Tree-Planner: Efficient Close-loop Task Planning with Large Language Models

Mengkang Hu[1], Yao Mu[1], Xinmiao Yu[2], Mingyu Ding[1*], Shiguang Wu[3], Wenqi Shao[4], Qiguang Chen[2], Bin Wang[3], Yu Qiao[4], Ping Luo[1*]

[1] The University of Hong Kong, [2] Harbin Institute of Technology,
[3] Noah's Ark Laboratory, [4] Shanghai AI Laboratory

ICLR 2024

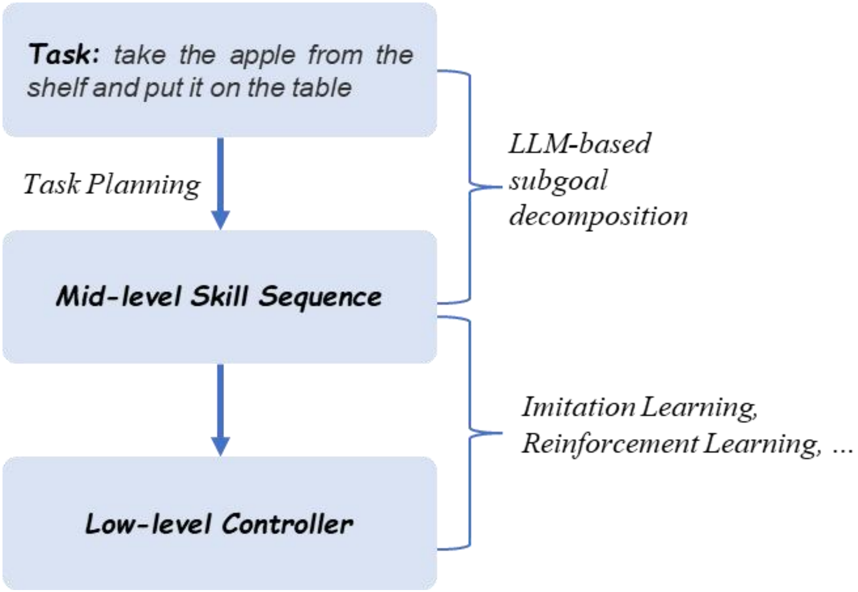
Logistics

- Background
  - Task Planning
  - Existing Methods
- Motivation
- Methodology
  - Plan Generation
  - Action Tree Construction
  - Grounded Deciding
- Experiments
  - Environment
  - Experimental Setup
  - Experimental Results
  - Analysis

Background

Task Planning

Decompose a high-level task description (microwave salmon) into a plan consisting of mid-level actions (open fridge, grab salmon, close fridge). We assume there is a low-level controller that can execute these mid-level actions (such as "grab cup").



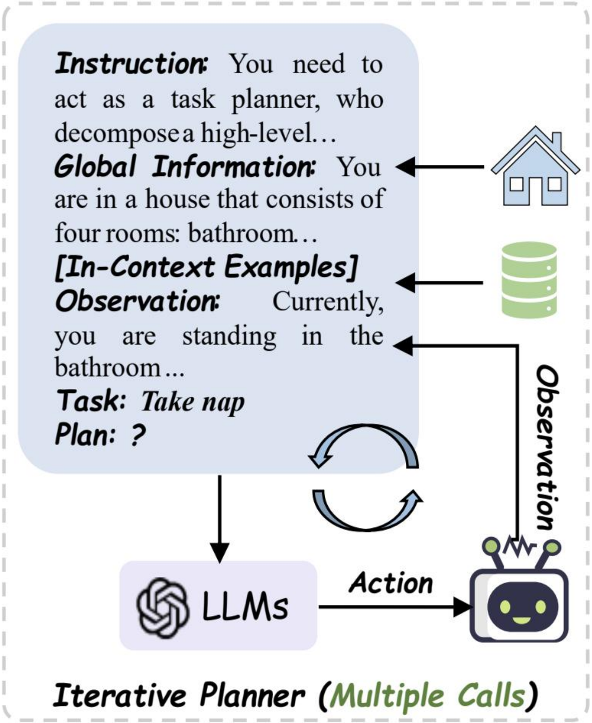
Existing Methods

1. Search-Based Methods:
    a. search in a pre-defined domain (hard to scale)[1]
    b. heuristics guided search[2]
    c. learning-based task planning (representation learning, hierachical learning)[3]
2. Generation-Based Methods: directly generate plans with LLMs
    a. generate an entire plan before execution. [4][5][6]
    b. dynamically generate actions at each timestep. (iterative planner) [7][8][9]

