Think Hierarchically, Act Optimally: Decoupled Hierarchical Planning and Execution for LLM Agents

Vikas Kumar[1], **Jyotsana Khatri**[1], **Shirish Karande**[1]
[1]TCS Research
kumar.vikas17@tcs.com, khatri.jyotsana@tcs.com, shirish.karande@tcs.com

Abstract

Large Language Models (LLMs) have demonstrated strong reasoning and problem-solving capabilities, making them promising candidates for autonomous agents. However, they struggle with long-horizon task planning due to error propagation, limited exploration, and inefficient handling of complex action spaces. Prior approaches either perform flat search over primitive actions or perform hierarchical subgoal decomposition with greedy execution, limiting their ability to explore diverse strategies. In this work, we propose a decoupled hierarchical planning framework that separates high-level planning from low-level execution. At the planning level, the agent generates structured subgoals with explicit control flow. At the execution level, we introduce targeted exploration using Monte Carlo Tree Search (MCTS) within each subgoal's action space, allowing the agent to evaluate alternative action sequences before committing to decisions. This two-level formulation enables efficient and focused exploration while mitigating the reliance on long context histories. Our approach combines the strengths of structured reasoning and search-based optimization, improving adaptability in long-horizon settings. Experimental results on task planning benchmarks demonstrate that our method outperforms existing LLM-based planning frameworks, achieving higher success rates.

Introduction

Recent advancements in Large Language Models (LLMs) have showcased remarkable intelligence across a wide range of domains, including reasoning (Wei et al. 2022; Yao et al. 2022; Kojima et al. 2022), self-verification (Wang et al. 2022; Miao, Teh, and Rainforth 2023; Madaan et al. 2023), tool usage (Schick et al. 2023; Tang et al. 2023), task planning (Yao et al. 2023b; Shinn et al. 2023), and instruction-following. Amongst these proficiencies, task planning has emerged as a particularly significant aspect, positioning LLMs as a foundational component for intelligent decision-making systems. This has opened new avenues in AI-driven automation, where LLMs can not only generate structured plans but also dynamically adapt to evolving tasks, making them a promising cognitive core for next-generation autonomous agents. There have been significant attempts in using LLMs as task planning agents as they are equipped with extensive amount of knowledge and significant strength in commonsense reasoning. Several studies have used LLM agents to solve the standard robotic environment tasks like BlockWorld(Vavilala et al. 2023), Alfworld(Shridhar et al. 2020b), Shopping environments (Yao et al. 2022; Zhou et al. 2023), and Travel planning (Xie et al. 2024).

The robust solution for designing to solve complex tasks like web shopping journey, robot navigation requires a step by step action plan with concrete reasoning/justification for each action. Initial investigations like (Wei et al. 2022; Yao et al. 2023b; Shinn et al. 2023), explores the agent's internal reasoning abilites and decision making for solving the task. While these techniques works well with LLMs with parameters size $\sim$ 1.8 trillion parameters like gpt-4 (Baktash and Dawodi 2023), but the small models with $\sim$ 7–8B parameters like Llama3.1-7B (Grattafiori et al. 2024) often struggles in solving these tasks due to hallucination and poor remembering long contexts capabilities. The recent literature has shown that improving prompting methods alone won't improve the planning capabilities of the LLM agent and hence require an augmented module to support or overcome certain limitations or challenges faced by the LLMs. Some papers uses an external memory module to help LLM remember the long contexts, while some use external heuristic module to score the action chosen by the agent. Other works uses a separate LLM acting as a critic which can judge the actor agent and guiding it's actions(Jang, Lee, and Kim 2022; Springenberg et al. 2024).

Although, the recent literature have made significant progress towards enhancing task planning abilities using LLM agents specially for $7 - 8B$ parameter models. There are several existing challenges which requires a robust planning agent framework. Our main contribution lies in proposing a decoupled hierarchical planning framework where we separate high-level subgoal generation from low-level action execution, enabling structured reasoning while reducing error propagation in long-horizon tasks. We apply MCTS selectively within a subgoal's action space, allowing efficient evaluation of alternative actions without global search overhead.

Related Work

Language Models for Task Planning

Preliminary methods explores multiple prompting strategies ranging from prompting agent to reason and act in a loop (Yao et al. 2023b), or generating a chain of sequence of decisions along with the reasoning for each step (Wei et al. 2022; Wang et al. 2022) in order to enhance agent's reasoning and decision making abilities.

