45.14	58.43	**69.42**	61.45	53.51	68.57	**78.19**
	Game-24	26.50	22.66	34.18	**43.33**	33.15	25.43	40.18	**47.85**
	MathVista	52.53	55.62	57.97	**63.14**	59.10	58.44	61.70	**65.58**
Text / Doc.	TextVQA	72.42	68.24	74.69	**82.26**	76.28	70.14	77.17	**85.43**
	Doc-VQA	83.28	83.10	84.23	**89.43**	87.11	82.13	89.39	**92.33**
	HotpotQA	37.29	46.48	48.11	**54.15**	43.77	51.82	53.14	**56.33**
	Average	56.37	56.92	62.94	**68.65**	61.53	60.01	67.80	**72.70**

Table 2: Comparison across 15 datasets in five domains. ToolTree consistently outperforms standard few-shot prompting, HuggingGPT, and OctoTools on both GPT-4o-mini and GPT-4o, achieving highest overall score.

recognition and domain understanding comprises TextVQA (Singh et al., 2019), Doc-VQA (Mathew et al., 2021), and HotpotQA (Yang et al., 2018). Details of each dataset are shown in Appendix B.1.

Baselines and Metric. We compare our proposed framework, ToolTree, against several established baselines to demonstrate its efficacy: (1) standard few-shot inference without specialized frameworks; (2) OctoTools (Lu et al., 2025), a representative LLM agent framework employing an iterative planner-executor-verifier paradigm for structured tool use; and (3) HuggingGPT (Shen et al., 2023), a pioneering tool-augmented LLM system that coordinates tasks across Hugging Face models under an LLM controller. To specifically assess the tool selection and reasoning capabilities of ToolTree as a plug-and-play module against other decision-making strategies, we further compare it with Chain-of-Thought with self-consistency (CoT-SC) (Wang et al., 2022), Tree of Thoughts (ToT) (Yao et al., 2023a), and ReAct (Yao et al., 2023b).

To ensure a fair comparison across all approaches, experiments are conducted using either GPT-4o-mini or GPT-4o as the backbone model, with all frameworks provided an identical set of tools and the same number of examples for prompting. Performance is reported using the primary evaluation metric native to each dataset: accuracy, exact match (EM), F1 score, or success rate.

Hyperparameter Setting. All ToolTree components, including *Planner*, *Tool Selector*, pre-evaluation LLM, and *Answer Generator*, run the same backbone LLM with either GPT-4o-mini GPT-4o, while post-evaluation scores are produced by Gemini 2.0. More details on the influence of different post evaluation LLMs can be found in Appendix 8. The *Planner* retains tools whose relevance exceeds $\tau = 0.3$. Evoked by (Zhou et al., 2024), during MCTS we set the exploration constant to $\lambda = 1.4$, allow at most $R_{\max} = 60$ roll-outs, and prune branches whenever $r_{\text{pre}} < 0.3$ or $r_{\text{post}} < 0.4$; search stops early if the best Q value increases by $< 10^{-3}$ over 10 consecutive roll-outs.

5 Result

5.1 Main Results

We evaluate ToolTree against three distinct multi-tool orchestration baselines: Few-Shot prompting, HuggingGPT, and OctoTools with two backbone models, GPT4o-mini and GPT4o. As covered in Table 2, our evaluation spans 15 datasets in five diverse domains, including general visual, medical, external knowledge, math, and text/document.

Our framework consistently achieves superior performance across five domains. Under GPT4o-mini, it attains an average of 68.65%, outperforming Few-Shot and HuggingGPT by over 11.7 points and OctoTools by 5.71 points on average. A similar trend is observed with the more capable GPT4o backbone, where ToolTree outperforms Few-Shot and HuggingGPT by more than 11.1 points and OctoTools by 4.9 points on average. Notably, ToolTree demonstrates substantial gains on traditionally challenging, domain-specific datasets such