Existing Methods - Reference

[1] Task Planning in Robotics: an Empirical Comparison of PDDL-based and ASP-based Systems. 2018
[2] A heuristic search approach to planning with temporally extended preferences. 2007
[3] Hierarchical Planning for Long-Horizon Manipulation with Geometric and Symbolic Scene Graphs
[4] Visually-Grounded Planning without Vision: Language Models Infer Detailed Plans from High-level Instructions. 2020 ACL Findings
[5] Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. 2022 ICML
[6] Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language
[7] Do As I Can, Not As I Say: Grounding Language in Robotic Affordances 2022.4
[8] Grounded Decoding: Guiding Text Generation with Grounded Models for Robot Control

Existing Methods – Iterative Planner (2.b)



Pipeline:

(i) Prompt an LLM to generate one action at a time;

(ii) Execute the generated action and then append the obtained observation to the LLM;

(iii) Generate the next action.

When errors occur during action execution:

(i) re-generate actions at the current timestep;[1][2]

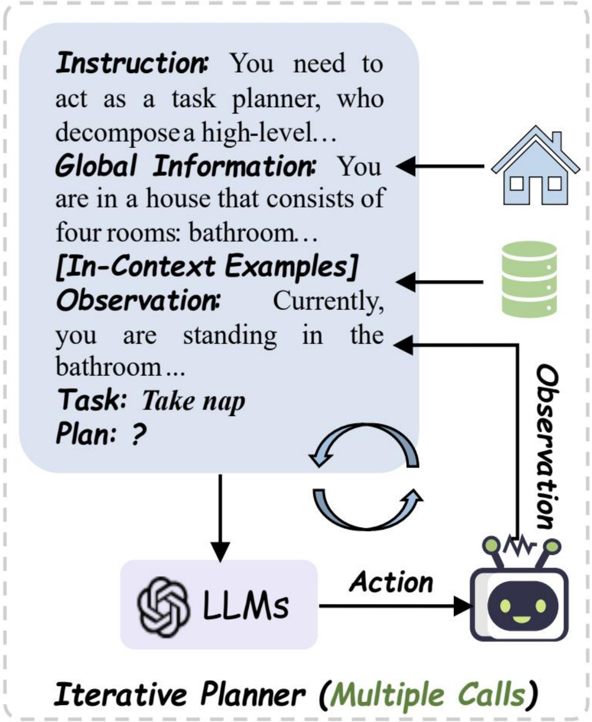
(ii) re-generate the entire plan from the initial timestep.[3]

[1] Planning with large language models via corrective re-prompting.
[2] Yanjiang Guo, Yen-Jen Wang, Lihan Zha, Zheyuan Jiang, and Jianyu Chen. Doremi: Grounding language model by detecting and recovering from plan-execution misalignment, 2023.
[3] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection.

