

# Appendix

## A PlanU Algorithm

### A.1 Algorithm

The PlanU algorithm is described in Algorithm 1.

---

#### Algorithm 1 The PlanU Algorithm

---

```

1: Require: Initial state  $s_0$ , environment dynamics  $D$ , prior action probability  $\pi : S \times A \rightarrow \mathbb{R}$ ,
   number of iterations  $I$ , number of quantiles  $N_q$ , depth limit  $L$ , prior weight  $\alpha$ ,
2: Initialize memory:  $c_a : A \rightarrow S$ ,  $c_s : S \rightarrow A$ , quantile probabilities  $\tau = \{\tau_1, \tau_2, \dots, \tau_{N_q}\}$ 
3: Initialize: value distribution  $F : S \times A \rightarrow \mathbb{R}^{N_q}$ , and state uncertainty networks  $\hat{f}, f$ 
4: EXPANSION( $s_0$ )
5: for  $i \leftarrow 1$  to  $I$  do
6:    $t \leftarrow 0$ 
7:   while  $s_t$  is not a leaf state node do
8:      $a_t \leftarrow \arg \max_{a \in c_s(s_t)} UCC(s_t, a)$ 
9:      $(s_{t+1}, \text{done}, r_t) \leftarrow D(s_t, a_t)$ 
10:     $t \leftarrow t + 1$ 
11:    if  $s_{t+1} \notin c_a(a_t, *)$  then
12:       $c_a(a_t, s_{t+1}) \leftarrow s_{t+1}$ 
13:    end if ▷ Selection
14:  end while
15:  while not done and  $t < L$  do
16:    EXPANSION( $s_t$ )
17:     $a_t \leftarrow \arg \max_{a \in c_s(s_t)} UCC(s_t, a)$ 
18:     $(s_{t+1}, \text{done}, r_t) \leftarrow D(s_t, a_t)$ 
19:     $t \leftarrow t + 1$  ▷ Simulation
20:  end while
21:  Update uncertainty estimation network  $\hat{f}$  with  $\{s_1, \dots, s_t\}$ 
22:  BACK-PROPAGATION
23: end for

```

---

### A.2 Inference Scaling and Vanilla MCTS

Inference-time scaling increases the performance of LLM through increasing computational resources and time used during the inference phase. Multiple inference-time scaling methods are proposed, such as the majority voting, best-of-n, and tree searches. Among them, the Monte Carlo Tree Search (MCTS) technique is one of the most widely utilized tree search methods. It has been used for LLM decision-making and reasoning, such as RAP [8], LATS [6], and TS-LLM [4].

In vanilla MCTS-based LLM planning (i.e., [8]), during planning, the LLM builds a search tree by iteratively selecting the most promising next steps (i.e., actions). It uses a world model to predict future states and rewards. The estimated rewards are used to update the value of the search tree, which improves the planning ability of LLM. Although MCTS-based LLM demonstrated remarkable planning capabilities through the use of MCTS methods, it may fail in simple planning problems with stochastic state transition dynamics.

Vanilla MCTS assumes a deterministic relationship among the next states and actions. When determining the child state node for a state  $s_t$  node, the child state node is determined deterministically rather than stochastically once an action  $a_t$  is taken from the state  $s_t$ . Figure 6 illustrates a failure case of an MCTS-based LLM agent for basketball shooting. In MCTS-based LLM [8, 5], when adding a state node into the search tree, an LLM-based world model is queried multiple times, and the most frequently appearing state is used as the state. For Figure 6, the player has a 60% chance of scoring; thus, the scoring state is returned. However, this state transition is not true, which could hurt MCTS performance. The deterministic state transitions assumption is valid in deterministic environments

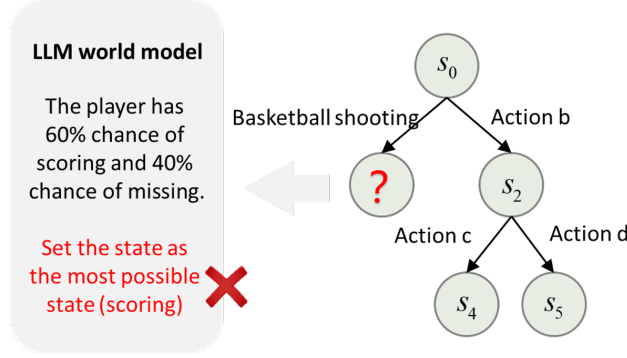


Figure 6: A basketball shooting example, where MCTS fails to model the stochastic state transition. Shooting can lead to the scoring or the missing state, but the vanilla MCTS only models the scoring state.

such as Go but is not valid in real-world environments. Due to the simplified assumption, MCTS performs poorly for stochastic environments.

## B Baseline Methods and PlanU variants

We consider five categories of methods used for comparison: (1) Prompt-based methods: Chain-of-Thought (CoT) and Reflexion; (2) Tree-search methods: ToT, RAP, LATS; (3) Tree-search methods considering uncertainty: RAP-D, RAP-D-UCC, RAP-E; (4) LLM-based decision making method under uncertainty: DeLLMa; (5) Reinforcement Learning method: QRDQN.

1. **CoT** [1]: a prompt-based method that guide LLM to perform chain-of-thought reasoning<sup>1</sup>.
2. **ToT** [3]: an LLM-based method performs deliberate decision-making inside a search tree<sup>2</sup>. It searches inside a tree whose nodes represent thoughts.
3. **Reflexion** [33]: a prompt-based method that considers environmental feedback, which is used to improve its performance<sup>3</sup>.
4. **DeLLMa** [9]: an LLM-based single-step decision-making method under uncertainty based on classical decision theory<sup>4</sup>.
5. **RAP** [8]: an LLM-based method that utilizes MCTS for strategic exploration<sup>5</sup>.
6. **RAP-D**: a variant of RAP designed by us. We replaces the MCTS component in RAP by DMCTS [48], which learns a posterior distribution over the utility of the different possible returns.
7. **RAP-D-UCC**: a variant of RAP-D. We combines RAP-D with the UCC scores proposed in this work.
8. **RAP-E**: a variant of RAP designed by us. We replaces the MCTS component in RAP by EMCTS [49], a method that considers epistemic uncertainty in decision-making<sup>6</sup>.
9. **LATS** [6]: an LLM-MCTS based decision making method which integrates reflection and API use into the MCTS search process<sup>7</sup>.
10. **QRDQN**[16]: a Reinforcement Learning method using quantile distribution to model uncertainty<sup>8</sup>.