Configuration	VQA-Rad	OK-VQA	MathVista	SQA	HotpotQA	AVG
Langchain	62.18	49.18	54.24	76.59	39.82	56.40
w/o COT-SC	65.14	54.37	56.88	82.17	44.90	60.69
w/o REACT	66.28	52.33	59.25	80.95	45.18	60.80
w/o ToT	63.04	48.12	65.33	78.33	**52.28**	61.42
w/o ToolTree	**67.72**	**54.27**	**65.74**	**81.33**	51.94	**64.20**
MetaGPT	64.13	53.84	54.88	78.16	37.72	57.75
w/o COT-SC	68.74	54.88	60.30	79.44	46.90	62.05
w/o REACT	66.32	55.11	58.94	80.54	49.56	62.09
w/o ToT	65.42	50.52	60.14	80.47	**56.21**	62.55
w/o ToolTree	**69.24**	**55.83**	**62.28**	**82.57**	54.77	**64.94**

Table 3: Comparison of ToolTree as a plug-and-play module with REACT, COT-SC and ToT modules. ToolTree achieves highest score on average.

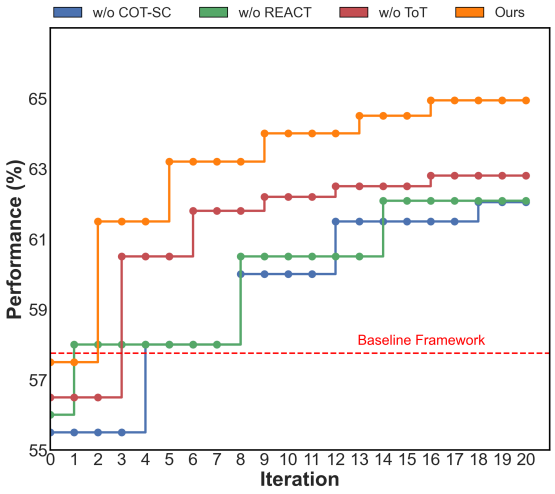


Figure 3: Relationship of performance and number of iterations. ToolTree consistently beats other modules.

as PathVQA and Game-of-24, with 22.22% and 16.83% performance gain compared with few-shot baselines under GPT4o-mini. These significant improvements underscore the superiority of our framework that integrates a domain specialized tool library and MCTS-based tool selector. Domain-grouped break down results can be found in Appendix A.1. Comparison with domain-specific agent frameworks can be found in Appendix A.3.

5.2 Plug-and-Play Module Comparision

We evaluated our plug-and-play modules ToolTree-Module on one representative dataset from each of the five domains under two off-the-shelf LLM-agent frameworks, Langchain and MetaGPT. For each framework, we start from the vanilla agent with no extra tool use module and then insert exactly one of four modules—Chain-of-Thought Self-Consistency (COT-SC), REACT, Tree-of-Thought (ToT), or our proposed ToolTree-Module—while holding all other settings like prompt format, tool APIs, number of iterations/trajectories, and random seeds identical.

As Table 3 shows, our ToolTree-Module consistently yields the highest accuracy on overall average and four of the five benchmarks across both frameworks, outperforming COT-SC, REACT, and ToT by 3–8 points on each dataset and 7 points on average against the unaugmented agent. The only exception is HotpotQA, where tree-of-thought's structured reasoning over LLM's hidden state excels at systematically decomposing the multi-hop problem and exploring diverse evidence-linking pathways crucial for this dataset. Nevertheless, this internal state search nature also makes it far worse than our module in domain-specialized tasks that require external tools such as vision, medical and knowledge, where our module's versatile integration and adaptive orchestration of these tools yields significantly better performance.

5.3 Number of Iteration Influence

To understand the relationship between computational budget and performance for different plug-and-play modules, we analyze their accuracy as a function of "iterations" as illustrated in Figure 3. Here, an iteration denotes one complete reasoning pass: for CoT-SC, it is one self-consistent sample, for ReAct, it is one chain-of-thought + tool-call cycle, for ToT, it is one branch expansion, and for our method, it is one MCTS rollout.

We observe a general trend where performance gains diminish as iterations increase, with all modules potentially introducing noise and underperforming the baseline framework at very low iteration counts (e.g., 0-2 iterations) before their structured reasoning takes effect. Notably, ToolTree-Module and ToT exhibit a more pronounced stepwise improvement curve, indicating their deeper search or structured exploration mechanisms progressively uncover better solutions with increased iterations. In contrast, COT-SC and REACT show flatter trajectories, suggesting they reach their performance plateaus more quickly with fewer iterations, possibly due to their less extensive exploration of the solution space. Crucially, across the board, our approach achieves the highest accuracy at every budget, confirming that dual evaluation and value-based pruning make each additional rollout more informative.

