

# **Agent Planning Ability**

---

Tianjiao Yu, Siddarth Madala, Risham Sidhu

# Agenda

- LLM+P: Empowering Large Language Models with Optimal Planning Proficiency (Required)
- AutoGPT+P: Affordance-based Task Planning using Large Language Models
- Language Agent Tree Search Unifies Reasoning Acting and Planning in Language Models (Required)
- TravelPlanner: A Benchmark for Real-World Planning with Language Agents

# **LLM+P: Empowering Large Language Models with Optimal Planning Proficiency**

---

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, Peter Stone

2023

# LLM+P Intro & Background

It has always been easy to generate prompts that expose Model's lack of true "understanding"

- Trained to generate sequences of words that might be plausible. Not to demonstrate understanding.
- The internet is awash with 5-year-old examples that ChatGPT fails to solve.

# LLM+P Intro & Background

## A Failure Example of GPT-4 in Planning

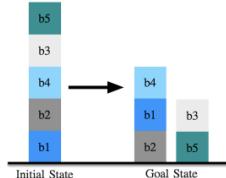
**Problem (P1):** You have 5 blocks. One cannot place more than one block on another block. b5 is on top of b3. b4 is on top of b2. b2 is on top of b1. b3 is on top of b4. b1 is on the table. b5 is clear. Your arm is empty.

Your goal is to move the blocks.

b1 should be on top of b2.

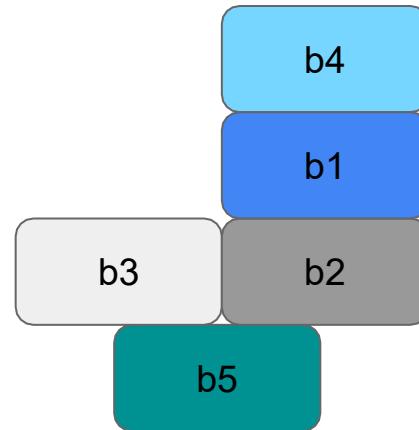
b3 should be on top of b5.

b4 should be on top of b1.



### GPT-4:

1. Move b5 from b3 to the table.
2. Move b3 from b4 to b5.
3. Move b4 from b2 to the table.
4. Move b2 from b1 to b5.
5. Move b1 from the table to b2.
6. Move b4 from the table to b1.



# LLM+P Intro & Background



You

What is the sum of the following numbers? \$27.02, \$8.20, \$19.28, \$2.29, \$0.82, \$17.71, \$13.75, \$5.99, \$97.50, \$56.69, \$82.00, \$91.09, \$5.07, \$1.25, \$2.24, \$7.67, \$85.90, \$91.29, \$29.21, \$7.29, \$8.18



ChatGPT

The sum of the given numbers is \$662.55.

The answer is **wrong**. It should be \$660.44

# LLM+P Intro & Background

Proficient at linguistic competence, but not at functional competence.

Good at **how** to say things, but not as good at **what** to say.

\*\*Does this mean we should include all arithmetic and planning problems in the training corpus? Why should it be necessary?