<sup>1</sup>CoT: <https://github.com/Ber666/llm-reasoners>

<sup>2</sup>ToT: <https://github.com/princeton-nlp/tree-of-thought-llm>

<sup>3</sup>Reflexion: <https://github.com/noahshinn024/reflexion>

<sup>4</sup>DeLLMa: <https://github.com/DeLLMa/DeLLMa>

<sup>5</sup>RAP: <https://github.com/Ber666/llm-reasoners>

<sup>6</sup>EMCTS: <https://github.com/emcts/e-alpha-zero>

<sup>7</sup>LATS: <https://github.com/lapisrocks/LanguageAgentTreeSearch>

<sup>8</sup>QRDQN: <https://github.com/toshikwa/fqf-qn-qrdqn.pytorch>

928 In this work, we consider a few PlanU variants to evaluate impact of its design. These variants are  
929 listed as follows.

- 930 1. **PlanU w/o dist**: a variant of PlanU, where the quantile distribution are removed. It does not  
931 capture environmental uncertainty through quantile distribution.
- 932 2. **PlanU w/o UCC**: a variant of PlanU, where the UCC scores are removed, the second term  
933 in Equation 8 is replaced by the UCT exploration term. It does not fully consider LLM  
934 uncertainty.
- 935 3. **PlanU prompt shuffling**: a variant of PlanU, where the prompt sentences for the policy  
936 and the world model are shuffled, respectively. This is used to evaluate the impact of LLM  
937 uncertainty caused by prompt shuffling.
- 938 4. **PlanU prompt injection**: a variant of PlanU, its prompts are injected with task-irrelevant  
939 but environment-related information. However, these environment-related information are  
940 useless for the specific task. This is used to evaluate the impact of LLM uncertainty caused  
941 by prompt injection.

## 942 C Experiment Results

943 In this experiments, we compare PlanU with 10 different methods, and show that PlanU performs  
944 better than all of them in 3 LLM decision making benchmarks. We show that the quantile distribution  
945 and the UCC scores play important roles for dealing environmental and LLM uncertainty.

### 946 C.1 Setup and Computing Resources

947 In this work, all the experiments are repeated with 5 different seeds. For the baseline methods, we  
948 use their default configuration. We set the number of quantiles to 50. The  $c_1$  in Equation 8 is set to  
949 0.25, and the update rate of the quantile distribution is fixed at 0.5.

950 The experiments were conducted on high-performance computing clusters equipped with NVIDIA  
951 A40 GPU, each with 48GB of memory. The CPUs used in the cluster are Intel(R) Xeon(R) Silver  
952 4216 processors, each running at 2.10GHz. The memory of each computing node is 1024GB.

### 953 C.2 Blocksworld

954 In Blocksworld tasks [8, 50], the goal is to stack blocks on a table. In this environment, all the  
955 states and actions are text-based, and the environmental transition is deterministic. We introduce  
956 environmental uncertainty by adding 20% failure rate for actions: STACK, UNSTACK, PUT, and  
957 PICKUP. Failed actions keep the state intact. The tasks are categorized into three difficulty tiers  
958 according to the minimum required action steps: 2-step (37 tasks), 4-step (76 tasks), 6-step (145  
959 tasks), and 8-step (143 tasks).

#### 960 C.2.1 Task Description

961 The agent is placed in a Blocksworld environment, where the goal is to stack blocks on a table. Only  
962 one block can be moved at a time, and a block cannot be moved if there are other blocks on top of  
963 it. Similarly, a block cannot be stacked on another block that already has a block on top. The goal  
964 specifies the stacking order, which may include multiple stacks or require some blocks to remain on  
965 the table.

966 **Observation Space**: The environment is a world consisting of a set of blocks, each identified by a  
967 color. The agent can only observe the positions of the blocks and their clear status (whether a block  
968 is on top of another). The global state includes the positions and statuses of all blocks, and the agent  
969 can only observe a limited set of blocks within a defined radius. The initial positions of all blocks are  
970 known to the agent. The observation data is directly used as input for planning models such as LLMs,  
971 represented through symbolic data.

972 **Action Space**: The task includes four fundamental actions that the agent can perform:

- 973 • **STACK**: The agent stacks a block onto another block.

Table 2: The Success Rate of the Blocksworld Benchmark. Tasks are organized according to minimal steps ( $n$ -step) to finish the task.

