PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Vote-Tree-Planner: Optimizing Execution Order in LLM-based Task Planning Pipeline via Voting

Chaoyuan Zhang*
University of Washington
cz86@uw.edu

Zhaowei Li*
University of Washington
lzw365@uw.edu

Seth Z. Zhao
UCLA
sethzhao506@g.ucla.edu

Wentao Yuan
University of Washington
wentaoy@cs.washington.edu

* indicates equal contribution

Motivation

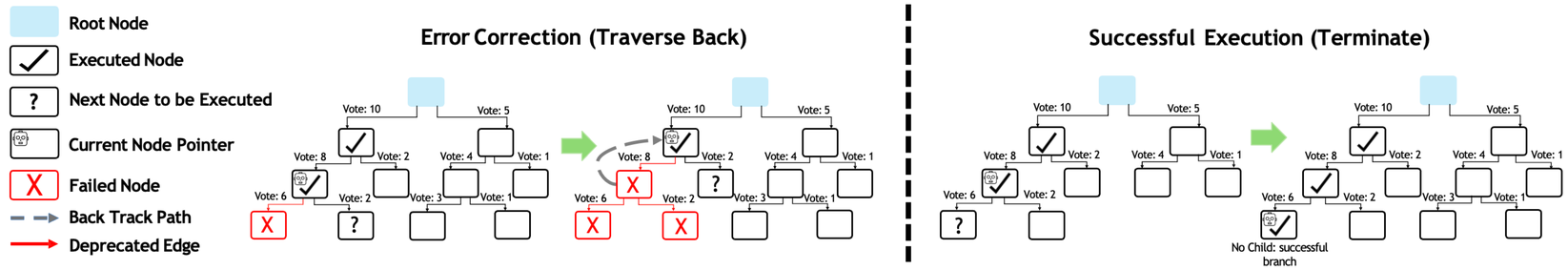
- The voting method enhances planning efficiency by prioritizing relevant commands and improves the task execution efficiency
- Compared with previous methods, the voting method integrates information from multiple plans and reduces the number of queries during execution
- We introduce the Vote-Tree-Planner that integrates tree and voting method to enhance the overall performance

Overview

- Vote-Tree-Planner consolidates multiple planning outputs into a cohesive tree-based structure.
- Votes indicate the relevance and frequency of commands across multiple plans, guiding the decision-making process.
- Vote-Tree-Planner can serve as a foundational approach for enhancing LLM-based robotic task planning and decision-making, improving both the efficiency and reliability.

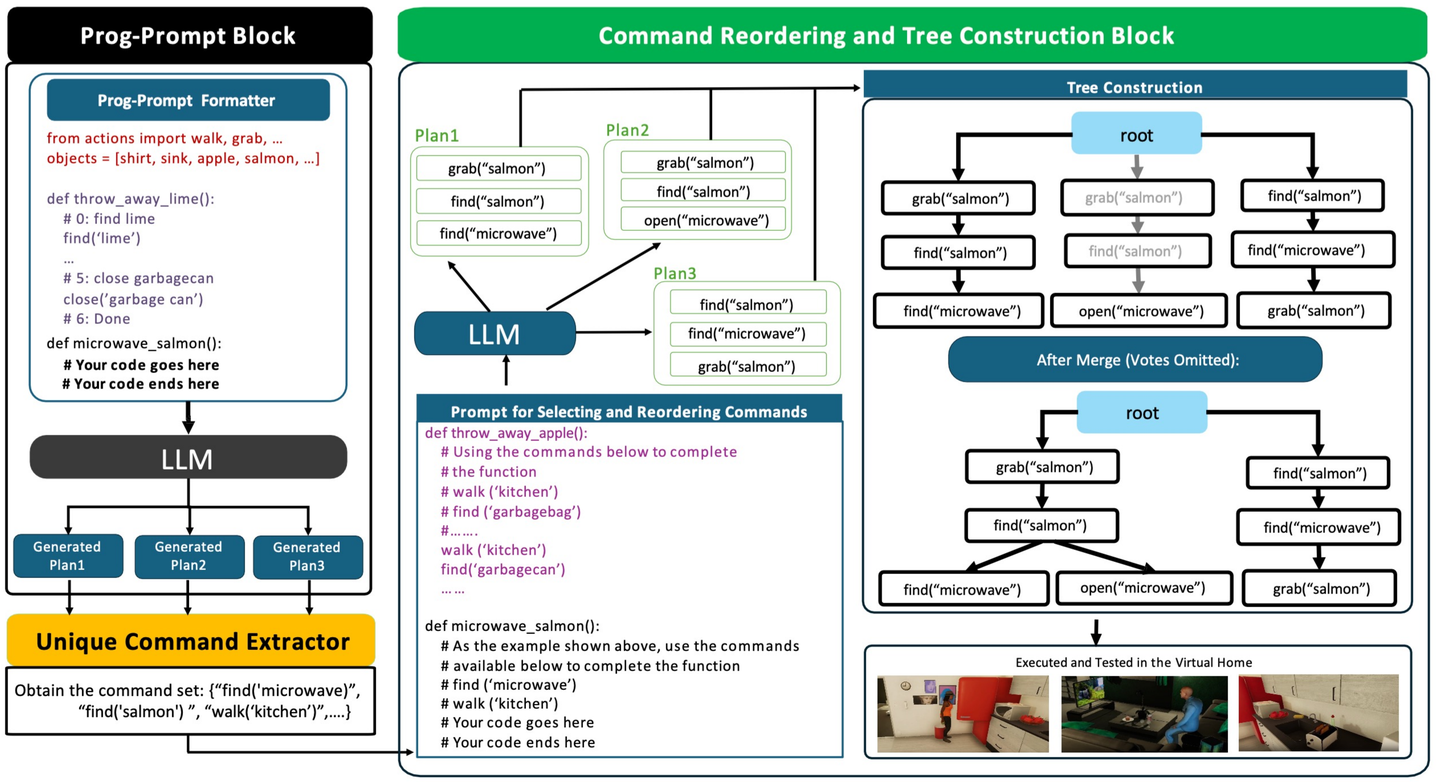
Order of Execution With Error Correction