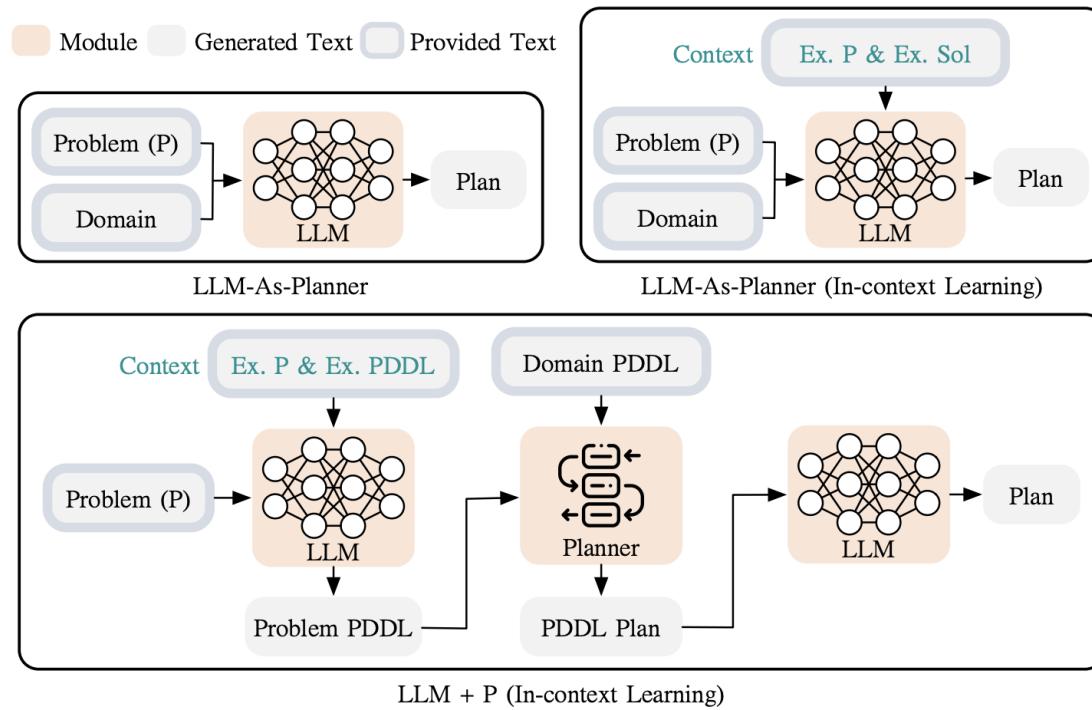
We already have calculators and symbolic planners, we should connect llm with these tools

## **LLM+P Method**

the paper introduce a *methodology that makes no changes to the LLM, LLM+P:*

- Outputs a problem description suitable as input to a general purpose planner (PDDL);
- Solves the problem using the general-purpose planner;
- Converts the output of the planner back to natural language (or connects to action executors of a robot).

# LLM+P Method



# LLM+P Method

## Predicates:

- |                        |   |
|------------------------|---|
| $\text{ROOM}(x)$       | – true iff $x$ is a room                                      |
| $\text{BALL}(x)$       | – true iff $x$ is a ball                                      |
| $\text{GRIPPER}(x)$    | – true iff $x$ is a gripper (robot arm)                       |
| $\text{at-roddy}(x)$   | – true iff $x$ is a room and the robot is in $x$              |
| $\text{at-ball}(x, y)$ | – true iff $x$ is a ball, $y$ is a room, and $x$ is in $y$    |
| $\text{free}(x)$       | – true iff $x$ is a gripper and $x$ does not hold a ball      |
| $\text{carry}(x, y)$   | – true iff $x$ is a gripper, $y$ is a ball, and $x$ holds $y$ |

## In PDDL:

```
(:predicates (ROOM ?x) (BALL ?x) (GRIPPER ?x)
             (at-roddy ?x) (at-ball ?x ?y)
             (free ?x) (carry ?x ?y))
```

# LLM+P Method

Formally, the input of a planning problem P is defined by a tuple  $\langle S, s^{init}, S^G, A, f \rangle$ :

- $S$  is a finite and discrete set of states used to describe the world's state.
- $s^{init} \in S$  is an initial world state.
- $S^G$  is a set of goal states.
- $A$  is a set of symbolic actions.
- $F$  is the underlying state transition function.  $F$  takes the current state and an action as input and outputs the corresponding next state.

# LLM+P Method

Re-writing planning prompts in the PDDL format

- This is essentially a machine translation task.
- And LLMs are good at describing and translating.
- However, the zero-shot example:
  - Missed the “B1 is on the table”
  - Made up Predicate “empty”

An Example PDDL Problem File Written by GPT-4

**Prompt:** Description of Problem (P1) + Provide me with the problem PDDL file that describes the planning problem directly without further explanations.