Model	Method	Tasks ( $n$ -step)			
		2	4	6	8
Mistral-7B	CoT	0.514 $\pm$ 0.06	0.276 $\pm$ 0.05	0.131 $\pm$ 0.03	0.000 $\pm$ 0.00
	ToT	0.486 $\pm$ 0.06	0.355 $\pm$ 0.05	0.117 $\pm$ 0.03	0.028 $\pm$ 0.01
	RAP	0.892 $\pm$ 0.03	0.514 $\pm$ 0.06	0.166 $\pm$ 0.04	0.000 $\pm$ 0.00
	RAP-D	0.973 $\pm$ 0.02	0.632 $\pm$ 0.05	0.324 $\pm$ 0.05	0.105 $\pm$ 0.03
	RAP-E	<b>1.000 <math>\pm</math> 0.00</b>	0.592 $\pm$ 0.05	0.338 $\pm$ 0.05	0.084 $\pm$ 0.03
	PlanU	<b>1.000 <math>\pm</math> 0.00</b>	<b>0.803 <math>\pm</math> 0.04</b>	<b>0.559 <math>\pm</math> 0.04</b>	<b>0.217 <math>\pm</math> 0.03</b>
LLama3.1-8B	CoT	0.351 $\pm$ 0.05	0.237 $\pm$ 0.05	0.124 $\pm$ 0.03	0.014 $\pm$ 0.01
	ToT	0.459 $\pm$ 0.06	0.263 $\pm$ 0.05	0.131 $\pm$ 0.03	0.056 $\pm$ 0.02
	RAP	0.946 $\pm$ 0.02	0.553 $\pm$ 0.05	0.255 $\pm$ 0.05	0.175 $\pm$ 0.03
	RAP-D	0.973 $\pm$ 0.02	0.750 $\pm$ 0.04	0.428 $\pm$ 0.05	0.070 $\pm$ 0.02
	RAP-E	0.946 $\pm$ 0.02	0.763 $\pm$ 0.04	0.414 $\pm$ 0.05	0.140 $\pm$ 0.03
	PlanU	<b>1.000 <math>\pm</math> 0.00</b>	<b>0.842 <math>\pm</math> 0.04</b>	<b>0.524 <math>\pm</math> 0.04</b>	<b>0.238 <math>\pm</math> 0.03</b>
DeepSeek-R1-Distill-Llama-8B	CoT	0.405 $\pm$ 0.06	0.158 $\pm$ 0.04	0.152 $\pm$ 0.04	0.077 $\pm$ 0.03
	ToT	0.514 $\pm$ 0.06	0.237 $\pm$ 0.05	0.145 $\pm$ 0.04	0.034 $\pm$ 0.02
	RAP	<b>1.000 <math>\pm</math> 0.00</b>	0.724 $\pm$ 0.04	0.200 $\pm$ 0.04	<b>0.196 <math>\pm</math> 0.03</b>
	RAP-D	0.973 $\pm$ 0.02	0.750 $\pm$ 0.04	0.400 $\pm$ 0.05	0.140 $\pm$ 0.03
	RAP-E	<b>1.000 <math>\pm</math> 0.00</b>	0.697 $\pm$ 0.04	0.448 $\pm$ 0.05	0.175 $\pm$ 0.03
	PlanU	<b>1.000 <math>\pm</math> 0.00</b>	<b>0.816 <math>\pm</math> 0.04</b>	<b>0.455 <math>\pm</math> 0.05</b>	<b>0.196 <math>\pm</math> 0.02</b>

• **UNSTACK:** The agent unstack a block from another block.

• **PUT:** The agent places a block on the table.

• **PICKUP:** The agent picks up a block from the table.

Each action can only be executed if the preconditions are met (e.g., the block is clear or the agent is holding the block). The agent must learn to sequence these actions in order to achieve the goal.

**Dynamics:** The transitions in this environment are deterministic but with some stochasticity introduced in the Stochastic Blocksworld setting. In this setting, the four fundamental actions (STACK, UNSTACK, PUT, PICKUP) have a 20% chance of failure during execution. If an action fails, the state remains unchanged, introducing uncertainty into the state transitions.

**Reward:** A sparse reward setting is used in which the agent only receives a +1 reward upon completing the goal. The task completion is defined as achieving the specified stacking order or leaving certain blocks on the table as required.

**Episode Termination:** Each episode terminates when the agent successfully completes the goal or reaches the maximum number of time steps. If the agent reaches the goal before the maximum time steps, it is rewarded, and the episode ends. If the maximum time steps are reached without completing the task, the episode ends without a reward.

## C.2.2 Experimental Results for Blocksworld

We conduct experiments on the Blocksworld tasks on three large language models. They are Mistral-7B, Llama3.1-8B, and DeepSeek-R1-Distill-Llama-8B. The experimental results are shown in Table 2. Some of the results are already shown in the main paper, for completeness and ease of reading, we have placed the results together.

As it is shown in Table 2, PlanU performs the best across all the tasks for the three LLMs. RAP performs better than ToT, which performs better than CoT. As DeLLMa does not consider environment feedback, thus it is not suitable for this task that requiring multiple steps of decision making. We have compared our approach against the four MCTS-based LLM approaches: RAP, RAP-D, RAP-E. The RAP variants (i.e., RAP-D, RAP-E), which consider environmental uncertainty, perform better than RAP. PlanU performs better than them, thanks to its ability to handle environmental uncertainty and LLM uncertainty.

### Comparing with LLM-based Decision-making method under uncertainty: DeLLMa

We evaluate DeLLMa in the Blocksworld environment with stochastic state transitions on Mistral-7B. As DeLLMa is not specifically designed for multi-step decision-making, we treat each planning task as a single-step decision problem. The prompt for DeLLMa is depicted in Appendix D. DeLLMa performs multi-step reasoning by integrating recent inference-time techniques grounded in decision and utility theory to produce accurate and auditable outcomes.

Table 3: The Success Rate of the Blocksworld Benchmark: DeLLMa, CoT, and ToT

Method	Step 2	Step 4	Step 6	Step 8
CoT	$0.514 \pm 0.062$	$0.276 \pm 0.051$	$0.131 \pm 0.027$	$0.000 \pm 0.000$
ToT	$0.486 \pm 0.062$	$0.355 \pm 0.055$	$0.117 \pm 0.026$	$0.028 \pm 0.014$
DeLLMa	$0.595 \pm 0.081$	$0.092 \pm 0.033$	$0.041 \pm 0.017$	$0.000 \pm 0.000$
PlanU	$1.000 \pm 0.000$	$0.803 \pm 0.045$	$0.559 \pm 0.041$	$0.217 \pm 0.034$

As shown in Table 3, DeLLMa outperforms CoT and ToT in short-horizon tasks (e.g., Step 2), demonstrating its effectiveness in simple reasoning scenarios. However, its performance deteriorates significantly as the number of required steps increases, primarily due to its heavy reliance on the LLM’s reasoning ability—where uncertainty compounds over longer reasoning chains—and its lack of explicit modeling for multi-step decision-making (lack of considering environmental feedback).

Our method PlanU maintains consistently high accuracy across all steps, demonstrating superior capability in handling complex, long-horizon decision-making tasks.