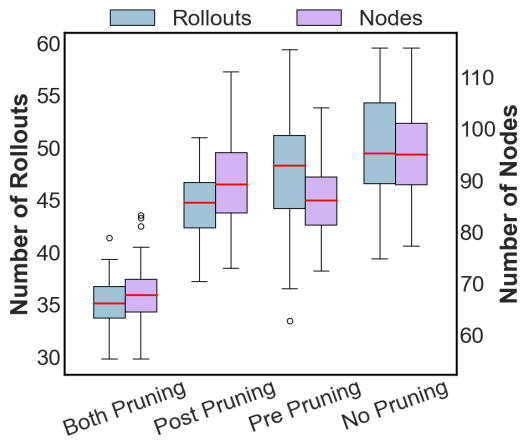


Figure 4: Impact of Pruning Strategies on Search Efficiency. Combined pruning yields the fewest median rollouts and nodes with the tightest variance.

5.4 Influence of Dual Evaluation on Efficiency

To evaluate the effectiveness of our dual-evaluation pruning strategy on search efficiency, we conducted experiments on 100 sampled examples from each dataset, setting a maximum rollout budget of 60. We compared four configurations: 1) "No Pruning", where LLM evaluation scores solely inform the MCTS value function without active discarding of trajectories; 2) "Pre Pruning", where pre-evaluation scores are used for pruning; 3) "Post Pruning," where post-evaluation scores are used to prune trajectories after tree expansion; and 4) "Both Pruning," combining both mechanisms as mentioned in Section 3.3.

As shown in Figure 4, pre-evaluation pruning substantially reduces the median number of nodes expanded to approximately 78 from 98 by directly curtailing unpromising branch explorations for a narrower search tree while it has less influence on the number of rollouts as it basically serves as a prior for reward function without dominance. Conversely, post-evaluation pruning more substantially reduces median rollouts to approximately 45 from 50, as its accurate rewards provide clearer solution quality signals for potentially earlier confident convergence. Crucially, employing "Both Pruning" mechanisms yields the most significant efficiency gains, achieving the lowest median nodes and rollouts, alongside more stable performance indicated by reduced variance. These findings demonstrate that the dual-evaluation pruning strategy effectively enhances search efficiency by reducing unnecessary exploration and converging on solutions rapidly.

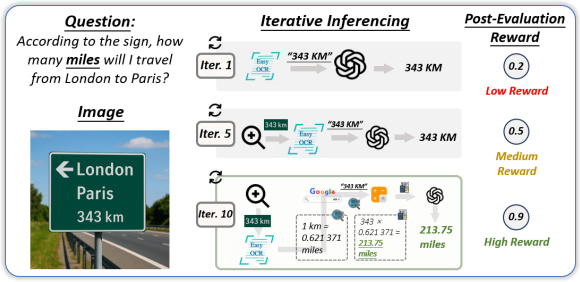


Figure 5: A Sample Case of ToolTree on TextVQA.

6 Qualitative Analysis

Figure 5 showcases how ToolTree progressively corrects itself on a TextVQA task. With the number of rollouts grows in the MCTS loop, ToolTree finds better tool trajectories guided by both the pre-evaluation score as the prior and the post-evaluation score as the dominant reward. The query asks, "According to the sign, how many *miles* is it from London to Paris?"; the photo shows "343 km." In its first rollout, the agent invokes a lightweight OCR tool, passes the raw text to the LLM, and naively returns "343 km," earning a low post-evaluation score (0.2). By the fifth rollout, the search has inserted the *patch-zoom* tool to crop the numeric region and rerun OCR, but it still reports kilometers and receives only a medium reward (0.5). Guided by these signals, the tenth rollout adds a unit-conversion API after OCR; the calculator multiplies $343 \times 0.621\,371$, and the LLM outputs the correct "213.75 miles," which the judge scores 0.9. The example illustrates how dual LLM feedback steers the MCTS toward richer tool chains. More case study can be found in Appendix A.5.