**GPT-4 (the generated problem PDDL):**

```
(:objects b1 b2 b3 b4 b5 - block)
(:init (on b5 b3) (on b4 b2) (on b2
b1) (on b3 b4) (clear b5) (empty))
(:goal (and (on b1 b2) (on b3 b5)
(on b4 b1)))
```

# LLM+P Method

## Context (shaded):

An example planning problem is: You have 5 blocks. b2 is on top of b5. b5 is on top of b1. b1 is on top of b4. b3 is on top of b2. b4 is on the table. b3 is clear. Your arm is empty. Your goal is to move the blocks. b4 should be on top of b3. The problem PDDL is:

```
(:objects b1 b2 b3 b4 b5)
(:init (arm-empty) (on b1 b4) (on
b2 b5) (on b3 b2)
    (on-table b4) (on b5 b1)
(clear b3))
(:goal (and (on b4 b3))))
```

**Prompt: Description of Problem (P1) + Context**  
+ Provide me with the problem PDDL file that describes the planning problem directly without further explanations.

## GPT-4 (the generated problem PDDL):

```
(:objects b1 b2 b3 b4 b5 )
(:init (arm-empty) (on-table b1)
(on b2 b1) (on b3 b4)
    (on b4 b2) (on b5 b3) (clear
b5)))
(:goal (and (on b1 b2) (on b3 b5)
(on b4 b1)))
```

## Planner:

```
(unstack b5 b3) (putdown b5)
(unstack b3 b4) (stack b3 b5)
(unstack b4 b2) (putdown b4)
(unstack b2 b1) (putdown b2)
(pickup b1) (stack b1 b2) (pickup
b4) (stack b4 b1)
```

# **LLM+P Method**

Assumptions made for LLM+P:

- The model knows when to trigger LLM+P based on its conversation with a user
- A domain PDDL is provided. Actions, predicates need to be predefined.
- Problem description in natural language and its corresponding problem PDDL file are also provided.

# LLM+P Method

Novelty:

- Previous classical planning systems need domain-specific programming languages
- LLM+P take advantage of both the LLM and the planner
  - Previous works using LLMs directly as planners either lack long-horizon reasoning ability or require prompting and fine-tuning
- WebGPT and ToolFormer are similar but this work shows PDDL can be another useful external module.

# LLM+P Experiment & Results

GPT4 with 0 temperature.

7 Planning domains, 20 automatically generated tasks in each domain:

- Blockworld
- Floor tile: use paint color patterns on floor tiles. Robots can move around and change colors but cannot step on painted tiles.
- Tyreworld: replace flat tires by inflating tires, tightening nut, etc.
- Others..

# LLM+P Experiment & Results

LLM-AS-P provides a plan in natural language for every problem, most of these plans are not feasible

Domain	Success Rate %				
	LLM <sup>-</sup>	LLM	LLM <sup>ToT</sup>	LLM+P <sup>-</sup>	LLM+P
BARMAN	0	0	0	0	<b>20 (100)</b>
BLOCKSWORLD	20	15 (30)	0 (5)	0	<b>90</b>
FLOORTILE	0	0	0	0	<b>0</b>
GRIPPERS	25 (60)	35 (50)	10 (20)	0	<b>95 (100)</b>
STORAGE	0	0 (25)	0	0	<b>85</b>
TERMES	0	0	0	0	<b>20</b>
TYREWORLD	5	15	0	0	<b>10 (90)</b>

# LLM+P Experiment & Results

LLM-AS-P fails in the same way with or without the example plan as context.

Domain	Success Rate %				
	LLM <sup>-</sup>	LLM	LLM <sup>ToT</sup>	LLM+P <sup>-</sup>	LLM+P
BARMAN	0	0	0	0	<b>20 (100)</b>
BLOCKSWORLD	20	15 (30)	0 (5)	0	<b>90</b>
FLOORTILE	0	0	0	0	<b>0</b>
GRIPPERS	25 (60)	35 (50)	10 (20)	0	<b>95 (100)</b>
STORAGE	0	0 (25)	0	0	<b>85</b>
TERMES	0	0	0	0	<b>20</b>
TYREWORLD	5	15	0	0	<b>10 (90)</b>

# LLM+P Experiment & Results

The hardest domains are the ones with complex spatial relationship. The LLM-AS-P methods (with or without context) completely fail at this type of problems.