Textual Feedback or Improving Context

While prior work leverage LLMs for planning but they do not incorporate the agent's past experiences to refine action decisions. Incorporating techniques like reflections and refinement enables agent's to correct their errors. (Madaan et al. 2023) introduces a cyclic framework involving generation, evaluation, and refinement. Recent studies (Shinn et al. 2023; Zhang et al. 2023; Wang and Li 2023; Zhao et al. 2024; Wang et al. 2022; Miao, Teh, and Rainforth 2023) explore mechanisms for integrating an agent's previous experiences into the input context, enabling iterative learning. By allowing agents to reflect on past mistakes and adjust their strategies accordingly, these approaches enhance adaptability and improve long-term planning effectiveness. Shinn et al. (2023) builds upon ReAct by integrating an evaluation module to analyze task trajectories. Upon detecting errors, the LLM generates self-reflective insights, enabling systematic corrections and improved decision-making termed as verbal reinforcement learning.

Augmenting LLMs with External Modules and RL

Incorporating past experience into an agent's input context can improve planning performance; however, such approaches are constrained by the limited context length of LLMs, making it impractical to retain full interaction histories. Reinforcement learning (RL), which has long been used for sequential decision-making, has therefore been explored as a complementary mechanism for extending LLM-based agents. Several methods train LLMs directly as policy models using RL-based objectives (Jang, Lee, and Kim 2022; Yang et al. 2024). For instance, Jang, Lee, and Kim (2022) fine-tunes GPT-2 via behavior cloning from critic-guided, self-generated feedback, preserving fluency while improving decision quality. Instead of embedding past experience in context, Xiang et al. (2024) combines LLM-generated action likelihoods with value estimates produced by an external RL critic. Other approaches incorporate classical planning components to enhance reasoning and long-horizon planning (Liu et al. 2023). Beyond RL-based training, memory-augmented architectures offer an alternative by enabling LLMs to retrieve and reuse past experiences (Lewis et al. 2020; Mao et al. 2020; Cai et al. 2022). By leveraging external memory modules, agents can maintain relevant information across interactions, reduce forgetting, and generate more coherent and context-aware plans over time.

Hierarchical Planning

Hierarchical task planning has emerged as an effective paradigm for addressing complex long-horizon decision-making problems by decomposing the given task into manageable subgoals. By organizing the problem at multiple levels of abstraction, this decomposition allows an agent to reason and act in a more structured and systematic manner. AdaPlanner (Sun et al. 2023) and DEPS (Wang et al. 2023) adopt bi-level hierarchical planning architectures.

HiPlan (Li et al. 2025) achieves hierarchical planning through milestone-based goal decomposition, while STEP Planner (Zhou et al. 2025) uses a leaf termination model that relies on real-time environment feedback to regulate subgoal expansion. Choi et al. (2025) introduces hierarchical taks planning framework where a complex goal can be recursively decomposed into manageable subgoals, whose execution is coordinated via control-flow nodes in a dynamically constructed agent tree.

Search Based Planning with LLMs

Search-based planning methods augments Large Language Models(LLMs) with search algorithms to explore multiple reasoning or action trajectories before committing to a final decision. By integrating the generative and reasoning capabilities of LLMs with systematic exploration strategies such as tree search or Monte Carlo methods these approaches (Besta et al. 2024; Hao et al. 2023; Wang et al. 2022; Xie et al. 2023; Yao et al. 2023a), demonstrates reduced brittleness compared to greedy LLM execution and improved execution and improve robustness in complex decision-making tasks. Zhuang et al. (2024) employ an A* search algorithm to explore the space of tool functions, enabling the discovery of effective sequences of tool calls that achieve a given goal. Zhou et al. (2024) applies MCTS over flat ReAct-style trajectories using LLM-powered value function while Xu et al. (2025) also applies MCTS to explore flat React-style trajectories and collect preference pairs from searched rollouts, which are subsequently used for preference optimisation.

In this work, we propose to combine the search-based exploration with the hierarchical planning by integrating Monte Carlo Tree (MCTS) which enables LLM agent to explore actions across multiple thinking direction. By Combining MCTS with hierarchical task planning, the agent can solve complex and long horizon problems by dividing in manageable subgoals while using simulation-based search to optimise decision making. This approach bridges knowledge-driven decomposition with data-driven search.

