Fast and Accurate Task Planning Using Neuro-Symbolic Language Models and Multi-level Goal Decomposition

Minseo Kwon, Yaesol Kim, and Young J. Kim

Presented By:
Abhinandan Vellanki (abhinanv)
3/18/26

Outline

- Background
- Problem Formulation
- Method
- Experiments
- Conclusion

Background

Robot Task Planning

Symbolic Planning :

- Planning Domain Definition Language (PDDL) effectively plans for different problems in different domains
- Hierarchical Planning + Task and Motion Planning (TAMP).
- Complexity is PSPACE-hard, i.e., computationally intractable and unscalable.

Large Language Models (LLMs):

- Shorter inference time, generalizability, possess ‘common sense knowledge’ from large scale data.
- Can be used as policy model (L-Policy) or world model (L-Model).
- Can generate robot action primitives (as Python code); infer physical constraints and spatial relationships.
- Suffer from token inefficiencies that are exposed over complex / long tasks; can hallucinate. Limited multi-step reasoning and weak failure recovery.

Background

Related Work

Hybrid Task Planning:

- LLM translates natural language into PDDL initial states and goals and could even solve simple problems.
- LLMs can be combined with vision models for scene understanding; they can inform heuristic planners.
- Yet to be tested on large-scale problems.

Tree Search + LLMs:

- Can be combined with Monte-Carlo Tree Search (MCTS) for iterative state-transitions, with LLMs employed as both L-Policy and L-Model to solve large-scale Partially-Observed Markov Decision Processes (POMDPs).

This Work:

- Builds on MCTS+LLM, but introduces **Goal Decomposition** into sub-problems that reduce the search space.
- Solves problems with symbolic planner or MCTS+LLM based on **Minimum Description Length (MDL).**

Problem Formulation

$$P \equiv (S, O, A, T, s_o, S^*)$$

S : Set of fully observable states

O : Objects in the environment

A : Set of possible actions

T : Deterministic State Transition Model [$T(S, A) = S'$]

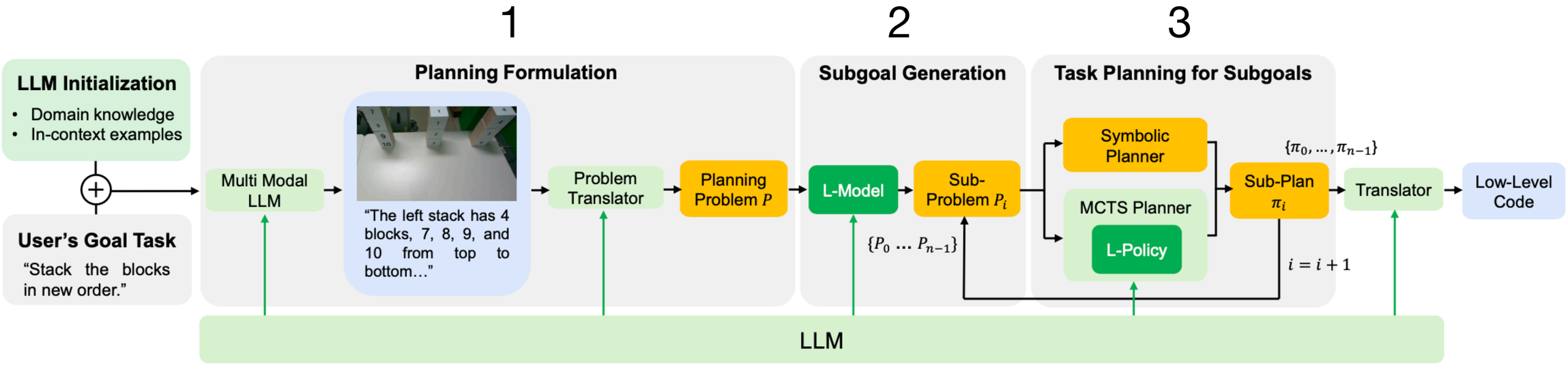
S* : Set of goal states [S* ⊂ S]

s_0 : Initial state

Goal: Find policy $\pi = \{a_1, \ldots, a_n \mid \forall a_i \in A\}$ to go from s_0 to $\exists s_n \in S^*$ in finite steps.