Domain	Success Rate %				
	LLM <sup>-</sup>	LLM	LLM <sup>ToT</sup>	LLM+P <sup>-</sup>	LLM+P
BARMAN	0	0	0	0	<b>20 (100)</b>
BLOCKSWORLD	20	15 (30)	0 (5)	0	<b>90</b>
FLOORTILE	0	0	0	0	<b>0</b>
GRIPPERS	25 (60)	35 (50)	10 (20)	0	<b>95 (100)</b>
STORAGE	0	0 (25)	0	0	<b>85</b>
TERMES	0	0	0	0	<b>20</b>
TYREWORLD	5	15	0	0	<b>10 (90)</b>

# LLM+P Experiment & Results

- LLM+P produces optimal plans.

Domain	Success Rate %				
	LLM <sup>-</sup>	LLM	LLM <sup>ToT</sup>	LLM+P <sup>-</sup>	LLM+P
BARMAN	0	0	0	0	<b>20 (100)</b>
BLOCKSWORLD	20	15 (30)	0 (5)	0	<b>90</b>
FLOORTILE	0	0	0	0	<b>0</b>
GRIPPERS	25 (60)	35 (50)	10 (20)	0	<b>95 (100)</b>
STORAGE	0	0 (25)	0	0	<b>85</b>
TERMES	0	0	0	0	<b>20</b>
TYREWORLD	5	15	0	0	<b>10 (90)</b>

# LLM+P Discussion

- LLM can give reasonable rankings on the plans, but it often fails to recognize whether the plan reaches the goal.
- Times out in most cases
- Context is important for LLM+P to work.

Domain	Success Rate %				
	LLM <sup>-</sup>	LLM	LLM <sup>ToT</sup>	LLM+P <sup>-</sup>	LLM+P
BARMAN	0	0	0	0	<b>20 (100)</b>
BLOCKSWORLD	20	15 (30)	0 (5)	0	<b>90</b>
FLOORTILE	0	0	0	0	<b>0</b>
GRIPPERS	25 (60)	35 (50)	10 (20)	0	<b>95 (100)</b>
STORAGE	0	0 (25)	0	0	<b>85</b>
TERMES	0	0	0	0	<b>20</b>
TYREWORLD	5	15	0	0	<b>10 (90)</b>

# **AutoGPT+P: Affordance-based Task Planning with Large Language Models**

---

Timo Birr, Christoph Pohl, Abdelrahman Younes, Tamim Asfour

2024

# AutoGPT+P Intro

Limitation of LLM+P:

- the system is restricted by the closed-world assumption of classical planners
- Need definitions of all the objects, actions, etc.
- No automated error correction.

# AutoGPT+P Intro

## AutoGPT

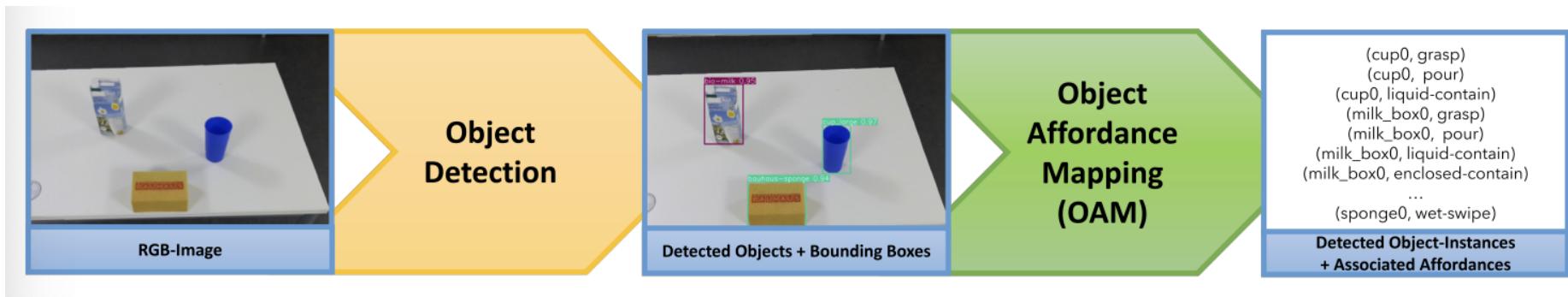
- Consider alternative tools/objects suitable for the task (base on similar affordance)
- Extend LLM+P with automated error correction and dynamic planning.
- Also allow robot to seek assistance from humans when the task surpass agent's capabilities.