7 Conclusion

This paper presents ToolTree, a training-free agent framework that integrates a plug-and-play MCTS-based tool selection module and domain-specialized tool library to enable robust multi-tool orchestration across diverse tasks. ToolTree explores a dual feedback mechanism from the environment to provide nuanced guidance for MCTS, enabling both efficient search via strategic pruning and effective discovery of optimal tool trajectory. Experiments over the 15 datasets across diverse domains demonstrate ToolTree consistently outperforms state-of-the-art agent systems by 5 percent on average success rate. We hope this framework will serve as a valuable foundation for future explorations into sophisticated tool orchestration and deliberate reasoning in more advanced AI agents.

Limitation

While ToolTree excels across a diverse range of tasks, its dependence on a powerful LLM endowed with extensive world knowledge and reasoning priors may limit applicability to smaller language models. Moreover, despite the benefits of pruning, the MCTS-driven search is still hard to to match the efficiency of greedy heuristics. To address these limitations, we intend to explore methods for enhancing the applicability of ToolTree with smaller language models, such as by developing more lightweight reasoning components or employing knowledge distillation techniques to transfer essential capabilities. We also intend to develop a better strategy in the future that enables dynamically switching from deliberate reasoning to heuristic reasoning.

References

AutoGPT. 2024. AutoGPT. https://github.com/Significant-Gravitas/AutoGPT. GitHub repository.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544. ACL.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Google. 2025. Gemini Deep Research. https://gemini.google/overview/deep-research/. Web page.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, et al. 2024. Is your llm secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. 2020. Pathvqa: 30000+ questions for medical visual question answering. *arXiv preprint arXiv:2003.10286*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 3(4):6.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*.

LangChain. 2024. LangChain. https://github.com/langchain-ai/langchain. GitHub repository.

Jason J. Lau, Soumya Gayen, Asma Ben Abacha, and Dina Demner-Fushman. 2018. A dataset of clinically generated visual questions and answers about radiology images. *Scientific Data*, 5(1):1–10. VQA-RAD dataset.

Binxu Li, Tiankai Yan, Yuanting Pan, Jie Luo, Ruiyang Ji, Jiayuan Ding, Zhe Xu, Shilong Liu, Haoyu Dong, Zihao Lin, and Yixin Wang. 2024. MMedAgent: Learning to use medical tools with multi-modal agent. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8745–8760, Miami, Florida, USA. Association for Computational Linguistics.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Xukun Liu, Zhiyuan Peng, Xiaoyuan Yi, Xing Xie, Lirong Xiang, Yuchen Liu, and Dongkuan Xu. 2024. Toolnet: Connecting large language models with massive tools via tool graph. *arXiv preprint arXiv:2403.00839*.

Yanming Liu, Xinyue Peng, Jiannan Cao, Shi Bo, Yuwei Zhang, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. 2025. Tool-planner: Task planning with clusters across multiple tools. In *The Thirteenth International Conference on Learning Representations*.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*.

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

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.

nlile. 2025. Game of 24 dataset. Hugging Face repository, https://huggingface.co/datasets/nlile/24-game. [Dataset].

OpenAI. 2024. Function Calling. https://platform.openai.com/docs/guides/function-calling. Online documentation.

OpenAI. 2025. Introducing Deep Research. https://openai.com/index/introducing-deep-research/. Blog post.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.

Gabriel Poesia, David Broman, Nick Haber, and Noah Goodman. 2024. Learning formal mathematics from intrinsic motivation. *Advances in Neural Information Processing Systems*, 37:43032–43057.

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.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. 2022. A-okvqa: A benchmark for visual question answering using world knowledge. In *European conference on computer vision*, pages 146–162. Springer.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180.

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.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2024. MedAgents: Large language models as collaborators for zero-shot medical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 599–621, Bangkok, Thailand. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.

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

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Self-evaluation guided beam search for reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, et al. 2024. Building math agents with multi-turn iterative preference learning. *arXiv preprint arXiv:2409.02392*.

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

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36:71995–72007.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2369–2380.

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

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 62138–62160. PMLR.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A. Rossi, Somdeb Sarkhel, and Chao Zhang. 2024. Toolchain*: Efficient action space navigation in large language models with a* search. In *The Twelfth International Conference on Learning Representations*.

A Additional Experiment Results

A.1 Performance Comparison for Each Domain

Figure 6 presents a detailed breakdown comparison of ToolTree's average performance across five specialized domains under two backbone models. ToolTree consistently achieves the highest performance across all domains, particularly excelling in the Math and External Knowledge domains. For example, in the Math domain under GPT4o-mini, ToolTree reaches 63.4%, significantly outperforming Few-Shot by 19.3%, HuggingGPT by 22.2%, and OctoTools by 11.2%. Similarly notable gains are observed under GPT4o.