Motivation – Existing Limitations of Iterative Planner

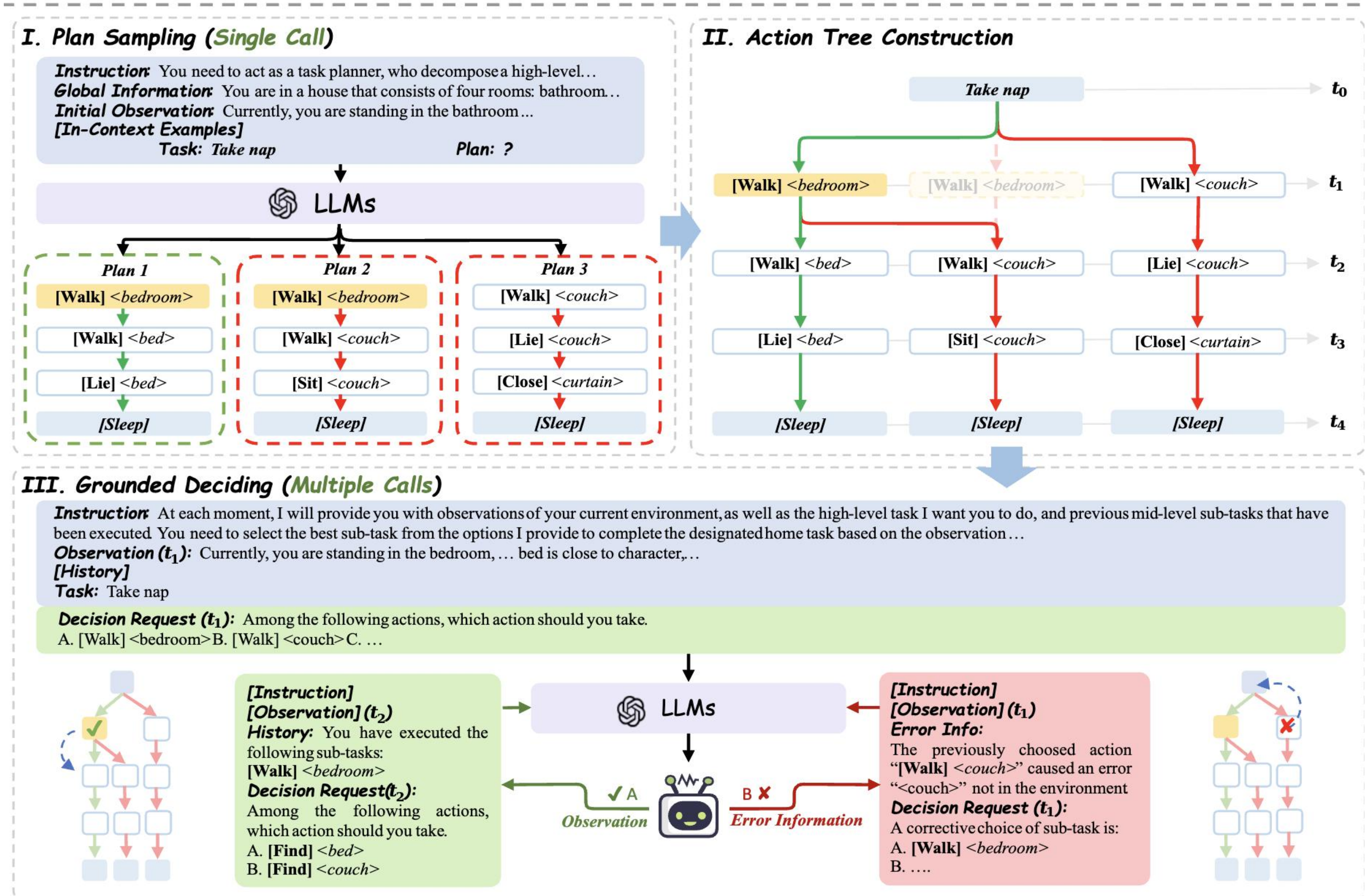
Token Inefficiency: Due to the multi-step nature of task planning (usually involving 5-20 steps), the prompt tokens incur repeated charges, leading to high costs of tokens (token inefficiency also relates to runtime inefficiency).

Correction Inefficiency: Replan with iterative planner can be viewed as a trial-and-error approach implemented at the execution-failed time step, which makes it difficult for the model to detect errors that occurred several time steps earlier.