# AutoGPT+P Method

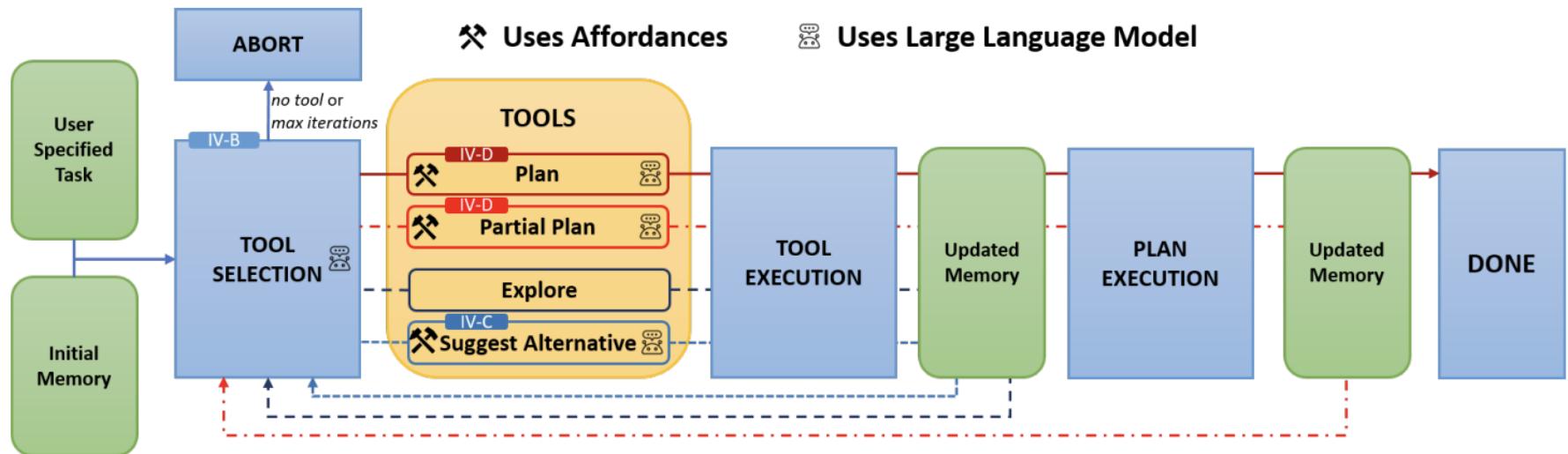
Affordance-based Scene Representation:

- Affordance provide information about possible actions
- Affordance provide functionality info, which can be used for object substitution
- Object affordance mapping using ChatGPT