Methodology

ReAct-based hierarchical planning frameworks like Choi et al. (2025) significantly improves the reasoning structure by introducing the recursive subgoals decomposition, they typically commits greedily to a single decomposition $(f, \mathcal{S})$ at each agent node and resolves primitive actions via a one-shot ReAct loop without explicit search. This greedy commitment can be suboptimal in complex, long-horizon tasks, where an early decomposition choice or an initial primitive action may lead to irreversible failures, and alternative action sequences within the same subgoal are never explored. We propose ReAcTree-MCTS, a decoupled hierarchical framework that separates high-level planning from low-level execution. The method operates in two stages:

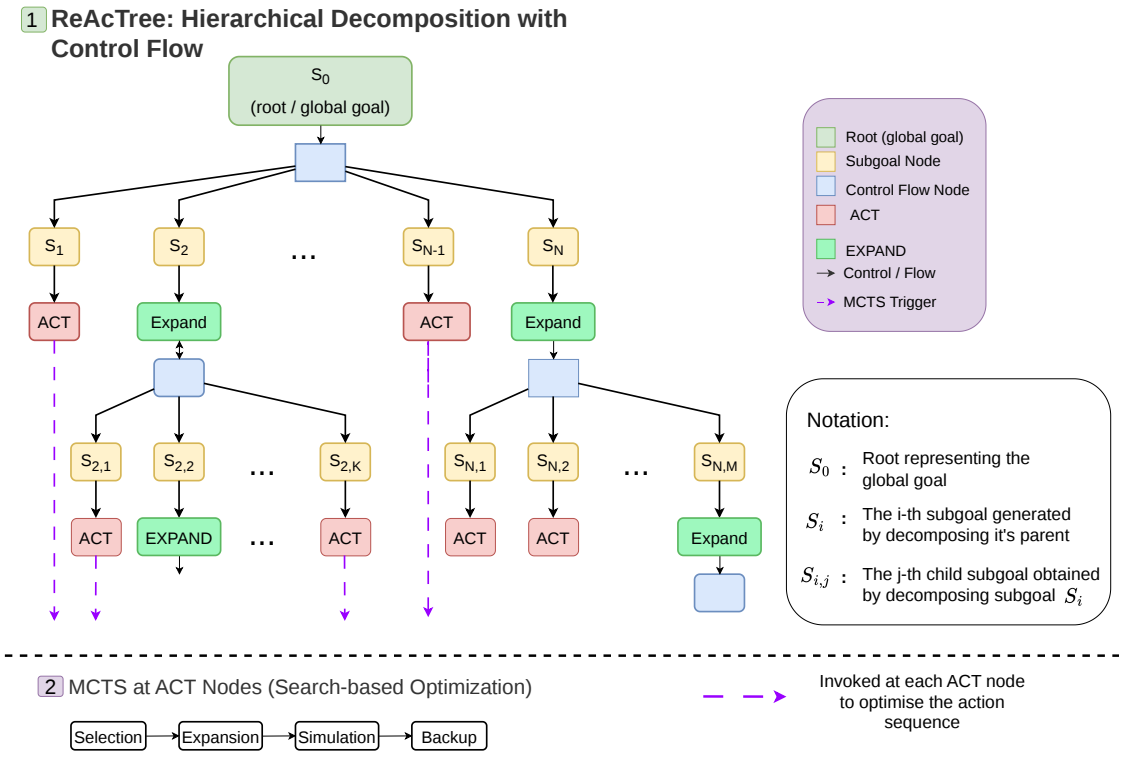

Figure 1: ReAcTree-MCTS. The agent recursively decomposes a goal into subgoals with control flow, while applying MCTS to primitive action execution within subgoals.