In the Medical domain, ToolTree surpasses HuggingGPT by 12.0% and OctoTools by 5.0% using GPT4o-mini, demonstrating its strength in tasks requiring specialized external knowledge and precise tool interactions. In the General Visual and Text/Document domains, ToolTree continues to show consistent improvements of roughly 5 – 7% over baselines for both backbone models. These results underscore the robustness of ToolTree's MCTS-based tool selection and dual evaluation across diverse reasoning challenges.

A.2 Performance Comparison with Baseline

We measured ToolTree's per-dataset improvement over a GPT4o few-shot baseline by subtracting the baseline accuracy from ToolTree's accuracy on each of the fifteen tasks and plotting the results in Figure 7. The chart shows gains on every benchmark: PathVQA sits at the top with an uplift exceeding twenty points, followed by the Game of 24 and HotpotQA climbing into the mid-teens, and VQA-Rad and A-OKVQA rising by around 15% and 10% respectively. Even general visual tasks like VQAv2 and TextVQA register solid improvements of roughly six to eight points. This pattern reflects ToolTree's strength in orchestrating multi-step, domain-specialized tool chains that is essential for medical and mathematical puzzles, while its verification and pruning mechanisms consistently enhance performance on more conventional downstream tasks.

A.3 Comparison With Domain-Specific Framework

We conducted experiments comparing ToolTree against several domain-specialized agent frameworks, including MMedAgent, LATS, and VIPERGPT, across five representative benchmarks from medical (VQA-Rad), external knowledge (OK-VQA), mathematics (MathVista), science reasoning (ScienceQA), and multi-hop reasoning (HotpotQA) domains. Each domain-specialized baseline is designed specifically for optimal performance in its own niche area.

Model	VQA-Rad	OK-VQA	MathVista	ScienceQA	HotpotQA	Average
MMedAgent	84.32	31.25	21.15	57.24	38.20	46.43
LATS	20.48	27.26	59.33	58.17	**77.54**	48.56
VIPERGPT	58.24	52.44	64.28	**88.64**	48.22	62.36
ToolTree (Ours)	74.12	59.27	**65.58**	87.33	56.33	**68.53**

Table 4: Performance (%) comparison between domain-specialized agent baselines and our ToolTree framework across five diverse benchmarks. Tooltree generalize well on different domain-specific tasks.

As illustrated in Table 4, our ToolTree consistently achieves the highest average accuracy of 68.53%. Specifically, ToolTree significantly outperforms MMedAgent in external knowledge, mathematics, science reasoning, and multi-hop reasoning tasks. Compared to LATS, which excels specifically in multi-hop reasoning, our framework substantially surpasses it by over 53.64% in medical image analysis (VQA-Rad), and around 32.01% in external knowledge (OK-VQA). ToolTree also achieves competitive performance compared with VIPERGPT, consistently outperforming it in medical tasks and external knowledge tasks. This improvement pattern indicates that domain-specific LLM agents suffer from poor cross-domain generalization, resulting in reduced overall accuracy across diverse tasks. Moreover, specialized LLM agent rely heavily on large-scale domain-specific datasets, many of which are not publicly available, limiting their reproducibility and adaptability

A.4 Effect of Dual Feedback on Accuracy

We measured how the choice of post-evaluation model affects overall and domain-specific accuracy by running the MCTS pipeline under four settings: no post-evaluation, GPT4o-mini as judge, GPT4o as judge, and Gemini 2.0 as judge as Figure 8. Across all tasks, accuracy steadily increases with more powerful judges, rising from 54.2% to 60.8% (Gemini 2.0). The largest improvements appear in vision and text/document tasks, where nuanced output verification matters most. These results show that richer post-execution feedback enables the agent to better discriminate useful tool calls, leading to more accurate final answers.