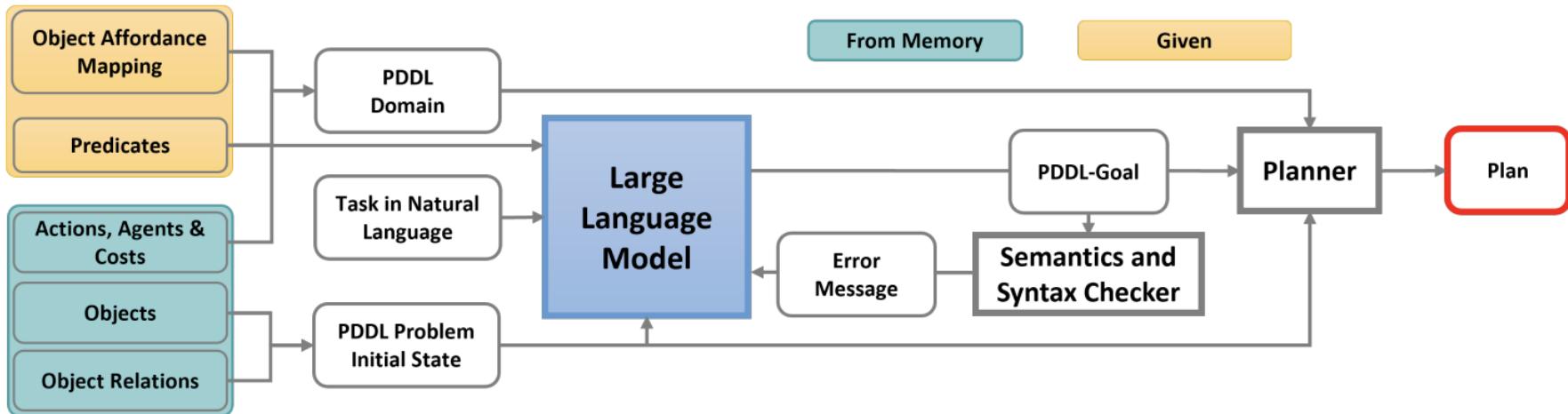
# AutoGPT+P Method



# AutoGPT+P Method



# AutoGPT+P Method



# AutoGPT+P Results

Instruction Family	<i>SayCan</i> (plan)	<i>GPT-3</i> As Planner	<i>GPT-3</i> As Planner+A	<i>GPT-4</i> As Planner	<i>GPT-4</i> As Planner+A	<i>GPT-3</i> Ours	<i>GPT-3</i> Ours (Auto)	<i>GPT-4</i> Ours	<i>GPT-4</i> Ours (Auto)
NL Primitive	0.93	0.47	0.53	0.93	0.93	0.73	0.73	<b>1.00</b>	<b>1.00</b>
NL Verb	0.60	0.00	0.00	0.67	0.87	0.27	0.33	0.93	<b>1.00</b>
NL Noun	0.93	0.13	0.07	0.26	0.20	0.27	0.40	0.93	<b>1.00</b>
Structured	<b>0.93</b>	0.20	0.13	0.87	0.60	0.00	0.13	0.20	<b>0.93</b>
Embodiment	0.64	0.09	0.00	0.55	0.55	0.64	0.64	0.82	<b>1.00</b>
Crowd-Sourced	0.73	0.13	0.07	<b>0.93</b>	0.73	0.27	0.33	0.73	<b>0.93</b>
Long-Horizon	0.73	0.00	0.00	0.40	0.33	0.20	0.33	0.80	<b>1.00</b>
Drawer	<b>1.00</b>	0.00	0.00	0.00	0.00	0.66	0.33	<b>1.00</b>	<b>1.00</b>
Average	0.81	0.14	0.12	0.66	0.59	0.34	0.40	0.78	<b>0.98</b>

# AutoGPT+P Results

Results show the potential of real-world applications. However,

- uncertainty can arise from object detection, affordance mapping, and vague task definitions
- No feedback from execution
- Did not consider plans longer than 20 steps.

# **Language Agent Tree Search Unifies Reasoning Acting and Planning in Language Models**

---

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, Yu-Xiong Wang

*ICML 2024*

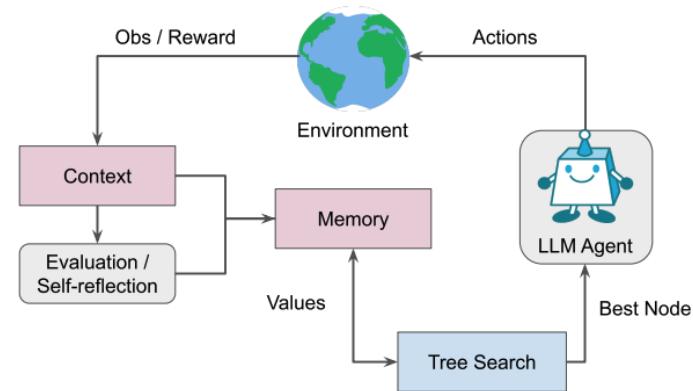
# LATS - Motivation/Background

- Chain-of-Thought (CoT) and ReAct face key limitations:
  - Error propagation in long reasoning chains
  - Inability to deliberate over multiple reasoning or acting paths

Thus, the authors propose **Language Agent Tree Search (LATS)**, which incorporates Monte Carlo Tree Search, environmental feedback, and LM-powered value functions/self-reflections for better exploration/planning.