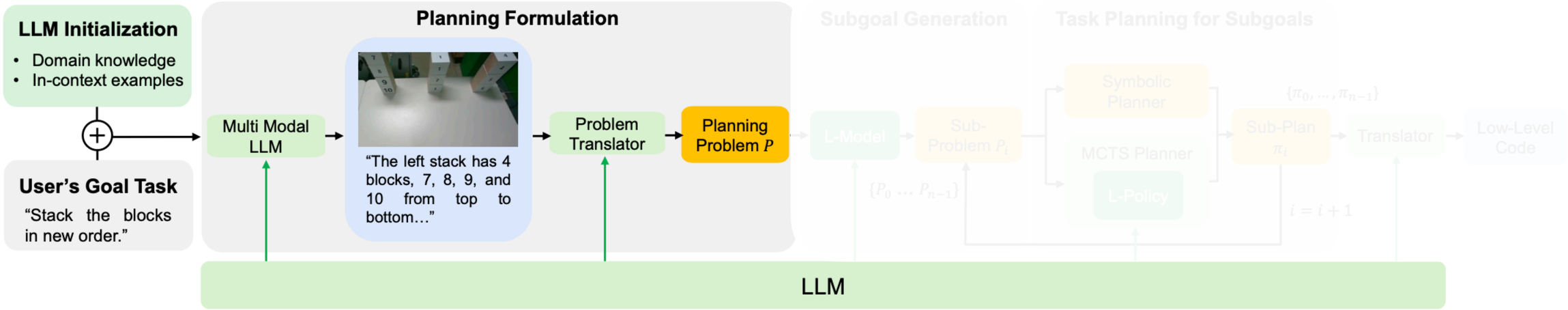
Method

Task Planning Pipeline



Method

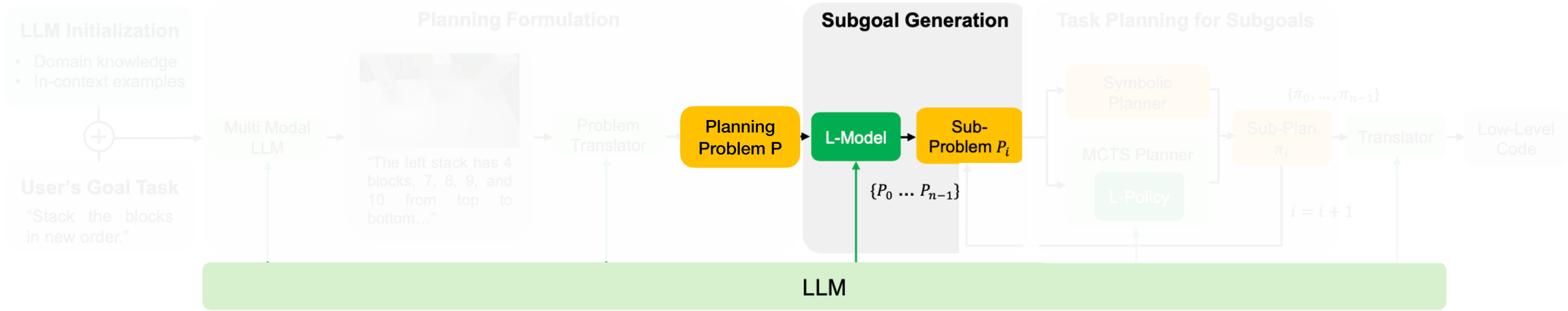
Planning Formulation Step



- GPT-4o : Image + Prompt to describe the problem (e.g.: 'What objects are on the table?).
- Domain PDDL, in-context example, user goal and scene description are used by LLM to generate problem PDDL consisting of O, s_0, S*, that together specify P.
- 2D open-vocabulary object detection model used to estimate bounding boxes that are later used for grasp planning.