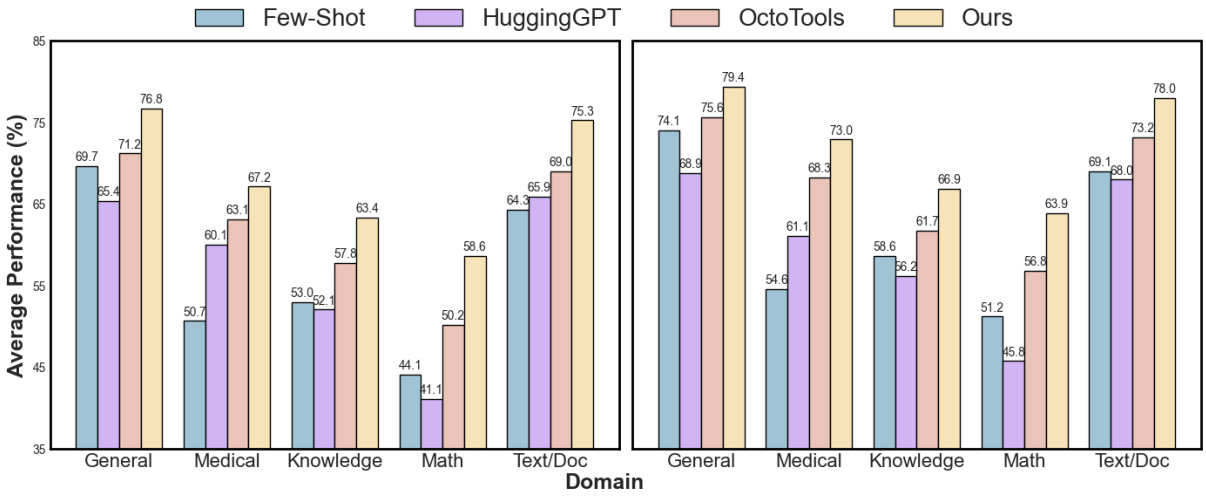


Figure 6: Break down comparison for each of the domain

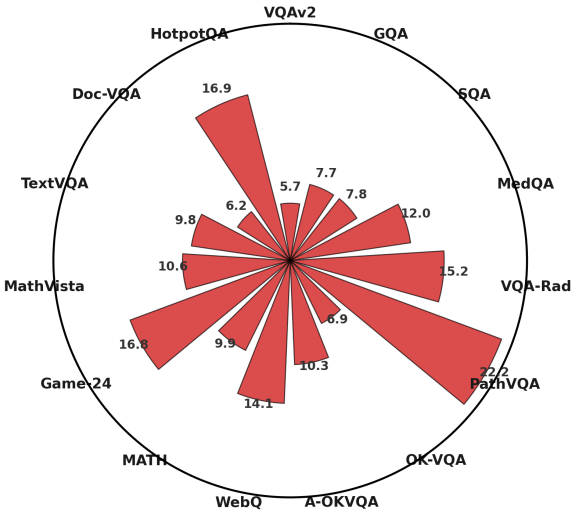


Figure 7: Break down comparison with the few-shot baseline setup under GPT4o-mini.

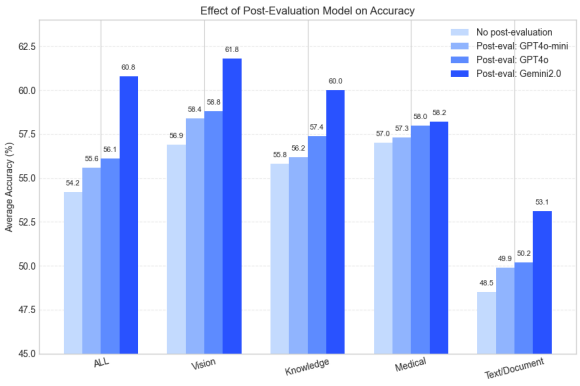
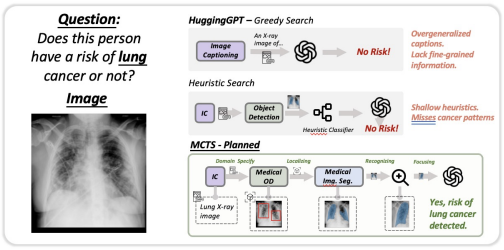
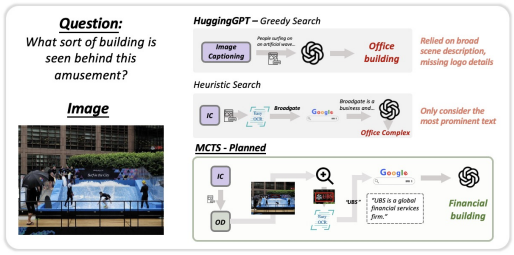


Figure 8: Effect of different LLM for post evaluation.