- Hierarchical subgoal generation via ReAcTree
- Targeted action optimization via Monte Carlo Tree Search (MCTS)

Hierarchical subgoal generation via ReAcTree Each agent node n operates as an LLM-based task planner with a specific natural language subgoal S_n, responsible for decision-making to achieve that goal. At each time step t, it access its context $c_{n_t} = (o_{n_1}, a_{n_1}, o_{n_2}, a_{n_2}, \ldots, a_{n_{t-1}}, o_{n_t})$, where o_{n_i} and a_{n_i} represent the observation and action at time step i. It then samples an action a_{n_t} from LLM policy $p_{\text{LLM}}(\cdot)$:

$$a_{n_t} \sim p_{\text{LLM}}(\cdot|P_n, S_n, c_{n-1}), \tag{1}$$

where the initial prompt $P_n = P_{sys}$ is a system prompt. The action space is extended to include the subgoals, $\hat{\mathcal{A}}_{n_t} = \mathcal{A}_{n_t} \cup \mathcal{L} \cup \mathcal{E}$, where $\mathcal{A}_{n_t}$ is the set of executable or low-level actions provided by the environment, $\mathcal{L}$ is the language space for self-reasoning; and $\mathcal{E} = \mathcal{F} \times \mathcal{L}$ is the expand space, with $\mathcal{L}$ and $\mathcal{F}$ represent the language space used to express subgoals and set of control flow types, respectively. Control flow nodes are responsible for coordination between the sequence of subgoals generated by the agent node. There are three types of supported control flow: sequence ($\rightarrow$), fallback ($?$), and parallel ($\Rightarrow$). The sequence node executes the subgoals sequentially and return success only when all of the subgoals returned success and returns failure immediately if any of the subgoals fails. The fallback node executes the subgoals sequentially but returns success as soon as one child returns success and returns failure only when all subgoals fails. The parallel node executes all subgoals sequentially irrespective of their success and failure and then aggregates all outcomes. Once, the execution of control flow nodes is completed it returns it success or failure to it's parent node. The details of the approach can be found at (Choi et al. 2025).

Targeted action optimization via Monte Carlo Tree Search (MCTS) At each node, when an LLM agent node decides to resolve a subgoal S_i directly via primitive actions, ReAcTree (Choi et al. 2025) selects this action greedily – sampling a single action $a_{n_t} \sim p_{LLM}(\cdot|P^n, g^n, c_t^n)$ and executing it in the environment without evaluating alternatives. While ReAcTree's `fallback` control flow node provides a mechanism for subgoal-level recovery by attempting alternative subgoals after failure, it operates one level above primitive action space and only activates *reactively* after a subgoal has already failed in the environment. Critically, no mechanism exists in ReAcTree to evaluate competing primitive actions *before* committing to one at timestep t. This greedy commitment is particularly harmful in long-horizon tasks where single suboptimal primitive action can irrecoverably alter the environment state, rendering subsequent subgoals unachievable. To address this, we apply a policy-guided MCTS (Xu et al. 2025) at the primitive action level. The search proceeds through four phases: *selection,*

expansion, simulation, and back-propagation.

Selection. Each node in the MCTS tree represents an environment state s_i for subgoal S_i and each edge represents a candidate primitive action $a \in \mathcal{A}_{n_t}$. The $\mathcal{A}_{n_t}$ is provided by the evironment for each state s_i. The MCTS algorithm recursively selects child nodes until reaching a leaf node using Predictor + Upper Confidence bounds applied to Trees(PUCT):

$$a_t^* = \arg\max_{a_t^{(k)} \in \mathcal{C}_t} \Bigg[Q(s_i, a_t^{(k)}) + c_{puct} \cdot p_{\text{LLM}}\Big(a_t^{(k)} \mid s_i, o_t\Big) \cdot \frac{\sqrt{N(s_i)}}{1 + N(s_i, a_t^{(k)})} \Bigg] \tag{2}$$

where $Q(s_t, a)$ is the estimated action value, $N(s_t)$ is the visit count of state s_t, $N(s_t, a)$ is visit count of action a at state s_t, c_{puct} is a coefficient, $p_{\text{LLM}}(\cdot|s_t)$ is action distribution given s_t.

Expansion. It introduces new candidate actions proposed by the LLM.

Simulation. Rolls out trajectories using the LLM as a policy prior.

$$r_0(s_{\text{terminal}}) = \begin{cases} 1.0, & \text{if } s_T = \texttt{True}, \\ r_{\text{env}}, & \text{else if } r_{\text{env}}(s_T) > 0, \\ R_h(g, \mathcal{O}), & \text{otherwise.} \end{cases} \tag{3}$$

where g is the current subgoal, $r_{env} \in [0, 1]$ is reward returned by the environment, and R_h is a partial reward assigned over the full observation history $\mathcal{O} = o_1, o_2, \cdots, o_t$.