# LATS - Intro

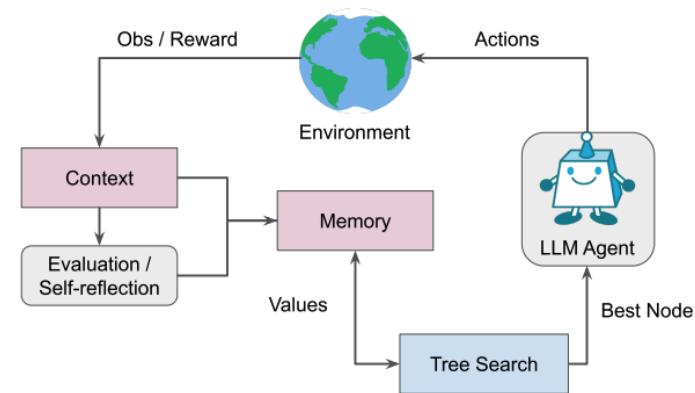
- LATS integrates three core capabilities in one framework:
  - **Reasoning:** Generates internal chains-of-thought
  - **Planning:** Employs a Monte Carlo Tree Search (MCTS) strategy for exploring alternatives
  - **Acting:** Interacts with external environments (e.g., API calls, web navigation)



*Figure 1.* Overview of LATS. Serving as a unified framework, LATS leverages an external environment and an MCTS-based search algorithm to improve reasoning and decision-making.

# LATS - Architecture/Workflow

- Input: Natural language query for LM
- Process:
  1. The LM **samples** multiple candidate reasoning/action trajectories
  2. **External feedback** is received from an environment (e.g., test suite results, API responses)
  3. **Self-reflection** is applied to failed trajectories to guide future searches
- Components:
  - LM Agent
  - Reflection Generator
  - **Value Function** for node evaluation



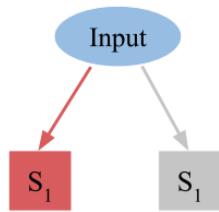
*Figure 1. Overview of LATS. Serving as a unified framework, LATS leverages an external environment and an MCTS-based search algorithm to improve reasoning and decision-making.*

# LATS - Monte Carlo Tree Search Recap

- Monte Carlo Tree Search (**MCTS**) - navigate combinatorial space of reasoning/action trajectories
  - **Selection:** Traverse the tree using a heuristic
  - **Expansion:** Sample multiple candidate actions
  - **Simulation:** Roll out trajectories until reaching a terminal state
  - **Backpropagation:** Update node values based on outcomes
- UCT - ~~I Inner Confidence~~ bounds applied to Trees
  - $$UCT(s) = V(s) + w \sqrt{\frac{\ln N(p)}{N(s)}}$$
  - $V(s)$ : Value of node  $s$  -  $V(s) = \frac{V_{\text{old}}(s)(N(s)-1)+r}{N(s)}$
  - $w$ : Exploration weight
  - $N(p)$  and  $N(s)$ : Visit counts