(a) Example inference trajectory for a medical VQA query.



(b) Example inference trajectory for a multi-hop reasoning query.

Figure 9: Two qualitative case studies showcasing ToolTree's iterative tool orchestration on (a) a radiology image question and (b) a multi-hop knowledge reasoning task.

A.5 Additional Case Study

B Experiment Details

B.1 Benchmark Dataset

B.1.1 General Visual Understanding

- **VQAv2 (Goyal et al., 2017):** A large-scale dataset for visual question answering based on COCO images with open-ended questions. It requires diverse visual understanding capa-

bilities like object recognition and counting. VQAv2 serves as a standard benchmark with a balanced question distribution.

- **GQA (Hudson and Manning, 2019):** Focuses on compositional reasoning and spatial understanding using scene graphs derived from Visual Genome images. Questions are generated with controlled reasoning structures and complexity. It also offers a balanced version to mitigate language priors.

- **ScienceQA (Lu et al., 2022):** A multimodal dataset featuring science questions from educational curricula, often including text and an image (diagrams, experiments). It requires scientific reasoning and involves chain-of-thought reasoning that provides explanations for answers. It Covers diverse topics like physics, chemistry, and biology.

B.1.2 Knowledge-Based Question Answering

- **OK-VQA (Marino et al., 2019):** A visual question answering dataset where questions necessitate external knowledge not present in the COCO images. It challenges models to connect visual concepts with world knowledge.

- **A-OKVQA (Schwenk et al., 2022):** Extends knowledge-based VQA by requiring verifiable and explainable reasoning for answers. It uses complex questions answerable via direct input or multiple choice and focuses on the reasoning process linking vision and knowledge.

- **WebQ (Berant et al., 2013):** A text-based question answering dataset using questions derived from web searches, answerable using facts from the Freebase knowledge graph. It tests the ability to map natural language questions to structured KB queries, containing primarily factoid questions.

B.1.3 Medical Question Answering

- **MedQA (Jin et al., 2021):** A text-based dataset containing multiple-choice questions from professional medical board exams. It requires specialized medical domain knowledge and clinical reasoning that covers a wide range of biomedical subjects.

- **VQA-Rad (Lau et al., 2018):** Focuses on visual question answering specifically for radiology images (X-rays, CT scans). Questions are posed by clinicians regarding image findings, anatomy, or potential diagnoses.

- **PathVQA (He et al., 2020):** A VQA dataset centered on pathology images (microscopic tissue slides). Questions require identifying fine-grained visual details relevant to disease diagnosis and analysis. Its questions often relates to cancer detection and grading.

B.1.4 Mathematical Reasoning

- **MATH (Hendrycks et al., 2021):** A dataset comprised of challenging mathematics problems from competitions (AMC, AIME), presented in text format. It requires complex, multi-step symbolic reasoning across various math subjects and provides step-by-step solutions.

- **Game of 24 (nlile, 2025):** A mathematical reasoning task requiring the use of four given numbers exactly once with arithmetic operations (+, -, *, /) to reach 24. It tests numerical reasoning, planning, and symbolic manipulation capabilities.

- **Mathvista (Lu et al., 2024):** A benchmark for evaluating mathematical reasoning within visual contexts like plots, charts, and diagrams. It requires integrating visual perception (e.g., reading values from axes) with mathematical problem-solving.

B.1.5 Text/Document Recognition and Understanding

- **TextVQA (Singh et al., 2019):** A VQA dataset where answering questions requires reading and understanding text depicted within real-world images (e.g., signs, labels). It necessitates integrating OCR capabilities with visual and language reasoning.

- **DocVQA (Mathew et al., 2021):** Focuses on visual question answering applied to images of documents with complex layouts. It requires understanding document structure, extracting text (OCR), and potentially synthesizing information from different regions. It is sourced from scanned or digital document pages.

- **HotpotQA (Yang et al., 2018):** A text-based question answering dataset specifically designed for multi-hop reasoning. Answering

questions requires finding and integrating information scattered across multiple source text passages (from Wikipedia).