### Comparing with LLM-based MCTS Decision-making method: LATS

Table 4: The Success Rate of the Blocksworld Benchmark: LATS and RAP variants

Method	Step 2	Step 4	Step 6	Step 8
RAP	$0.892 \pm 0.050$	$0.513 \pm 0.057$	$0.166 \pm 0.031$	$0.000 \pm 0.000$
RAP-D	$0.973 \pm 0.022$	$0.632 \pm 0.053$	$0.324 \pm 0.049$	$0.105 \pm 0.030$
RAP-E	<b><math>1.000 \pm 0.000</math></b>	$0.592 \pm 0.054$	$0.338 \pm 0.052$	$0.084 \pm 0.031$
LATS	$0.946 \pm 0.037$	$0.684 \pm 0.053$	$0.352 \pm 0.040$	$0.077 \pm 0.023$
PlanU	$1.000 \pm 0.000$	$0.803 \pm 0.045$	$0.559 \pm 0.041$	$0.217 \pm 0.034$

We evaluate the performance of LATS in the stochastic Blocksworld environment on Mistral-7B. As shown in Table 4, LATS consistently outperforms RAP across different planning horizons. This improvement can be attributed to its *reflection* mechanism, which allows the language model to reason over its past trajectories and revise future actions accordingly. Such a mechanism indeed enhances robustness in the face of uncertainty. However, compared to our approach, which leverages value distribution modeling for uncertainty estimation, LATS still lacks the precision required for long-horizon planning. The performance gap becomes more pronounced as the number of steps increases, highlighting the limitations of reflection-based reasoning in capturing fine-grained stochastic dynamics.

## C.3 Overcooked

In the Overcooked environment [51, 7], agents prepare and deliver tomato salad and tomato-lettuce salad using provided resources. The states and actions are text-based. There are two tasks in this environment: Tomato salad, and Tomato lettuce salad. The original environment assumes deterministic transition, we add uncertainty into the environments by adding 20% failure rate for the Chop action. A failed action leads the state unchanged.

### C.3.1 Task Description

The agent is placed in the Overcooked kitchen and aims to cook a specified dish using the provided ingredients and tools, delivering it to the "star" cell as soon as possible. The agent must learn the correct procedure, including picking raw vegetables, chopping them, combining the chopped ingredients in a bowl, and then delivering the dish.

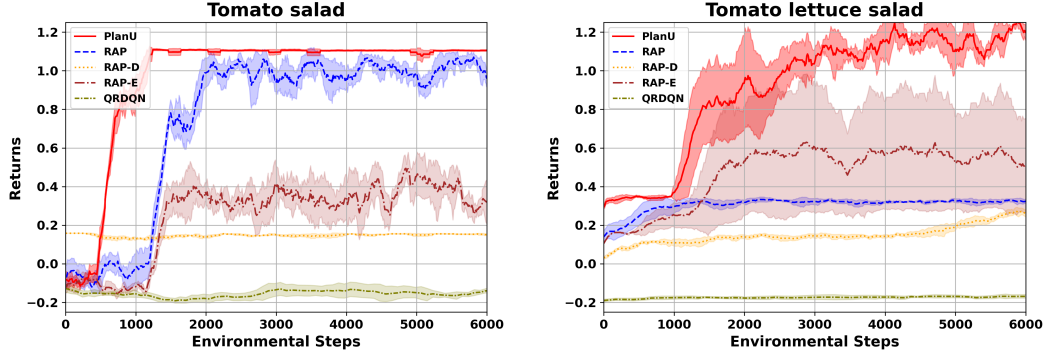


Figure 7: Experimental Results for Overcook: Tomato salad (left) and Tomato lettuce salad (right) in Overcooked.

**Observation Space:** The environment is a  $7 \times 7$  grid world that includes ingredients, bowls, cutting boards, and a delivery counter. In the tomato salad task, the environment provides one tomato, one bowl, one cutting board, and one delivery counter. In the tomato-lettuce salad task, the environment provides one tomato, one lettuce, one onion, one bowl, two cutting boards, and one delivery counter. The onion serves as an interference, though it does not appear in the recipe. The global state information includes the positions of the agent and the objects, as well as the status of each ingredient (chopped or unchopped). The environment is partially observable, and the agent can only observe the positions and statuses of objects within a  $5 \times 5$  square centered around the agent. Other unseen objects are masked in the observation. The initial positions of all objects are known to the agent. The raw observation data is converted into prompts through scripts, which are then used as input for the LLM.

**Action Space:** This work mainly focuses on using high-level macro-actions, as they usually have richer semantics. Each macro-action may take several time steps to complete. The following is a brief description of the macro-actions used in this study:

- **Chop:** The agent stands next to a cutting board with an unchopped ingredient and chops it into pieces.
- **Get-Tomato, Get-Lettuce, Get-Onion, Get-Bowl, Go-Cutting-Board-1, and Deliver:** These actions navigate the agent to the location of the corresponding object and execute the corresponding operation. In the tomato salad task, the available macro-actions are Get-Tomato, Get-Bowl, Go-Cutting-Board-1, and Deliver. In the tomato-lettuce salad task, all macro-actions are valid.

**Dynamics:** The transitions in this task are deterministic with some randomness. The Chop action has a 20% chance of failing, in which case the state remains unchanged. If the agent delivers the wrong item, that item will be reset to its initial position.

**Reward:** +0.2 for chopping a correct ingredient, +1 terminal reward for delivering the correct dish, 0.1 for delivering the wrong dish, and 0.001 for every time step.

**Episode Termination:** Each episode terminates when the agent successfully delivers the target dish to the delivery counter or reaches the maximum time steps (200).

### C.3.2 Experimental Results for Overcooked

For ease of reading, we have partitioned the experimental results into Figure 7 and Figure 8. The experimental results for PlanU, RAP, RAP-D, RAP-E, and QRDQN are presented in Figure 7, while the results for RAP-D-UCC, CoT, and ToT are shown in Figure 8. In these figures, the horizontal and vertical axes represent environmental steps and returns, respectively. *PlanU achieves the best performance across both tasks.*

In Figure 7, RAP and RAP-E rank second on the *Tomato Salad* and *Tomato Lettuce Salad* tasks, respectively. Notably, on the *Tomato Lettuce Salad* task—which is more complex than *Tomato Salad*—PlanU is the only method that successfully completes the task, owing to its strong capability in

