PRIME: Planning with Reflective, Iterative, Multi-agentic Exploration

Chelsea Zou
Stanford University
cyzou@stanford.edu

Samuel Liu
Stanford University
samzliu@stanford.edu

Jui Khankari
Stanford University
juik@stanford.edu

Abstract

Top-down planning remains unsolved, with existing reasoning techniques applied in an ad-hoc manner. We propose an algorithm for Planning with Reflective, Iterative, Multi-agentic Exploration (PRIME), that frames multi-step planning in LLMs as an option discovery problem in reinforcement learning (RL). Using Monte Carlo Tree Search (MCTS), PRIME decomposes tasks, selects optimal reflective reasoning strategies, and dynamically executes specialized subagents to solve each task. Our structured approach outperforms the current state-of-the-art on accuracy across Webshop, PlanBench, and Game of 24, and works surprisingly well for small LLMs with limited compute.

1 Introduction

Top-down planning is currently an unsolved problem. Due to the combinatorial complexities of different ways to achieve a goal, much of the research has instead focused on sequential search to explore bottom-up planning trajectories. Various techniques have been developed to improve the multi-step reasoning capabilities such as reflexion, debate, voting, and self-feedback [1, 2, 3]. However, when used in isolation, these approaches are applied in an ad-hoc manner, lacking a principled way to determine which techniques are most effective for an overall problem type. These techniques follow a step-by-step framework, with vague guidance and signaling to assess its trajectory towards the final goal. Overall, it lacks a structured top-down approach to systematically select the best techniques given a planning task.

We propose an algorithm for planning with reflective, iterative, multi-agentic exploration (PRIME), to frame multi-step reasoning in LLMs as an option discovery problem in reinforcement learning (RL), which can lead to more computationally efficient and effective reasoning strategies. We treat these different reasoning techniques as 'options' and dynamically define subtasks to select the most suitable reasoning framework for each subtask, leading to better overall performance. Specifically, we decide to use a Monte-Carlo Tree Search (MCTS) approach, which we call PRIME, to learn how to decompose a complex reasoning problem into subtasks, select the best framework for each subtask, and instantiate self-reflective subagents that solve them efficiently. We hypothesize that this "smarter" options-based reasoning framework will outperform existing methods that indiscriminately combine techniques, particularly for small LLMs with constrained computational budgets.

Our evaluation will focus on different planning benchmarks (WebShop, Game of 24, and Planbench), measuring both accuracy and efficiency gains. The main contribution of this project is that it formalizes the combination of reasoning techniques into a structured, learnable framework, rather than relying on trial-and-error or manually designed heuristics.

Preprint. Under review.

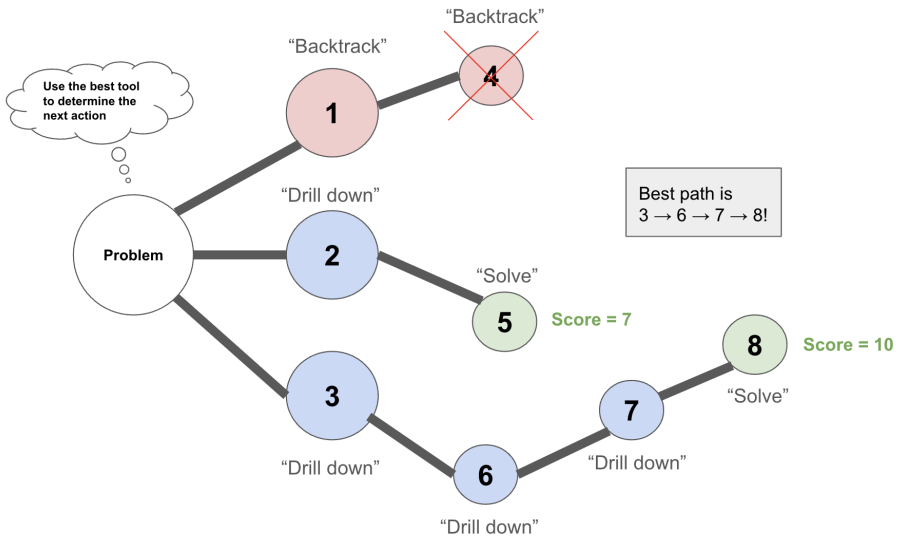

Figure 1: Planning framework

2 Related Work

Recent advancements in combining reasoning and acting within language models have focused on integrating structured search, feedback, and temporal abstraction. We summarize four key works below.