Method

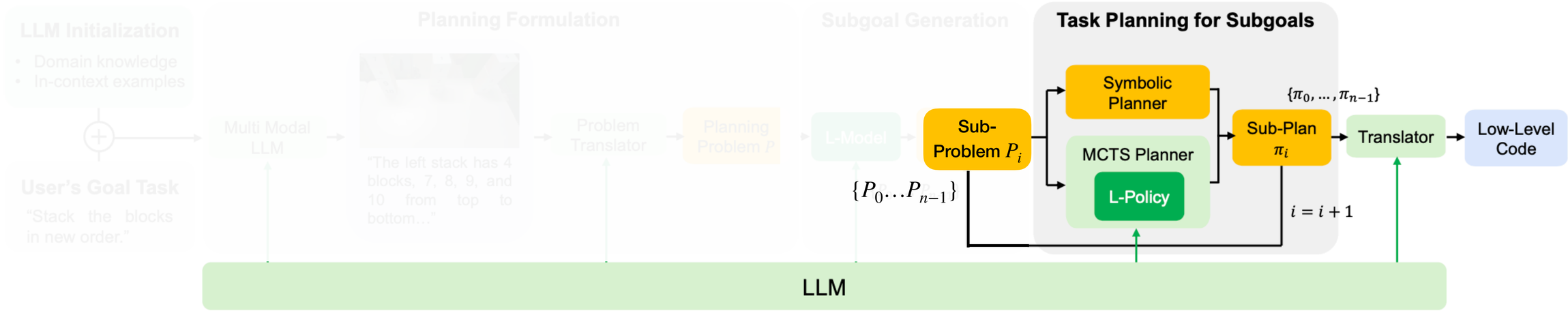
Subgoal Generation



- Break goal down into subgoals: $G = \{S_0^*, S_1^*, \ldots, S_n^*\}$ iff S_i^* is reachable from S_{i-1}^* for $1 \leq \forall i \leq n$ via a finite number of state transitions from $\exists s_{i-1} \in S_{i-1}^*$ to $\exists s_i \in S_i^*$ and $S_o^* = \{s_o\}, S_n^* = S^*$.
- Decompose the problem P into n smaller problems $0 \leq \forall i \leq n-1, P_i \equiv (S, O, A, T, s_i, S_{i+1}^*)$
- LLM prompt is domain knowledge and a one-shot planning example along with explanation of the steps and ask it to generate G based on how the example problem is solved

Method

Task Planning For Subgoals



- Find a policy $\pi_i \subset \pi$ for each sub-planning problem P_i.
- $\pi(s_i) = a_i, s_{i+1} = T(s_i, a_i)$. If $s_{i+1} \in S^*_{i+1}$, π_i is a valid policy.
- s_{i+1} is the initial state for P_{i+1}
- Final Policy $\pi = \bigcup_i \pi_i$ presented as a PDDL plan which is then translated into low-level code with an LLM.