Back-propagation. Rewards are propagated back through action path.

From a computational standpoint, our framework lowers the effective search complexity relative to flat MCTS. A flat MCTS approach has an exponential cost of $O(b^d)$ over the entire action space, where b denotes the branching factor and d the planning horizon. In contrast, our method decomposes the task into smaller subproblems, leading to a total complexity approximated by $\sum_i O(b^{d_i})$, where each subgoal horizon $d_i \ll d$. This localized exploration improves practical efficiency, with some additional overhead due to repeated MCTS executions within individual subgoals.

Experiments and Discussion

In our experiments, we evaluate the proposed approach on the ALFWORLD benchmark (Shridhar et al. 2020a), a text-based embodied environment derived from the ALFRED dataset. ALFWorld models household tasks in interactive indoor scenes, where an embodied agent follows natural language instructions to complete multi-step goals. At each step, the agent receives a textual state description and executes discrete high-level actions such as navigation, object manipulation, and environment interaction. We evaluate all task-planning frameworks using the **Goal Success Rate (GSR)** metric. GSR is defined as the percentage of tasks in which the agent successfully achieves the final goal, computed over all goal instructions in the test dataset. For our experiments, we compare our approach against ReAct, ReAcTree, and a flat MCTS baseline on a held-out test set of 100 goal instructions from the ALFWorld dataset. We sample 100 problems by selecting one representative instance from each grounded task template directory in the ALFWorld valid seen set, which collectively span a large number of distinct task variations.

Method	Goal Success Rate
React	19%
Reactree	24%
MCTS	28%
Reactree-MCTS	65%

Table 1: Comparison of different task planning agent frameworks for ALFWorld dataset.

We use Llama-3.1-8B-Instruct as our agent interacting with the ALFWorld environment. In **stage 1: Hierarchical subgoal generation via ReAcTree**, we set parameters to $(max\_depth = 4, max\_steps = 20 and max\_decision = 30)$, where max_depth is maximum depth of control-flow node tree, max_steps is maximum number of environment steps allowed per node-budget and $max\_decisions$ is the maximum number of *Think/Act/Expand* calls per agent node.

In **stage 2: Targeted action optimisation via MCTS**, We configure $(computation\_budget = 3, max\_sim\_steps = 10)$, where $computation\_budget$ specifies the number of MCTS iterations per *Act* decision, and max_sim_steps controls the maximum number of environments steps per simulation rollout.

From our analysis over 100 goal instruction, we observe that the number of environment steps required to successfully complete the task ranges between 4 and 12, In only two cases, the agent did not perform subgoal expansion and instead directly selected Act actions. In one such instance, the agent entered a loop and failed to make progress, while in other, it successfully completed the task using only 4 consecutive *Act* steps without hierarchical decomposition.

Conclusion

In this work, we introduced a decoupled hierarchical planning framework for LLM-based agents that separates high-level planning from low-level execution. By generating structured subgoals with explicit control flow and performing targeted exploration using MCTS within each subgoal, our approach enables more effective decision-making in long-horizon tasks. This formulation allows the agent to balance abstraction and exploration, reducing error propagation and improves the efficiency of search. Our results highlights the advantages of combining hierarchical decomposition with localized search, the approach still depends on the

quality of subgoal generation and introduces computational overhead. Future work will focus on improving decomposition strategies, optimizing search efficiency, and extending the framework to more complex and dynamic environments.

References

Baktash, J. A.; and Dawodi, M. 2023. Gpt-4: A review on advancements and opportunities in natural language processing. *arXiv preprint arXiv:2305.03195*.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; Nyczyk, P.; et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 17682–17690.

Cai, D.; Wang, Y.; Liu, L.; and Shi, S. 2022. Recent advances in retrieval-augmented text generation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 3417–3419.

Choi, J.-W.; Kim, H.; Ong, H.; Yoon, Y.; Jang, M.; Kim, J.; et al. 2025. Reactree: Hierarchical task planning with dynamic tree expansion using llm agent nodes.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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 Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 8154–8173. Singapore: Association for Computational Linguistics.

Jang, Y.; Lee, J.; and Kim, K.-E. 2022. GPT-critic: Offline reinforcement learning for end-to-end task-oriented dialogue systems. In *International Conference on Learning Representations*.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474.