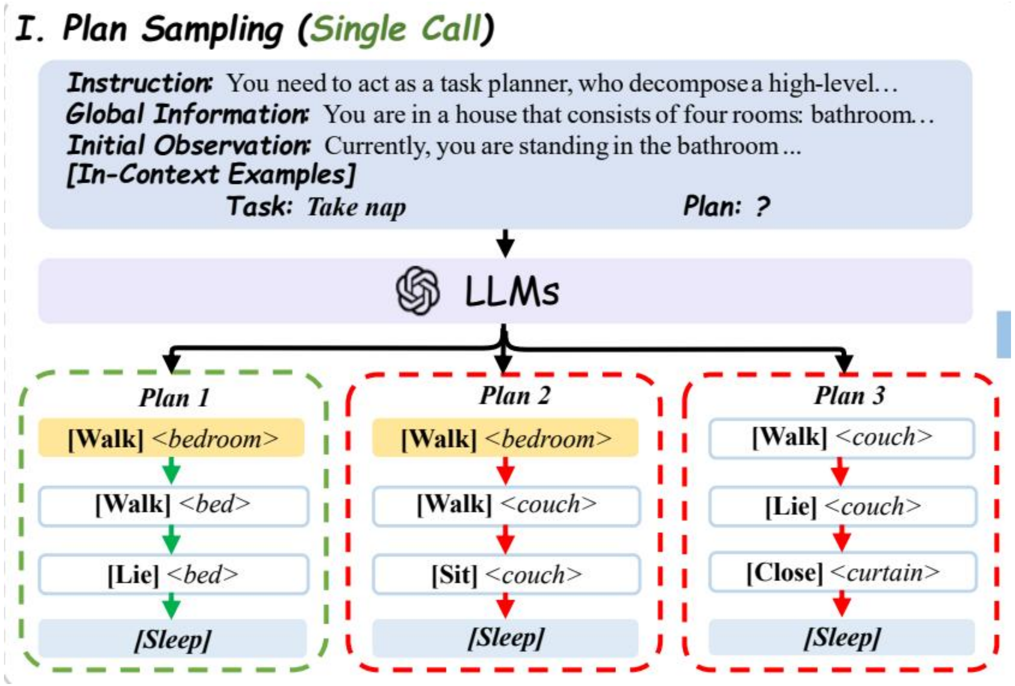
Methodology

Overview



I. Plan Sampling (Single Call)
Instruction: You need to act as a task planner, who decompose a high-level…
Global Information: You are in a house that consists of four rooms: bathroom…
Initial Observation: Currently, you are standing in the bathroom …
[In-Context Examples]
Task: Take nap
Plan: ?
LLMs
Plan 1
[Walk] <bedroom>
[Walk] <bed>
[Lie] <bed>
[Sleep]
Plan 2
[Walk] <bedroom>
[Walk] <couch>
[Sit] <couch>
[Sleep]
Plan 3
[Walk] <couch>
[Lie] <couch>
[Close] <curtain>
[Sleep]
II. Action Tree Construction
Take nap
t_0
[Walk] <bedroom>
[Walk] <bedroom>
[Walk] <couch>
t_1
[Walk] <bed>
[Walk] <couch>
[Lie] <couch>
t_2
[Lie] <bed>
[Sit] <couch>
[Close] <curtain>
t_3
[Sleep]
[Sleep]
[Sleep]
t_4
III. Grounded Deciding (Multiple Calls)
Instruction: At each moment, I will provide you with observations of your current environment, as well as the high-level task I want you to do, and previous mid-level sub-tasks that have been executed. You need to select the best sub-task from the options I provide to complete the designated home task based on the observation…
Observation (t_1): Currently, you are standing in the bedroom, … bed is close to character,…
[History]
Task: Take nap
Decision Request (t_1): Among the following actions, which action should you take.
A. [Walk] <bedroom> B. [Walk] <couch> C. …
[Instruction]
[Observation] (t_2)
History: You have executed the following sub-tasks:
[Walk] <bedroom>
Decision Request(t_2):
Among the following actions, which action should you take.
A. [Find] <bed>
B. [Find] <couch>
LLMs
✓ A
Observation
B ✗
Error Information
[Instruction]
[Observation] (t_1)
Error Info:
The previously choosed action "[Walk] <couch>" caused an error "<couch>" not in the environment
Decision Request (t_1):
A corrective choice of sub-task is:
A. [Walk] <bedroom>
B. ….