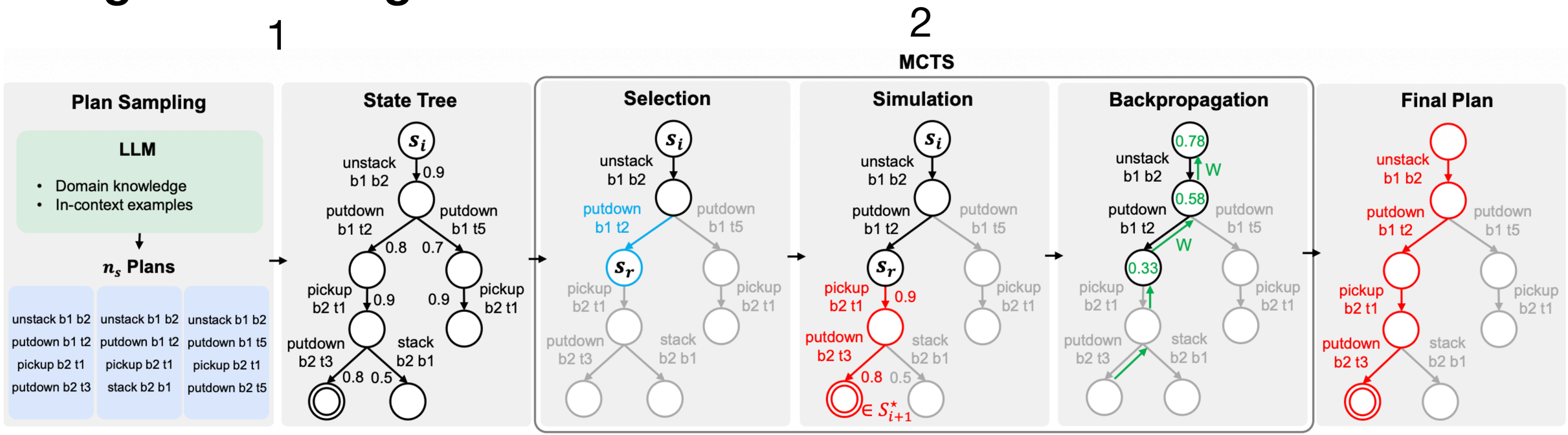
Method

Subgoal Planning

- Choose approach based on MDL
- MDL is $|P_i|$, which is hard to estimate.
- Empirically measure the planning time spent by running the LLM+MCTS planner or symbolic planner for P_i.
- If high, use LLM+MCTS. If low, use symbolic planner.
- Authors chose Fast Downward (FD) Planner for sym, which builds graphs from PDDL problems and solves them with heuristic search.