Li, Z.; Chang, Y.; Yu, G.; and Le, X. 2025. Hiplan: Hierarchical planning for llm-based agents with adaptive global-local guidance. *arXiv preprint arXiv:2508.19076*.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

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.

Mao, Y.; He, P.; Liu, X.; Shen, Y.; Gao, J.; Han, J.; and Chen, W. 2020. Generation-augmented retrieval for open-domain question answering. *arXiv preprint arXiv:2009.08553*.

Miao, N.; Teh, Y. W.; and Rainforth, T. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*.

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.

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.

Shridhar, M.; Thomason, J.; Gordon, D.; Bisk, Y.; Han, W.; Mottaghi, R.; Zettlemoyer, L.; and Fox, D. 2020a. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10740–10749.

Shridhar, M.; Yuan, X.; Côté, M.-A.; Bisk, Y.; Trischler, A.; and Hausknecht, M. 2020b. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Springenberg, J. T.; Abdolmaleki, A.; Zhang, J.; Groth, O.; Bloesch, M.; Lampe, T.; Brakel, P.; Bechtle, S.; Kapturowski, S.; Hafner, R.; et al. 2024. Offline actor-critic reinforcement learning scales to large models. *arXiv preprint arXiv:2402.05546*.

Sun, H.; Zhuang, Y.; Kong, L.; Dai, B.; and Zhang, C. 2023. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36: 58202–58245.

Tang, Q.; Deng, Z.; Lin, H.; Han, X.; Liang, Q.; Cao, B.; and Sun, L. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

Vavilala, V.; Jain, S.; Vasanth, R.; Bhattad, A.; and Forsyth, D. 2023. Blocks2world: Controlling realistic scenes with editable primitives. *arXiv preprint arXiv:2307.03847*.

Wang, D.; and Li, L. 2023. Learning from mistakes via cooperative study assistant for large language models. *arXiv preprint arXiv:2305.13829*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wang, Z.; Cai, S.; Chen, G.; Liu, A.; Ma, X.; and Liang, Y. 2023. Describe, explain, plan and select: Interactive planning with large language models enables open-world multitask agents. *arXiv preprint arXiv:2302.01560*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xiang, Y.; Shen, Y.; Zhang, Y.; and Cam-Tu, N. 2024. Retrospex: Language Agent Meets Offline Reinforcement Learning Critic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 4650–4666.

Xie, J.; Zhang, K.; Chen, J.; Zhu, T.; Lou, R.; Tian, Y.; Xiao, Y.; and Su, Y. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*.

Xie, Y.; Kawaguchi, K.; Zhao, Y.; Zhao, J. X.; Kan, M.-Y.; He, J.; and Xie, M. 2023. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36: 41618–41650.

Xu, H.; Yu, Z.; Tang, Y.; Hu, P.; Tang, Y.; and Dong, H. 2025. MCTS-EP: Empowering Embodied Planning with Online Preference Optimization. *arXiv preprint arXiv:2509.17116*.

Yang, Z.; Li, P.; Yan, M.; Zhang, J.; Huang, F.; and Liu, Y. 2024. React meets actre: When language agents enjoy training data autonomy. *arXiv preprint arXiv:2403.14589*.

Yao, S.; Chen, H.; Yang, J.; and Narasimhan, K. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35: 20744–20757.

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)*.

Zhang, D.; Chen, L.; Zhang, S.; Xu, H.; Zhao, Z.; and Yu, K. 2023. Large language models are semi-parametric reinforcement learning agents. *Advances in Neural Information Processing Systems*, 36: 78227–78239.

Zhao, A.; Huang, D.; Xu, Q.; Lin, M.; Liu, Y.-J.; and Huang, G. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19632–19642.

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 Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 62138–62160. PMLR.

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. *arXiv preprint arXiv:2307.13854*.

Zhou, T.; Wang, Z.; Ao, H.; Chen, G.; Xing, B.; Cheng, J.; Yang, Y.; and Yue, Y. 2025. STEP Planner: Constructing cross-hierarchical subgoal tree as an embodied long-horizon task planner. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 16731–16738. IEEE.

Zhuang, Y.; Chen, X.; Yu, T.; Mitra, S.; Bursztyn, V.; Rossi, R.; Sarkhel, S.; and Zhang, C. 2024. Toolchain*: Efficient action space navigation in large language models with a* search. In *International Conference on Learning Representations*, volume 2024, 4524–4549.