Planning efficiency depends on how many steps it takes to find a good strategy. Wan and Sutton (2022) [6] introduce a new objective in RL to make planning faster by finding better subsets of actions to choose from at each time step, what they frame as "option discovery". Their approach enables agents to make more strategic decisions, significantly improving planning efficiency in four-room domains by reducing the need to evaluate individual actions. At a high level, we adopt this "option discovery" heuristic in our algorithm to search for the best reflective tools to execute at each intermediate subgoal.

Recent advancements in LLMs have explored incorporating reasoning techniques to enhance decision-making. Wei et al. (2022) [7] propose Chain-of-Thought (CoT) prompting, which encourages LLMs to generate intermediate reasoning steps before arriving at a final answer. This approach led to substantial improvements in mathematical reasoning, commonsense understanding, and symbolic logic tasks.

Building on this, Yao et al. (2023) [10] developed the Reasoning and Acting in Language (ReAct) framework, integrating CoT reasoning with environmental interactions to improve decision-making. Their approach enabled models to refine their responses iteratively, outperforming standard CoT reasoning in question answering and web navigation. However, ReAct lacked a structured search mechanism, limiting its adaptability in complex problem-solving scenarios.

To address these limitations, Zhou et al. (2024) [11] introduced Language Agent Tree Search (LATS), which integrates MCTS with self-reflection and external feedback to improve reasoning efficiency. LATS allows models to explore multiple reasoning paths, backtrack when necessary, and refine their approach dynamically. Despite these advancements, LATS primarily relies on heuristics to select reasoning strategies and does not explicitly decompose problems into modular, reusable subtasks, which could enhance generalization across different domains.

Our approach builds upon these prior works by introducing a structured recursive planning framework that systematically decomposes complex problems into actionable subgoals. By integrating MCTS with a dynamic selection of reasoning strategies—including self-reflection, debate, and self-refinement—we enable more adaptive and efficient problem-solving while maintaining flexibility in strategy selection.

3 Methods

Our approach extends the idea of LATS (the current state of the art) by dynamically selecting the optimal reflective reasoning strategies from a given set (as opposed to applying the same reasoning strategy to every problem). This approach allows for a more flexible and adaptive response based on each unique state. An overview of our novel process is shown in Figure 1.

Planning Decomposition: Given an initial problem, PRIME invokes a language model to generate N plausible actionable subgoals that break down the problem and when combined will allow us to solve the final goal state. In this stage, a tree structure is initialized, where child nodes represent subgoals. At each node, the planner evaluates its state, selects a self reflection tool to use (e.g., self-reflection, self-refinement, debate) and takes one of the following actions:

- **Drill Down:** If the current subgoal is too broad or vague, the planner further decomposes it into finer-grained subgoals, recursively breaking down the problem until reaching well-defined atomic tasks.
- **Solve:** When a node reaches a base-case state and becomes a leaf node (determined by either a ground truth or our value function for more subjective plans), this means it has reached (or attempted to reach) the final goal. The planner then asks the most appropriate self-improvement agent tool to solve directly. The selection is guided by a reasoning prompt that asks: "What is the best tool from the following set to accomplish this subgoal?"
- **Backtrack:** If the current subgoal is deemed irrelevant or ineffective for solving the overarching problem, the planner abandons it and backtracks to explore alternative subgoals, ensuring efficient resource allocation and preventing wasted computation.

Backpropagation: The outcomes of each step are propagated back through the tree, updating parent node values to reinforce effective strategies and discourage ineffective ones. This allows the planner to iteratively improve its decision-making. The node values are updated based on the outcome, using:

$$N(s_i) = N(s_{i-1}) + 1, \quad V(s_i) = \frac{V(s_{i-1}) \cdot N(s_{i-1}) + r}{N(s_i)}.$$

where $N(s_i)$ is the number of visits to node s_i, $V(s_i)$ is the value of node s_i, and r is the reward for reaching the terminal state s_i.