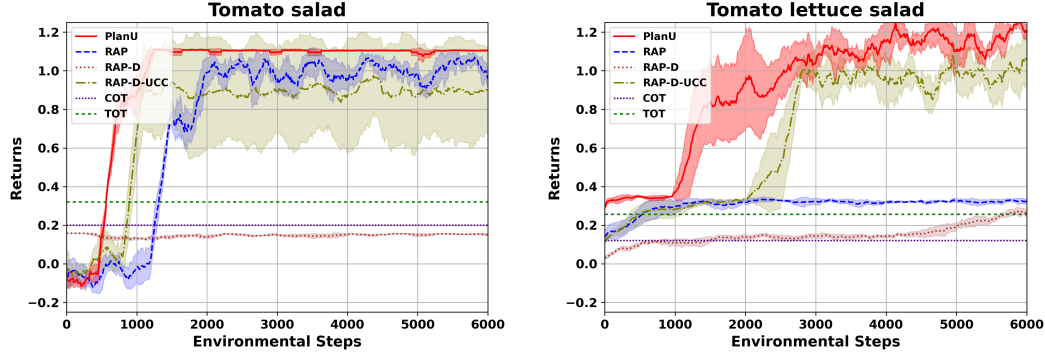


Figure 8: Experimental Results for Overcooked benchmark: Tomato salad (left) and Tomato lettuce salad (right).

handling decision-making uncertainty. Compared to RAP-D and RAP-E, PlanU demonstrates greater efficiency in exploring novel states via the UCC score, as well as more effective backpropagation through value distributions, ultimately leading to optimal reasoning paths.

In Figure 8, the results for CoT, ToT, and RAP-D-UCC. For COT and TOT, we adopt the same base prompt as used in PlanU, augmented with task-specific guiding prompts to enhance LLM reasoning. For each task, we allow the LLM to complete it within 15 steps and evaluate its performance over 10 trials on the same task. The reported result is the average reward across these trials. RAP-D-UCC is a RAP variant designed by us for this task. It considers environmental uncertainty by using DMCTS [48] and LLM uncertainty through using the UCC score (the state novelty part). As it is shown in the left part of Figure 8, it performs inferior to RAP-D. However, it performs better than RAP-D in the right part of Figure 8. This indicates that a *simple integration of methods deal with environmental uncertainty and LLM uncertainty does not work effectively*.

#### C.4 VirtualHome

The VirtualHome benchmark [52, 7] models a furnished house, where a simulated embodied agent execute actions to complete tasks. We evaluate two tasks: Food Preparation (heating a pancake in the microwave) and Entertainment (coordinating snack preparation with TV viewing). To increase the uncertainty of the tasks, we increase the failure rate for opening appliances (e.g., microwave) as 50%. Failed actions do not impact the environment.

##### C.4.1 Task Description

The agent, represented as a humanoid avatar, is placed in a fully furnished household with various rich objects to interact with. For the **Food Preparation** task, the agent needs to find the pancake on the table in the kitchen and place it in the microwave to heat, this is a relative simple task. For the **Entertainment** task, the agent needs to find and bring chips and milk from the kitchen to the living room and succeed in sitting on the sofa with the TV on and the chips and milk nearby. The challenge arises when the agent, already holding both the milk and chips, lacks an additional hand to turn on the TV. Consequently, the agent needs to learn to place at least one item on the nearby coffee table before operating the TV.

**Observation Space:** The environment is partially observable. The agent can only see the objects in the current room and cannot see the objects in the other room. The observation consists of a set of Boolean values, representing whether the agent sees the relative object, whether these objects are close to the agent, and the status of the objects, such as whether the TV is on and whether the milk is on the coffee table. The symbolic raw observations are converted to prompts with scripts to serve as input for the LLMs.

**Action Space:** This work mainly focuses on using high-level macro-actions, as they usually have richer semantics. Each macro-action may take several time steps to complete. The following is a brief description of the macro-actions used in this study:

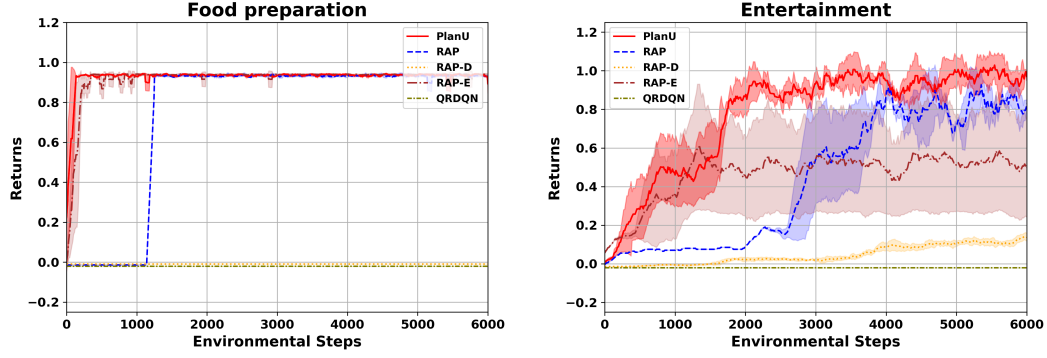


Figure 9: Experimental Results for VirtualHome: Food preparation (left) and Entertainment(right).

- **Get-Pancake:** The agent picks up the pancake from the table in the kitchen.
- **Place-Pancake-Microwave:** The agent places the pancake into the microwave and closes the microwave door.
- **Get-Chips:** The agent retrieves the chips from the kitchen.
- **Get-Milk:** The agent retrieves the milk from the kitchen.
- **Place-Item-Coffee-Table:** The agent places an item on the coffee table.
- **Sit-On-Sofa:** The agent sits on the sofa.
- **Turn-On-TV:** The agent turns on the TV.
- **Walk-to-A place:** The agent walks to a specific room including the living room, the bedroom, the kitchen and the bathroom.

**Dynamics:** The transitions in this task involve some randomness. In the food preparation task, due to its relative simplicity, there is a 50% failure probability when the agent attempts to open the microwave. In contrast, in the entertainment task, there is a 20% failure probability when the agent tries to get chips or get milk.

**Reward:** We adopt a sparse reward setting in Food preparation, where the agent only receives a +1 reward upon completing the task. In Entertainment, there are three subtasks: get chips, get milk and turn on the TV. The agent will get a +0.1 reward upon completing each of the sub task and a +1 reward if the agent sits on the sofa while every subtask is finished.