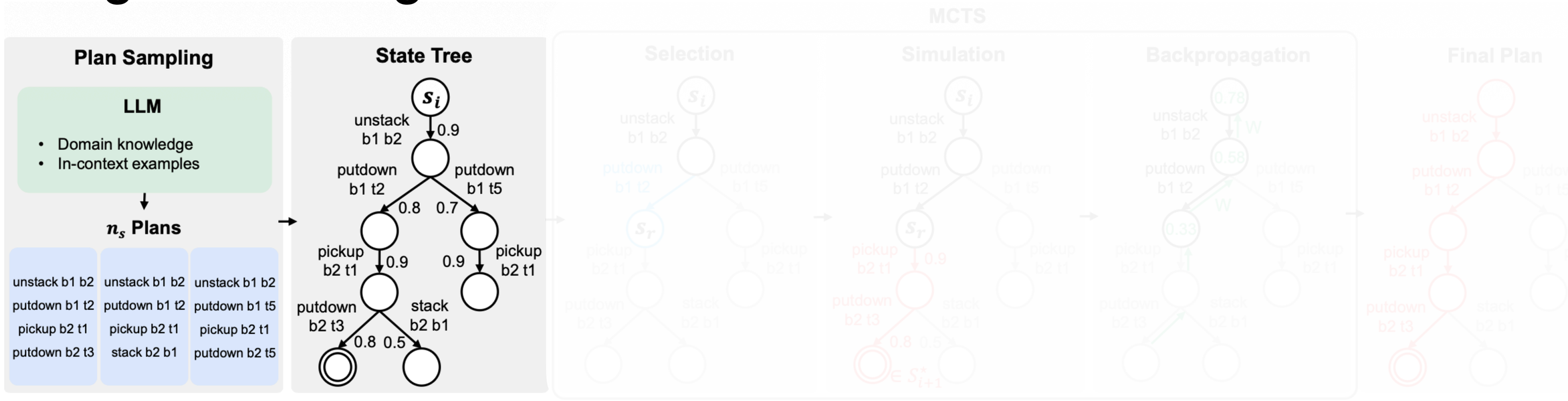
Method

Subgoal Planning: MCTS+LLM Planner



Method

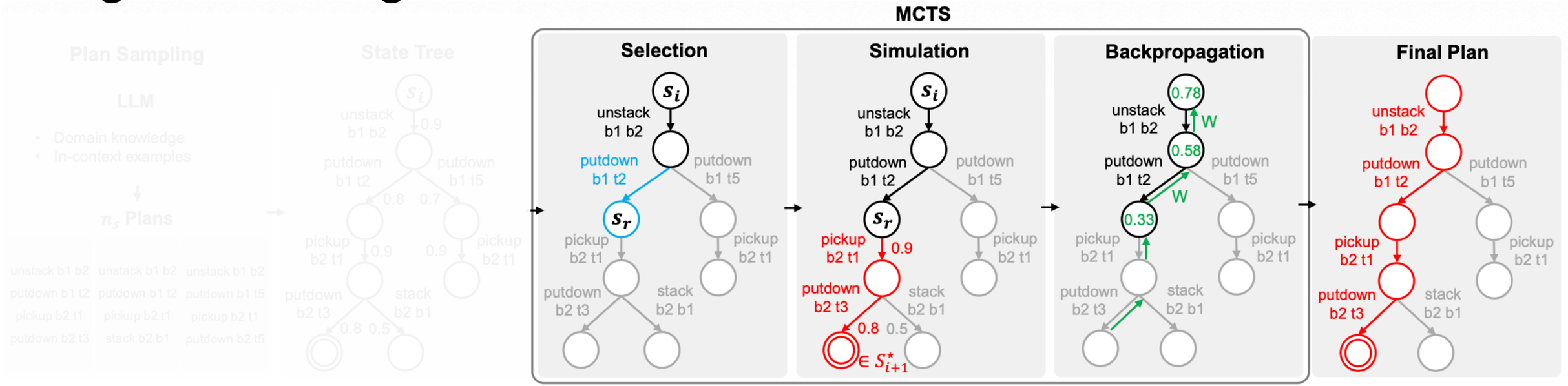
Subgoal Planning: MCTS+LLM Planner



- Authors hypothesize that sampling for a subgoal P_i rather than the entire problem P should 'lead to presumably higher accuracy'.
- T_i generated for a P_i from all the n_s plans, with nodes representing states s and edges corresponding to actions $a \in A$, connecting s_i and s_{i+1} if $s_{i+1} = T(s_i, a_i)$.
- Action weight is computed as sum of token log probabilities for each LLM-generated action.
- If preconditions(a) hold for s, then a is valid and added to T_i. If not, subsequent actions are removed.

Method

Subgoal Planning: MCTS+LLM Planner



- Search already expanded T_i to find π_i using estimated reward per node. Even rollouts are within T_i.
- Recursively select child with highest Upper Confidence Bound (UCB) score from {visited nodes} till a node with unvisited children is reached.
- Randomly select one of the children, s_r. If $s_r \in S^*$ return 1 as the reward. Else, estimate reward through simulation:
  - Choose child with highest action weight till lead node s^* is reached. If $s^* \in S^*$, return reward scaled down by distance d to s_r : $\frac{1}{1+d}$, else return 0.
- Backpropagate the reward by incrementing visit counts and adding the reward. Final plan chains max reward nodes.