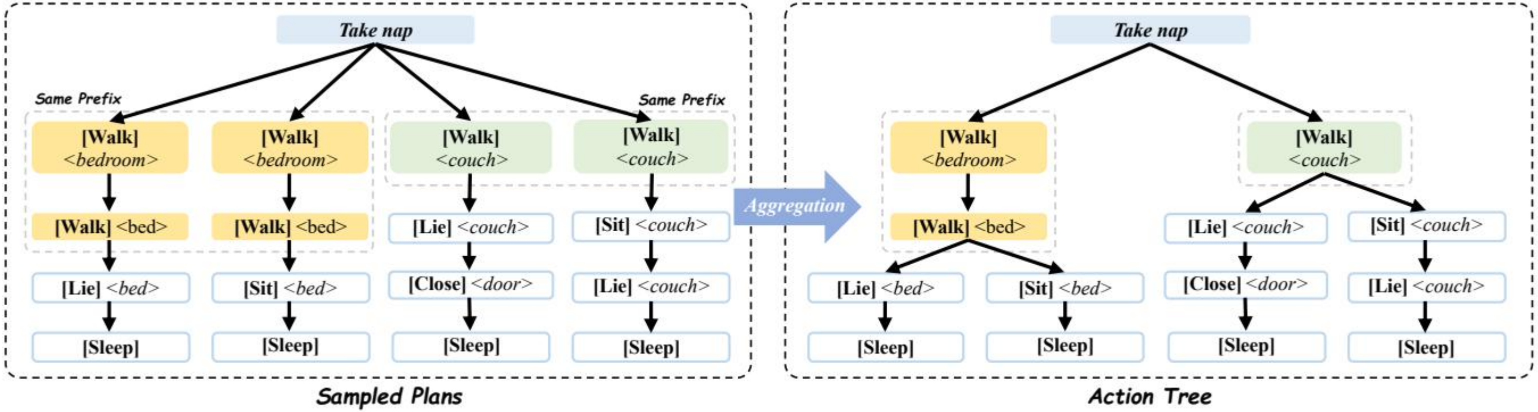
Plan Sampling



LLMs trained on large-scale data encode commonsense knowledge about the real-world.

The sampled plans serve as prior knowledge for task planning

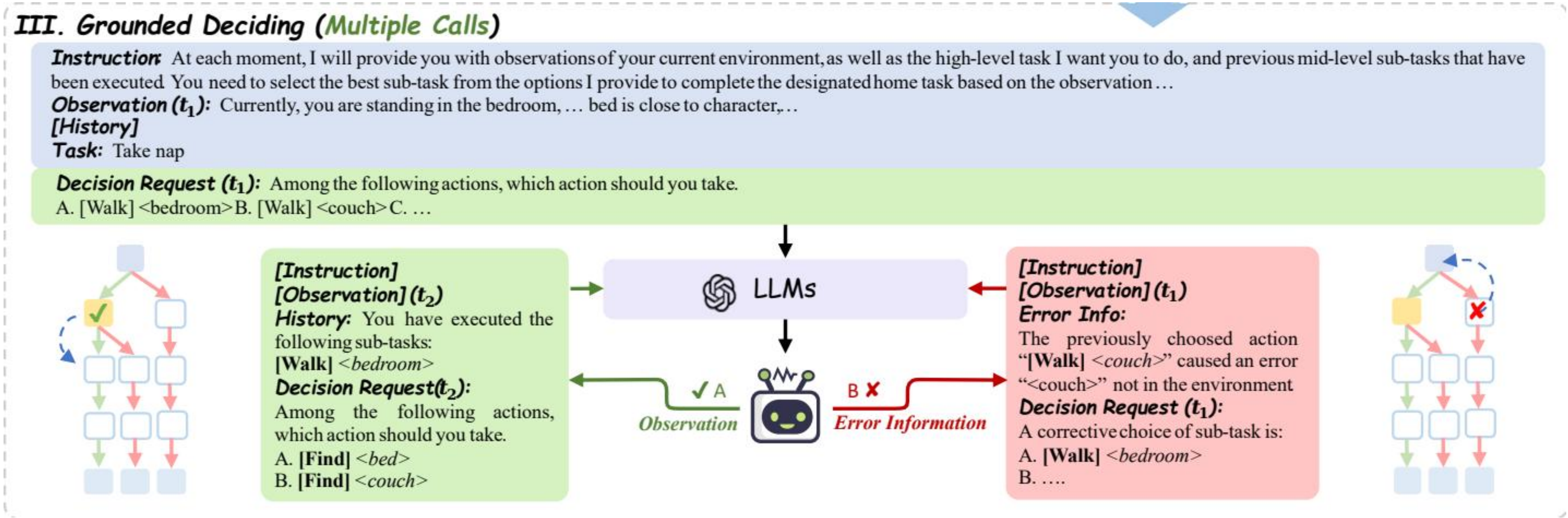
Action Tree Construction



When two plans share a common prefix but differ in their actions at a specific time step, their shared prefix is aggregated into a single branch.