B.2 Baselines

Baselines. We compare ToolTree against the following methods, using each dataset's primary evaluation metric (accuracy, exact match, F1 or success rate):

- **Few-Shot Inference.** Direct prompting of the backbone LLM (GPT-4o-mini or GPT-4o) with the same in-context examples but *without* any external tools or specialized framework.
- **OctoTools** (Lu et al., 2025). An LLM-agent framework that uses an iterative planner–executor–verifier loop over "tool cards" for structured API invocation, but relies on greedy selection and single-stage verification.
- **HuggingGPT** (Shen et al., 2023). A pioneering tool-augmented system where an LLM plans and dispatches sub-tasks to Hugging Face models; it lacks dynamic search or multi-step pruning.
- **Chain-of-Thought with Self-Consistency (CoT-SC)** (Wang et al., 2022). A prompting technique that samples multiple chain-of-thought traces and aggregates answers by majority vote, but does not incorporate external tool calls.
- **Tree of Thoughts (ToT)** (Yao et al., 2023a). A search-based reasoning module that branches on intermediate reasoning tokens in the prompt, offering look-ahead over textual hypotheses but limited to internal LLM operations.
- **ReAct** (Yao et al., 2023b). A tool-use schema that interleaves chain-of-thought with action calls in a fixed loop, providing basic tool invocation but no strategic backtracking or pruning.

B.3 Tool Library

Table 6 summarizes the external tools and models integrated within the ToolTree library, categorized by their domain specialization. The library offers broad coverage across general visual understanding, knowledge-based VQA, medical QA, mathematical reasoning, and text/document tasks. For each domain, a diverse set of functions—ranging from object detection and image segmentation to knowledge graph querying, medical report generation, and OCR—are supported by state-of-the-art models and APIs. This comprehensive and modular toolset enables ToolTree to handle a wide spectrum of complex, multi-modal tasks with domain-adaptive precision.

B.4 Tool card Metadata Example

We hereby attach the metadata for medical object detection tool as an illustrative example in Table 5.

Field	Type	Description / Example
`tool_name`	string	`"Medical_Object_Detection"`
`description`	string	A tool that detects the organs within a given medical image such as CT, MRI, X-Ray and pathology images.
`input`	`image: str`	Path to the image file (e.g. `"lung_cancer_Image.png"`)
	`prompt: str`	Prompt to guide detection (default: "Detect the organs in the given image.")
`output`	dict	Detected organs with their bounding box, organ name, and confidence score.
`example`	`input`	`{"lung_cancer_Image.png"}`
	`output`	`{"object_1": {"name":"left lung", "bounding box":[27,45,31,102], "confidence":0.82}, "object_2": {"name":"right lung", "bounding box":[57,48,35,98], "confidence":0.82}}`

Table 5: Metadata schema for the `Medical_Object_Detection` tool.

Domain	Tool	Function / Model
General Visual	Object Detection	GroundingDINO v2
	Image Segmentation	Segment Anything Model (SAM)
	Image Captioning	GPT-4o-mini
	Image Tagging	RAM (Recognize Anything Model)
	Patch Zooming	Vanilla Patch Zoomer (4×)
Knowledge-based VQA	Search Engine	Google Search API
	Knowledge Graph	Wikidata SPARQL
	Object Detection	GroundingDINO v2
	Image Segmentation	SAM
	Image Captioning	GPT-4o-mini
	Image Tagging	RAM
	Patch Zooming	Vanilla Patch Zoomer (4×)
Medical QA	Image Retrieval	PubMed Search API
	Object Detection	BioMedParse
	Image Segmentation	BioMedParse
	Image Classification	BioMedCLIP
	Report Generation	ChatCAD
	Retrieval-Augmented	ChatCAD+ (RAG)
Math Reasoning	Calculator	Arithmetic API
	Code Interpreter	Python Code Interpreter
	Math Solver	Wolfram Alpha
	Image Captioning	GPT-4o-mini (when visual input)
Text/Document	OCR	EasyOCR
	Layout Parsing	PDFMiner
	Knowledge Graph	Wikidata SPARQL
	Object Detection	GroundingDINO v2
	Image Segmentation	SAM
	Image Captioning	GPT-4o-mini
	Image Tagging	RAM
	Patch Zooming	Vanilla Patch Zoomer (4×)

Table 6: Summary of external tools and models in the ToolTree library, organized by domain specialization.