**Episode Termination:** Each episode terminates when the agent successfully completes the task or reaches the maximum time steps (50), as each macro-action takes one time step to execute. In the pancake heating task, the agent succeeds when it places the pancake in the microwave and closes the microwave door. In the TV watching task, the agent succeeds when it sits on the sofa, the TV is on, and the chips and milk are on the coffee table or held in hand.

#### C.4.2 Experimental Results for VirtualHome

Figure 9 shows that for both the Food preparation and the Entertainment tasks, *PlanU performs the best for these two tasks*. The left part of Figure 9 presents the performance of each algorithm on the *Food Preparation* task. As this task is relatively simple for large language models (LLMs), LLM-based methods are able to quickly discover the optimal path while RL method QRDQN fails to reach the optimal path in limited interactions with the environment. The Entertainment task is more difficult than the Food Preparation task, As it is shown in the right part of Figure 9, RAP performs the second.

#### C.5 Token Usage

PlanU is a MCTS-based LLM decision making task under uncertainty. As MCTS is computational intensive, in this section, we evaluate the computational resource consumption of PlanU through



1142 using the Token Usage and the Query times metrics. **Token usage** is the number of tokens consumed  
 1143 during the task, while **Query times** refers to the number of time the agent queries a LLM.

1144 We evaluate the performance of PlanU and its variants, RAP, CoT and, ToT over 10 trials on the  
 1145 Tomato Lettuce Salad task. For CoT and ToT, we prompt the LLM to find the optimal path within 15  
 1146 steps. The Token Usage and the Query time are reported in Table 5.

1147 PlanU consumes the least number of tokens and a number of queries among MCTS methods (e.g.,  
 1148 RAP), which demonstrates its ability to explore decision space and obtain the optimal path with less  
 1149 resource consumption. PlanUs consume nearly 3 times less tokens than RAP, a vanilla MCTS-based  
 1150 decision making method.

1151 Although PlanU consumes more tokens than CoT and ToT, it can find correct plan uncertainty,  
 1152 whereas CoT and ToT fail most of the time for the three benchmarks evaluated in this work.

Table 5: Token usage for the Tomato Lettuce Salad task

Method	PlanU	PlanU w/o dist	PlanU w/o UCC	RAP	COT	TOT
Token Usage(k)	922.69 $\pm$ 6.6	1197.99 $\pm$ 7.6	1521.86 $\pm$ 2.9	2643.84 $\pm$ 8.9	35.98 $\pm$ 3.2	159.85 $\pm$ 4.6
Query times	1389.6 $\pm$ 10.0	2353.0 $\pm$ 14.9	2289.2 $\pm$ 4.5	4313.0 $\pm$ 14.8	150.0 $\pm$ 0.0	150.0 $\pm$ 0.0

## 1153 C.6 Ablation Study

1154 We evaluate the impact of the quantile distribution and the UCC score. The results are presented in  
 1155 Figure 10. **PlanU w/o dist** refers to the variant of PlanU where the quantile distribution is replaced  
 1156 with the mean value. **PlanU w/o ucc** denotes the variant without the UCC score, in which the  
 1157 exploration term is replaced by that used in UCT.

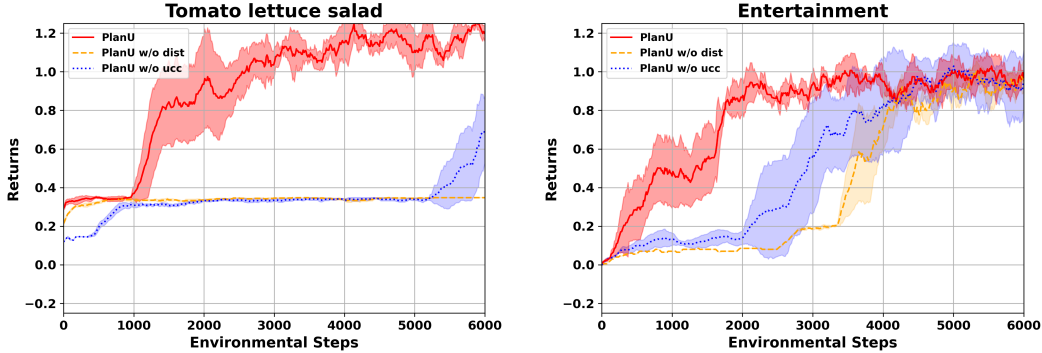


Figure 10: The impact of quantile distribution and UCC score.

## 1158 C.7 Impact of Uncertainty

1159 In this section, we evaluate whether PlanU matches the design goal that making good decision under  
 1160 uncertainty. We evaluate the impact of environmental uncertainty for PlanU in Appendix C.7.1, and  
 1161 the impact of LLM uncertainty in Appendix C.7.2. We find that PlanU can deal with environmental  
 1162 uncertainty well thanks to the use of quantile distributions; that PlanU is robust in the face of LLM  
 1163 uncertainty with different prompts and temperatures.

### 1164 C.7.1 Impact of Environment Uncertainty

1165 Figure 11 shows the performance for PlanU and PlanU w/o dist, a PlanU variant without value  
 1166 distribution under increased environmental randomness. With increased randomness (increased  
 1167 Action Failure Rate), the success rate of both methods decreases. With quantile distribution, PlanU  
 1168 enjoys a higher success rate than without.

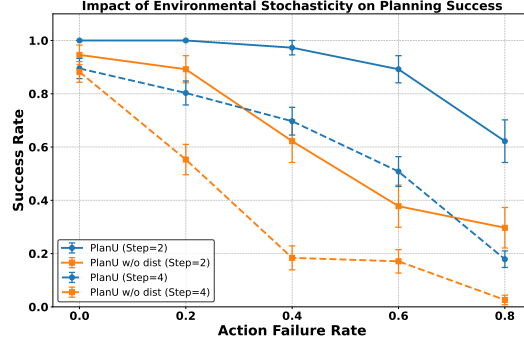


Figure 11: Impact of Environmental Uncertainty for PlanU and PlanU w/o dist. The horizontal axis depicts the action failure rate (AFR). The increase of the AFR lead to the increase of environmental uncertainty.