Benefits of Action Tree: Converting the filtering of the plan level into a search at the action level, thereby reducing the execution time in the environment.

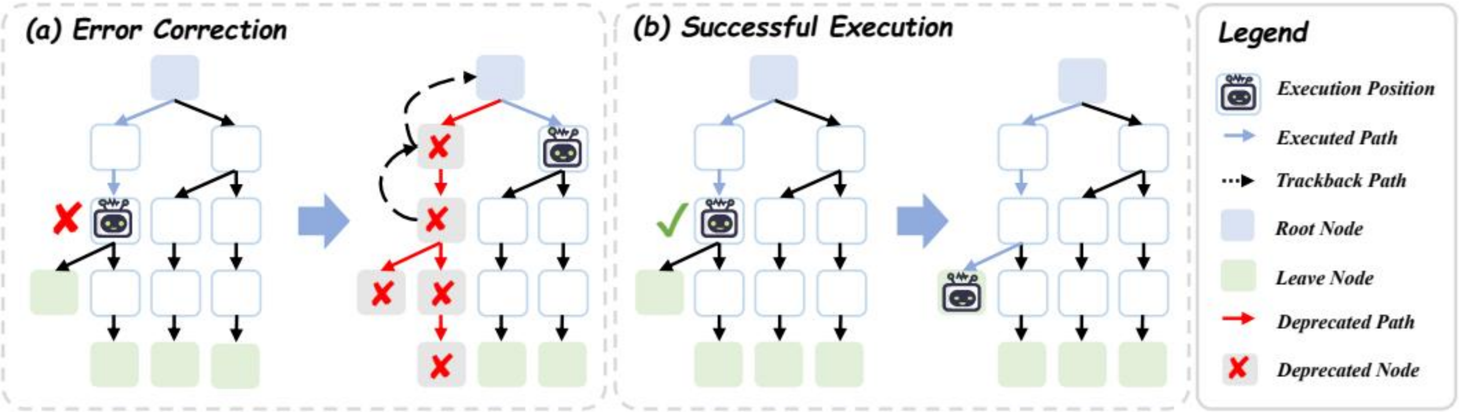
Grounded Deciding



During grounded deciding, an LLM functions as the policy $\pi(a_t \mid g, h_t, o_t)$.

This process simulates the decision making process of humans, who first propose several action options and then combine their current real-world observations to make decisions.

Grounded Deciding – Error Correction



Left: When an error occurs, the agent tracks back and marks the nodes along the way as invalid. Afterward, it makes a new decision at the previous fork node.

Right: After the action is successfully executed, the agent makes a decision at the current node moves on to the next level.

Experiment

Environment - VirtualHome

Task : Watch TV

Program:

[Walk] <television> (1)

[SwitchOn] <television> (1)

[Walk] <sofa> (1)

[Find] <controller> (1)

[Grab] <controller> (1)

Experimental Setup

Evaluate Metrics:

- Success Rate (SR) : SR is the fraction of executions that achieved all task-relevant goal-conditions.
- Goal Conditions Recall (GCR) : the set difference between ground truth final state conditions g and the final state achieved g' with the generated plan, divided by the number of task-specific goal-conditions;
- Executability (Exec.) : the fraction of actions in the plan that are executable in the environment, even if they are not relevant for the task.
- Cost: money spent to perform experiments.
- Number of Error Correction (No.EC)

LLM Backbone: Text-davinci-003

Baseline Models:

- Zero-Shot Planner[1]: Iterative Planner without grounding (No observation at each timestep).
- ProgPrompt[2]: Open-loop Planner.
- Iterative Planner: Iterative Planner with grounding.

[1] Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. 2022 ICML
[2] Singh I, Blukis V, Mousavian A, et al. Progprompt: Generating situated robot task plans using large language models[C]//2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023: 11523-11530.

Experimental Results

SOTA on **success rate**.