Vote-Tree-Planner can execute the tree with or without correction. When a node is executed successfully, the next step is to find and attempt to execute its highest-voted child node. If execution fails, the system tries the next highest-voted child, continuing until all options are exhausted before moving to another branch. The process terminates upon successfully executing a node without any child nodes or there is no more nodes to execute. If the execution is without correction, the system does not revert to previous nodes.



Entire Pipeline of Our Proposed Method

Vote-Tree-Planner uses Prog-Prompt framework to create multiple plans initially, then unique commands are extracted. Then reordering these unique commands into new plans, so that LLM only choose from a narrowed set of commands. These plans are then merged into an execution tree, which is executed based on votes of nodes.



Experimental Results

Experiment results without error correction.

Methods	SR ↑	GCR ↑	Exec ↑
Zero-Shot Planner [9]	0.01±0.01	0.02±0.01	0.16±0.03
Prog-Prompt [16]	0.34±0.06	0.67±0.07	**0.92**±0.02
Tree-Planner [8]	0.28±0.02	0.40±0.01	0.55±0.01
Vote-Tree-Planner (Ours)	**0.43**±0.04	**0.70**±0.04	0.89±0.02

Vote-Tree-Planner outperforms the SOTA by 10% in SR, with higher GCR and similar executability, indicating its ability to complete more sub-goals and generate more executable plans.

Vote-Tree-Planner improves SR and GCR over baselines. The Exec is slightly lower than that of Prog-Prompt due to Prog-Prompt's more repeated, but mostly executable plans.

Experiment results with error correction.

Methods	SR ↑	GCR ↑	Exec ↑
Iterative-Planner [8](Global)	0.37±0.02	0.52±0.01	0.82±0.02
Prog-Prompt [16]	0.38±0.07	0.66±0.08	**0.93**±0.04
Tree-Planner [8]	0.41±0.03	0.60±0.03	0.88±0.03
Vote-Tree-Planner (Ours)	**0.48**±0.07	**0.81**±0.06	0.90±0.04

Experiment results (without error correction) of different node selection methods in Vote-Tree-Planner.

Node Selection Method	SR ↑	GCR ↑	Exec ↑
Randomly Selected Node	0.33±0.08	0.62±0.05	0.83±0.04
Maximum Voted Node	**0.43**±0.04	**0.70**±0.04	**0.89**±0.02

Vote-Tree-Planner shows the voting method largely improves performance by guiding correct executions, comparing randomly selecting a child node and starting with the child node with the highest vote (w/o correction). This proves the effectiveness of the voting mechanism, as well as the unique command extractor, since the performance from randomly selecting a child node is still comparable to the result of baseline models (w/o correction).

Discussions on Other Related Aspects

- Qualitative analysis demonstrates that our method reduces redundancies and repeated commands significantly.
- Refining LLM integration in the correction phase for better adaptability, improving token efficiency, and comparing the planner with different LLM backbones are future directions.