Iterative Refinement: The system traverses the most promising paths and selects the most promising node, balancing exploration and exploitation using the upper confidence bound (UCT) algorithm in MCTS:

$$UCT(s) = \frac{w_i}{n_i} + C\sqrt{\frac{\ln N(s)}{n_i}}.$$

where w_i is the cumulative reward of node i, n_i is the number of visits to node i, $N(s)$ is the total number of visits to the parent node s, and C is the exploration constant. The process repeats until the most effective plan is identified, continuously adapting by learning new reasoning patterns and integrating successful solutions.

Value Function: Base-case leaf states are evaluated using a langauge model based self-judgement technique to verify how well defined the subgoal is towards the final goal. Each non-leaf child node is assigned a value based on the langauge model reasoning and self-consistency, defined as:

$$V(s) = \lambda \cdot LM(s) + (1 - \lambda) \cdot SC(s).$$

where λ is a hyperparameter controlling the balance between the LM-based value and self-consistency, $LM(s)$ is the language model score for state s, and $SC(s)$ is the self-consistency score for state s.

4 Experiments

Baseline Comparison

We evaluated four model configurations: PRIME with GPT 4o-mini, LATS with GPT 4o-mini, brute force with GPT 4o-mini, and one-shot with GPT-o1.

Table 1: Success Rates on Three Benchmarks

Method	PlanBench	WebShop	Game of 24
PRIME	36.2%	40%	75%
LATS	2.2%	38%	44%
GPT-4o-mini	0%	0%	0%
GPT-o1	100%	100%	100%
PRIME upgraded planner (4o)	95%	94%	100%
PRIME upgraded execution (4o)	96%	92%	100%

4.1 Data

To demonstrate the general applicability of PRIME, we evaluate our method on three distinct domains that require reasoning and acting: PlanBench [4], WebShop [8], and Game of 24 [9]. Due to limited compute, we randomly sample 60 questions, or 60 runs of Game of 24.

4.2 Evaluation method

Counterfactual Evaluation: To evaluate whether our planning approach truly grounds LLM reasoning, rather than merely providing post-hoc justification, we test whether structured planning grounds LLM reasoning by introducing controlled perturbations and measuring their propagation through the execution chain.

Reverse-Ablation We conducted a "reverse-ablation" study by selectively upgrading individual components from GPT-4o-mini to the more capable GPT-o1 model. This approach reveals which parts of the system would benefit most from increased model capability, where we systematically upgraded each component while keeping the others constant.

4.3 Experimental details

PlanBench: For a task that requires both high-level planning and adaptive strategy selection, we evaluate our method on PlanBench (Valmeekam et al., 2022), a benchmark designed to assess the reasoning capabilities of language models in sequential decision-making environments. PlanBench requires agents to generate structured plans that adaptively combine multiple reasoning strategies for effective task completion. Due to limited compute, we randomly sampled 60 questions from PlanBench. This evaluation took about 3 hours to complete.

Webshop: For a complex decision-making setting with real-world implications, we utilize WebShop (Yao et al., 2022), an online shopping platform featuring 1.18 million authentic products and 12,000 human-authored instructions. In this environment, agents are tasked with browsing the website using a set of predefined commands to locate and purchase an item that aligns with a given user specification. The action space is composed of search and click commands, while the observation space includes browser feedback along with agent-generated reflections. We randomly sampled 60 queries on WebShop in about 30 minutes. To evaluate performance, we rely on the success rate, which represents the proportion of tasks where the selected item meets all specified criteria.

Game of 24: To show how PRISM can be applied to purely internal reasoning tasks, we additionally evaluate on Game of 24 (Yao et al., 2023a), a mathematical reasoning task where the agent must construct 24 out of a set of numbers and basic operations. We ran 60 iterations of Game of 24 in about 5 minutes.

4.4 Results

Planbench: PRIME achieves a success rate of 36.2% on PlanBench, a 2.2% improvement over LATS (the current SOTA), far surpassing 4o-mini's performance of 0%, and approaching GPT-o1's performance of 37.3% [5].