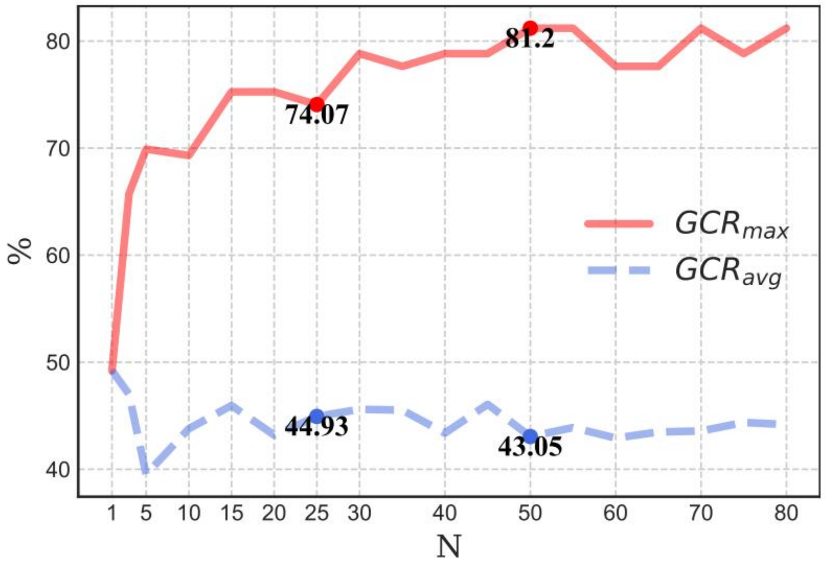
Token consumption is reduced by 92.2% compared to the previously best-performing model.

40.5% decrease in **error corrections**.

	EXEC. ↑	SR ↑	GCR ↑	$COST ↓	NO.EC ↓
w/o correction					
ZERO-SHOT PLANNER	16.49±3.08	1.07±0.76	1.52±0.75	1.36±0.09	N/A
PROGPROMPT	35.04±3.98	12.54±2.20	19.99±2.83	**1.25**±0.55	N/A
ITERATIVE-PLANNER	44.54±6.09	27.04±4.65	33.25±5.32	5.12±0.14	N/A
TREE-PLANNER$_{N=25}$	**55.74**±0.92	**28.33**±1.18	**39.96**±0.16	2.39±0.44	N/A
TREE-PLANNER$_{N=50}$	49.01±5.67	28.14±2.45	35.84±4.20	3.48±0.04	N/A
with correction					
LOCAL REPLAN	79.66±2.33	37.46±1.71	51.9±0.15	12.88±0.17	3.29±0.46
GLOBAL REPLAN	82.09±1.32	37.93±1.22	52.46±0.86	42.55±0.09	3.43±0.15
TREE-PLANNER$_{N=25}$	**89.13**±0.17	35.30±1.78	56.65±1.09	**3.30**±0.01	**1.85**±0.05
TREE-PLANNER$_{N=50}$	88.26±2.47	**41.58**±3.20	**59.55**±3.20	4.54±0.16	2.04±0.26

N represents the number of sampled plans.

Analysis – The upper limit of Plan Sampling



Maximum and average GCR for all sampled plans.
The x-axis represents the chosen N for plan sampling.

Conclusions:

1. $GCR_{\{max\}}$ being 81.2% indicates that plan sampling is effective.
2. As N increases, there is a noticeable increase in $GCR_{\{max\}}$, but it eventually reaches a threshold.
3. $GCR_{\{avg\}}$ does not consistently increase with an increased N. This implies that as N becomes larger, the proportion of “correct” plans to sampled plans may not necessarily increase.

Analysis – The effectiveness of Grounded Deciding

	EXEC.	SR	GCR
w/o correction			
TREE-PLANNER$_{N=25}$	55.74	28.33	38.96
† with oracle ↑	7.16	9.84	8.5
TREE-PLANNER$_{N=50}$	49.01	28.14	35.84
† with oracle ↑	3.41	6.54	4.78
with correction			
TREE-PLANNER$_{N=25}$	89.13	35.3	56.65
† with oracle ↑	8.45	26.8	19.76
TREE-PLANNER$_{N=50}$	88.26	41.58	59.55
† with oracle ↑	6.9	10.57	7.47

† represents the performance improvement after adding a gold plan to action tree construction.

Conclusions:

1. After incorporating the gold plan, there was a significant improvement in performance.
2. The improvement in performance for Tree-Planner$_{N=25}$ was greater than that for Tree-Planner$_{N=50}$.

Thanks