### 1169 C.7.2 Impact of LLM Uncertainty

1170 To evaluate the impact of LLM uncertainty, we evaluate the impact of using different prompts and the  
1171 impact of using different LLM temperatures.

### 1172 LLM Uncertainty: Changing Prompts

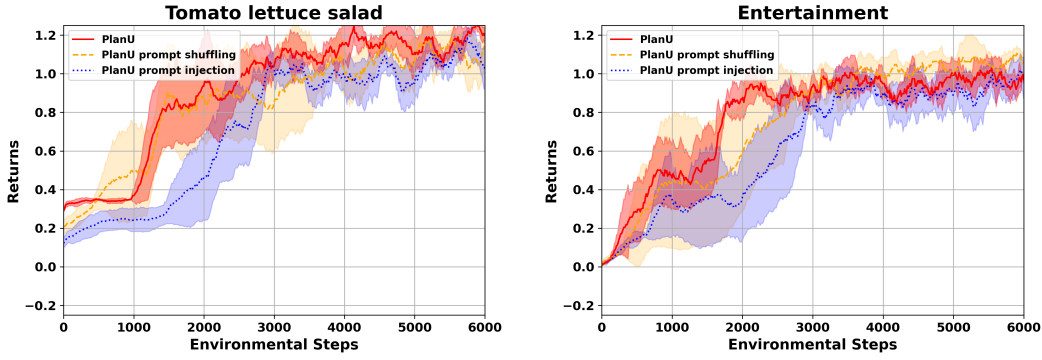


Figure 12: The impact of LLM uncertainty caused by changing prompts.

1173 To evaluate our method’s robustness to LLM uncertainty, we introduce two types of perturbations to  
1174 LLM prompts, which lead to different outputs from LLMs given the same state: (1) **Prompt**  
1175 **Shuffling**: We shuffle the sentences in the provided prompt while preserving the sentence boundaries  
1176 within the task and environment description. (2) **Prompt Injection**: We inject task-irrelevant but  
1177 environment-related information into the prompt that does not aid in solving the task. These forms of  
1178 uncertainty commonly arise when prompts include content generated by LLMs. Examples of such  
1179 prompts and results for the *Tomato Lettuce Salad* task are provided in Appendix D. The experimental  
1180 results are shown in Figure 12. As it is shown in the picture, the performance of PlanU does not  
1181 significantly affected by the changed prompts. These results demonstrate that LLM uncertainty  
1182 only slightly affects the convergence speed of PlanU. PlanU is robust to LLM uncertainty caused by  
1183 changed prompts.

### 1184 LLM uncertainty: Generation Temperatures

1185 To assess the robustness of planning strategies against the inherent uncertainty of large language  
1186 models (LLMs), we examine the effect of varying the temperature parameter—a key factor influencing  
1187 output stochasticity—on task success rates. The temperature  $T$  modulates the softmax distribution  
1188 over token logits  $z_i$  as follows:

$$P(x_i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \quad (7)$$

1189 where higher values of  $T$  lead to flatter distributions, increasing randomness in generated outputs.

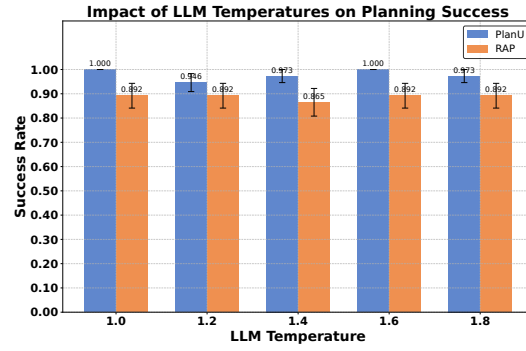


Figure 13: LLM Temperature for PlanU and RAP

1190 As illustrated in Figure 13, PlanU exhibits a high degree of resilience to temperature variation.  
1191 Across a range of temperature settings, the success rates remain largely stable, indicating that both  
1192 methods are capable of maintaining consistent performance despite increasing randomness in LLM  
1193 generations. PlanU consistently outperforms RAP under all temperature settings with shows lower  
1194 variance, demonstrating its stronger robustness in uncertain generation environments.

## 1195 D Prompts for LLM

1196 In this section, we describe the prompt we use for LLM-based methods for the Blocksworld, Over-  
1197 cooked(Fig 14) and Virtualhome(Fig 15).

1198 **Overcooked and Virtualhome.** For both environments, we basically follow the experimental settings  
1199 of [7], with additional uncertainty introduced in the prompts to evaluate our methods' capability to  
1200 model such uncertainty.

### EXAMPLE PROMPT FOR OVERCOOKED

Basic Prompt:

There is a fixed cutting board in the room. Currently you don't have anything in hand. You notice a tomato on the table. Your goal is to serve the dish of a bowl only containing chopped tomato. Your next step is to

Prompt After shuffling:

Currently you don't have anything in hand. Your goal is to serve the dish of a bowl only containing chopped tomato. There is a fixed cutting board in the room. You notice a tomato on the table. Your next step is to

Prompt after injection:

There are two fixed cutting boards in the room. Earlier this day, you made a dish with chopped potatoes. Currently you don't have anything in hand. You notice a tomato on the table. Your goal is to serve the dish of a bowl only containing chopped tomato. Your next step is to

Figure 14: Example Prompt for Overcooked

### EXAMPLE PROMPT FOR OVERCOOKED

Basic Prompt:

There are four rooms: the kitchen, bathroom, bedroom, and living room. You are in the living room and you notice a coffee table, a TV and a sofa. Currently, you are not grabbing anything in hand. Your goal is to enjoy the chips and the milk while watching TV. Your next step is to

Prompt after shuffling:

Your goal is to enjoy the chips and the milk while watching TV. There are four rooms: the kitchen, bathroom, bedroom, and living room. Your goal is to enjoy the chips and the milk while watching TV. You are in the living room and you notice a coffee table, a TV and a sofa. Your next step is to

Prompt after injection:

There are four rooms: the kitchen, bathroom, bedroom, and living room. Earlier in the day, you were in the bedroom and sowe cookies on the table. You are in the living room and you notice a coffee table, a TV and a sofa. Currently, you are not grabbing anything in hand. Your goal is to enjoy the chips and the milk while watching TV. Your next step is to

Figure 15: Example Prompt for Virtualhome.