Webshop: PRIME achieved a success rate of 40%, while 4-o1 achieved a success rate of 80% and LATS (current SOTA) achieved a success rate of 38% [11].

Game of 24: PRIME achieved a success rate of 75% on the Game of 24, which far outperformed the current SOTA of 44% (achieved by LATS) and approached 4-o1's performance was 100% [11].

Thus, in all three settings, PRIME resulted in performance gains, sometimes matching the larger model of GPT-o1 despite the approximately 10x less parameter count of GPT-4o-mini. While the hierarchical planner uses more API calls than the GPT-o1, this represents an estimated[1] 4x decrease in cost using the OpenAI pricing chart as of December 2024.

This performance increase across multiple domains is consistent with our hypothesis that different planning are best suited for different types of problems, as PRIME is the only framework that can adapt its planning process based on the type of problem. The results of all four methods, including the reverse ablation study (last two rows) are also noted in Table 1.

5 Analysis

PlanBench: While planning can slightly enhance performance on questions requiring reasoning, larger gains were observed with search methods since we could sample and explore more outputs [10]. We noticed that increasing the number of nodes n consistently improved performance, although at greater computational cost. PRIME outperformed LATS due to the subgoal-generation process combined with the tree search (i.e. adopting a top-down vs bottom-up approach). We observed that PRIME consistently struggled with tasks in which the utility of individual actions was unclear — that is, when it was difficult to determine whether a given action contributed meaningfully to the overall goal. In these cases, the framework's behavior indicated that it repeatedly explored different trees without clear direction, suggesting it was likely getting trapped in local minima.

On the contrary, PRIME performed exceedingly well in situations where the outcome of each action was clear. Example questions and responses for both cases are included below (denoted obfuscated and non-obfuscated). In the non-obfuscated case, when the outputs of the actions were clear, despite the fact that it was a multi-step reasoning task, the framework was able to successfully generate a plan. This was not the case for the case where the outputs of the actions were unclear. We hypothesize this is likely due to the fact that the framework couldn't fully or accurately "unroll" the tree, since it wasn't clear where each action lead. This resulted in none of the sub-goals being met, and a plan not being generated at all. Truncated examples of both types of inputs are located in Figures 2 and Figures 3.

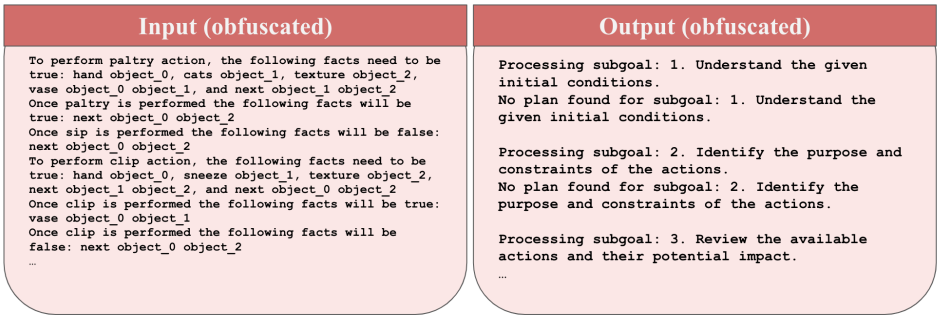

Figure 2: Obfuscated Query

Webshop: Sampling k = 60 trajectories with PRISM and LATS resulted in a similar performance, suggesting subgoal generation is not as helpful in complex environments like WebShop. We find that generated reflections are often generic and do not provide useful feedback, resulting in a tendency for the agent to become stuck in local minima.

Game of 24: PRIME outperforms Language Agent Tree Search (LATS) in the Game of 24 because it directly addresses the game's inherent complexity and variability through dynamic selection of reasoning tools tailored to each puzzle. Unlike LATS, which applies a uniform planning approach, PRIME utilizes an adaptive planning approach that decomposes each puzzle into strategically actionable subgoals (e.g., identifying promising arithmetic combinations or decomposing numerical

[1]Exact difference depends on tokens and number of steps required for each question