Experiments

Different Planners Compared

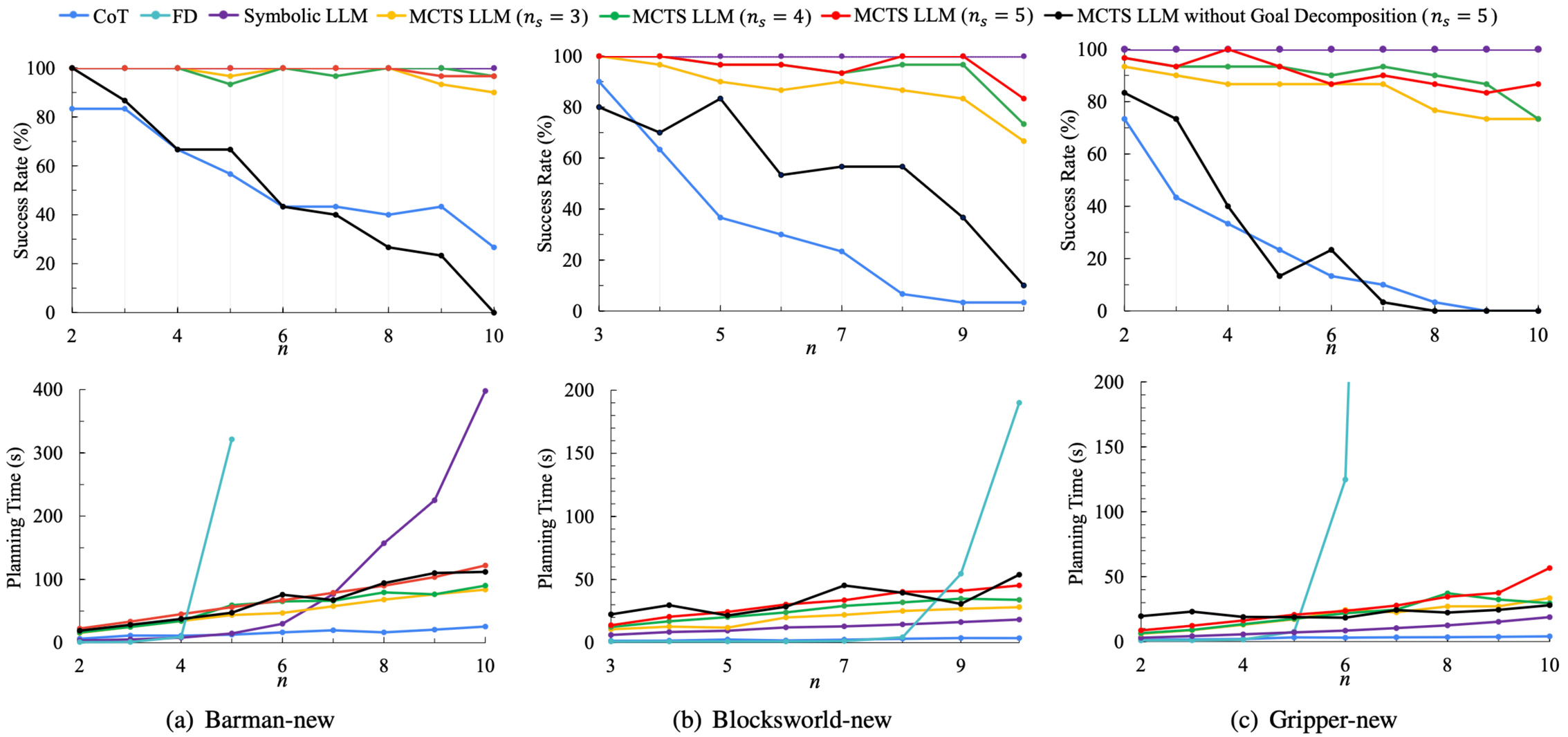
1. CoT Planner: baseline LLM planner with chain-of-thought reasoning.
2. Fast Downward (FD) planner: baseline symbolic planner (full problem)
3. Symbolic LLM Planner: FD as subgoal planner
4. MCTS LLM Planner: MCTS as subgoal planner

Domains Used

1. Barman-new: dual manipulators making $2 \leq n \leq 10$ cocktails, each poured into a shot glass. 3 ingredients, n+1 shot glasses.
2. Blocksworld-new: single arm stacking $3 \leq n \leq 10$ blocks into 3 stacks. Only six intermediate placement positions available.
3. Gripper-new: four robots move $2 \leq n \leq 10$ balls to four different rooms from their initial location. Multi-agent scenario.