#### EXAMPLE DELLMA PROMPT FOR BLOCKSWORLD

##### Prompt 1

I am interacting with a Blocksworld environment. I need to make optimal decisions about which action to perform next (e.g., pick up, stack, unstack, or put down a block). My goal is to place the blue block on top of the orange block. Today's environment is partially observable, and I must reason under uncertainty.

Here is the current observation:

- The blue block is clear.
- The orange block is clear.
- The hand is empty.
- The blue block is on the red block.
- The red block and the orange block are on the table.

The full world state is hidden and contains 16 latent variables (e.g., hand status, block locations, perception errors, etc.). *First, think about the unknown factors in the current environment that could affect my decision.*

##### Prompt 2

<SAME CONTEXT>

Now I have enumerated the unknown variables that affect my decision:

"block visibility", "block on block stability", "perception error", "hand grip strength", ...

*Given these unknowns, think about the belief distribution over their most likely values, with verbal confidences such as: "very likely", "likely", etc.*

Example format:

{"block visibility": {"blue": "very likely", "orange": "very likely"}, ...}

##### Prompt 3

<SAME CONTEXT>

Now, given the belief distribution over the environment state and the goal ("blue block on orange block"), devise a step-by-step plan.

You may use the following actions:

- (1) pick up a block
- (2) unstack a block from on top of another block
- (3) put down a block
- (4) stack a block on top of another block

Output one action per line. End the plan with [PLAN END].

Example Output:

unstack the blue block from on top of the red block  
stack the blue block on top of the orange block  
[PLAN END]

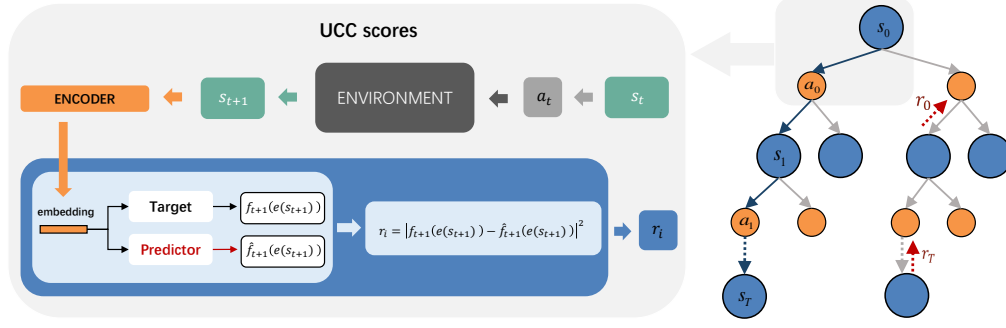


Figure 16: The Schematic Plot for the UCC score

## E UCC Score

During tree search, PlanU chooses optimism in the face of uncertainty; it encourages exploration through the Upper Confidence Bound with Curiosity (UCC) score. It calculates the uncertainty of text-based states based on the value distribution and novelty of states. The schematic plot for the selection phase using the UCC score is depicted in Figure 16. To estimate the novelty of a given state, we first encode the natural language description of the state using a sentence-level encoder. Specifically, we utilize Sentence Transformers<sup>1</sup> to encode textual states into dense embeddings. These embeddings are fed into a fixed target network and a learnable predictor network, with their output discrepancy quantifying the state’s novelty as a measure of epistemic uncertainty.

Specifically, the UCC score is defined as follows.

$$UCC(s_t, a_t) = \psi[Z(s_t, a_t)] + c_1 \cdot \frac{r_i(s_t)}{N(s_t, a_t)}, \quad (8)$$

where  $\psi[Z(s_t, a_t)]$  is a uncertainty-aware operator that maps a quantile distribution into a real-value. Multiple implementations of  $\psi$  can be used. In default, we use the expectation operator  $\mathbb{E}$  as  $\psi$ .  $c_1$  is a hyperparameter,  $N(s_t, a_t)$  represents the visit count of the action node  $a_t$ , and  $r_i(s_t)$  is a novelty reward.

The novelty reward  $r_i(s_t)$  is designed to encourage the agent to explore less-visited states from a global perspective through estimating the uncertainty of  $s_t$ . Inspired by [17],  $r_i(s_t)$  calculates the feature difference among a predictor network  $\hat{f}$  and a target network  $f$ . Specifically, it is defined as  $r_i(s_t) = |\hat{f}(e(s_t)) - f(e(s_t))|^2$ , where  $\hat{f}$  and  $f$  are both three-layer Multi-Layer Perceptron (MLP) networks.

Following [17], the hyperparameters used for the predictor network  $\hat{f}$  are listed in Table 6. We use  $r_i$  as the loss to train the predictor network.

Table 6: Hyperparameter Settings for UCC

Hyper-parameter	Value
learning_rate	1e-5
hidden_size_list	[64, 64, 128]
update_per_collect	5
obs_norm	True
obs_norm_clamp_min	-1
obs_norm_clamp_max	1
intrinsic_reward_weight	0.01

<sup>1</sup><https://huggingface.co/sentence-transformers/all-mpnet-base-v2>

## 1222 **F Broader Impact and Limitation**

1223 The goal of PlanU is to advance the field of Machine Learning technique for Decision Making. Our  
1224 work can be applied in multiple domains. However, we do not feel any thing must be highlighted  
1225 here.

1226 There are two limitations of PlanU. Firstly, PlanU does not use another tool (e.g., web search) during  
1227 decision-making. This may limit its ability for problems requiring external knowledge. In such cases,  
1228 a web search could help make the decision. We plan to combine PlanU and ReAcT to strengthen  
1229 its ability. Secondly, PlanU is an MCTS-based decision-making method. It faces challenges when  
1230 applied to problems with high-dimensional action. PlanU may struggle to explore sufficient paths  
1231 with practical time constraints, leading to suboptimal performance. We plan to explore hierarchical  
1232 MCTS, which decomposes the problem into hierarchical layers or combine PlanU with progressive  
1233 widening techniques, which dynamically limit the number of actions explored per node.