Input (non-obfuscated)

```
I am playing with a set of objects. Here are the actions
I can do

Paltry object_0 object_1 object_2.
Sip object_0 object_1 object_2.
Clip object_0 object_1 object_2.
Wretched object_0 object_1 object_2 object_3.
Memory object_0 object_1 object_2.
Tightfisted object_0 object_1 object_2.

I have the following restrictions on my actions:
To perform paltry action, the following facts need to be
true: hand object_0, cats object_1, texture object_2,
vase object_0 object_1, and next object_1 object_2
Once paltry is performed the following facts will be
true: next object_0 object_2
Once paltry is performed the following facts will be
false: vase object_0 object_1
```

Output (non-obfuscated)

```
[STATEMENT]
As initial conditions I have that, cats object_0, cats
object_1, collect object_10 object_2, collect object_11
object_3, collect object_12 object_3, collect object_13
object_3, collect object_14 object_4, collect object_15
…

My plan is as follows:

[PLAN]
clip object_23 object_7 object_15
clip object_18 object_7 object_15
clip object_17 object_7 object_15
wretched object_7 object_15 object_14 object_4
tightfisted object_23 object_7 object_14
sip object_20 object_1 object_14
```

Figure 3: Non-obfuscated Query

targets). By recursively breaking down vague or complex numerical operations ("Drill Down") and selectively solving simpler arithmetic expressions directly ("Solve"), the planner efficiently navigates toward the exact combination that achieves the goal number.

Counterfactual analysis: Our counterfactual analysis demonstrated that changes to the initial planning steps propagate systematically through the execution chain, suggesting that our framework achieves meaningful grounding of LLM reasoning. When we introduced controlled modifications to early planning steps, we observed corresponding changes in both the subgoal generation and final outputs, indicating that the sub-agents genuinely follow the planning agent's directives. However, while we've established a causal relationship between plans and outputs, the semantic alignment between the plan's language and human interpretations of those instructions remains an important area for future investigation.

6 Conclusion and Future Work

Our approach provides a robust framework for addressing complex problems by combining recursive planning with a dynamic selection of reasoning strategies. By leveraging a recursive tree structure and applying a range of self-improvement tools, the planner decomposes problems into actionable subgoals, ensuring systematic exploration of potential solutions. The iterative process of allows the system to adapt and learn from each step, ultimately identifying the most effective plan. The integration of a value function, driven by LLM-based self-judgement, ensures that each subgoal is rigorously evaluated, leading to more accurate and efficient problem-solving strategies. We have shown marked performance over the current state of the art (LATS) and have also created a structured method to apply different reasoning frameworks, as compared to the previously-used ad-hoc approach.

In future work, we plan to introduce several techniques that may improve performance.

1. **Clustering problems:** We will evaluate different types of problems present in the training set, cluster similar ones, and label them accordingly. By doing so, we can optimize the planning process by associating the most effective set of tools with each problem label. During testing, problems that resemble those in the labeled clusters will be able to leverage the optimal subset of tools, improving the system's efficiency and accuracy in solving similar problems.
2. **Improved Self-Reflection Techniques:** We aim to expand the range of self-reflection tools available to the planner, allowing for more nuanced evaluation of subgoals and strategies. By incorporating more advanced reasoning mechanisms, such as meta-cognitive processes, we expect to refine the decision-making process, leading to better goal decomposition and more effective tool selection.
3. **Adaptive Value Function:** Another area for improvement is the adaptation of the value function over time. As the system encounters a wider variety of problems, it could dynamically adjust the criteria used to evaluate subgoals based on ongoing performance, leading to more accurate prioritization of tasks and refinement of the overall planning process.

Team contributions

All three members ideated on the the overall framework. Chelsea implemented the baseline codes, Samuel amd Jui implemented the planning and dataset codes.

References

[1] Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R Bowman, Tim Rocktäschel, and Ethan Perez. Debating with more persuasive llms leads to more truthful answers. *arXiv preprint arXiv:2402.06782*, 2024.

[2] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

[3] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[4] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *arXiv preprint arXiv:2206.10498*, 2022.

[5] Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *arXiv preprint arXiv:2409.13373*, September 2024.

[6] Yi Wan and Richard S Sutton. Toward discovering options that achieve faster planning. *arXiv preprint arXiv:2205.12515*, 2022.

[7] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. 2023.

[8] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

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

[10] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2022.

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