Experiments

- 30 Problems per domain
- n_s is the number of sampled plans

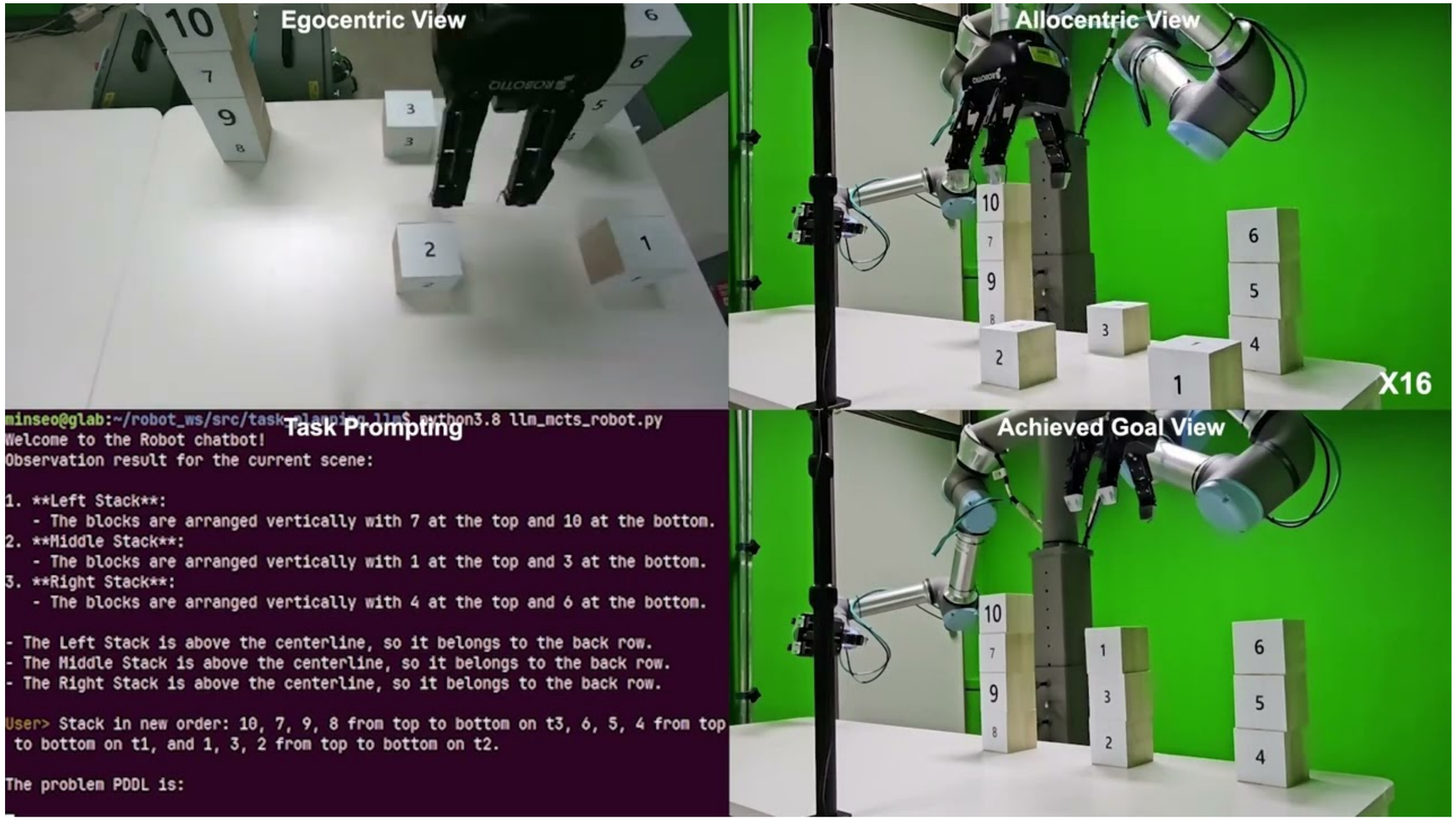


(a) Barman-new (b) Blocksworld-new (c) Gripper-new

Experiments

Robot Demonstrations

https://graphics.ewha.ac.kr/LLMTAMP/



Conclusion

Contributions

1. Neuro-symbolic pipeline that utilizes LLMs as both L-Model and L-Policy.
2. Goal decomposition as an effective way to solve complex problems.
3. Significantly lower planning time than baseline symbolic planners.

Future Work

1. Auto-select level of goal decomposition.
2. Explicit integration with TAMP.
3. Ensuring generalizability of goal decomposition.
4. Expand to diverse environments and multi-agent systems.