# LATS - Method Details

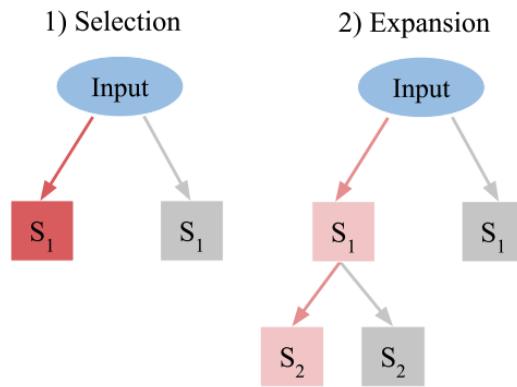
## 1) Selection



- **Selection**

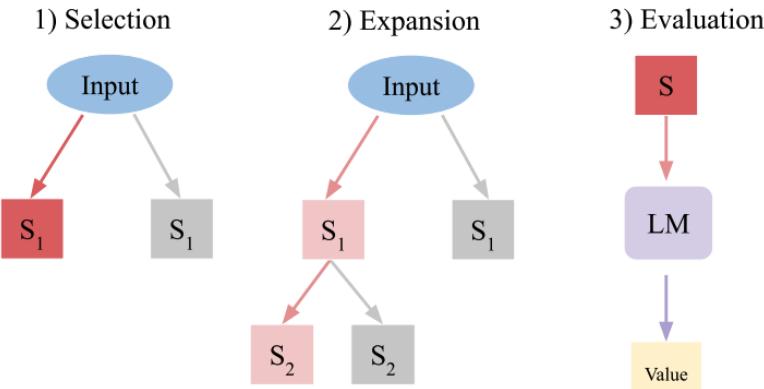
- Identify segment of the current tree most suitable for subsequent expansion
- Starting from the root node ( $s_0$ ), select children nodes until leaf is reached

# LATS - Method Details



- **Expansion:**
  - Expand tree by **sampling n actions** from language model ( $p_{\theta}$ )
  - Environment receives action and returns feedback as observation
  - This results in  $n$  new child nodes added to the tree
  - This tree is stored in an external long-term memory structure

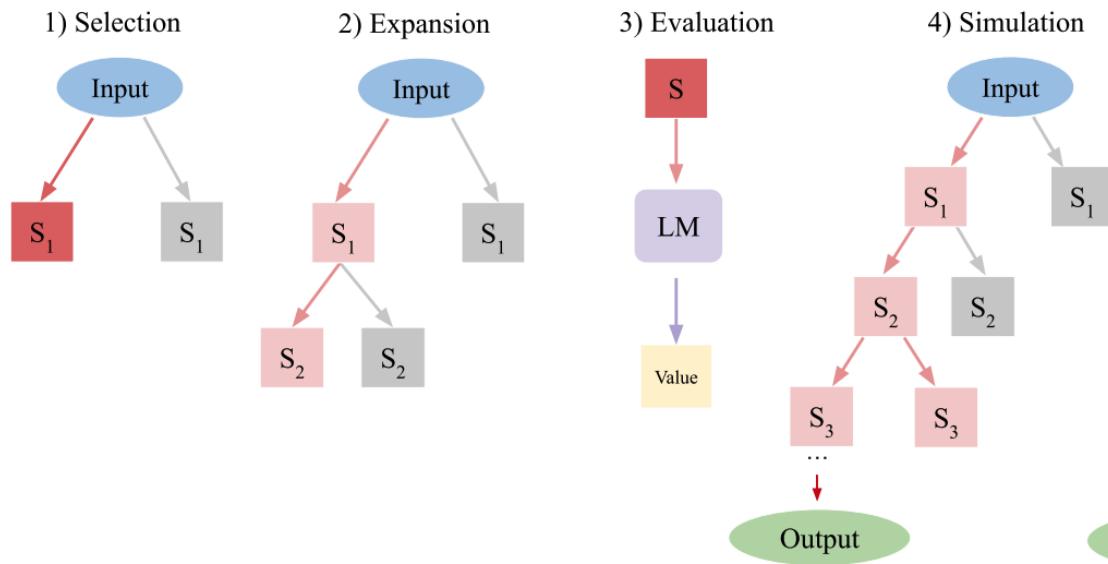
# LATS - Method Details



- **Self Scoring:** Ask LLM to end reasoning trace with score of correctness
  - Different than ToT since value is obtained **after** environmental feedback
- **Self Consistency:** Sample actions at given state multiple times and vote
$$V(s) = \lambda * LM(s) + (1 - \lambda) * SC(s)$$

- **Evaluation:**
  - Assign scalar value to each child for selection/backprop
  - LATS does not involve training: novel value function based on
    - self-generated LM score
    - self-consistency score

# LATS - Method Details

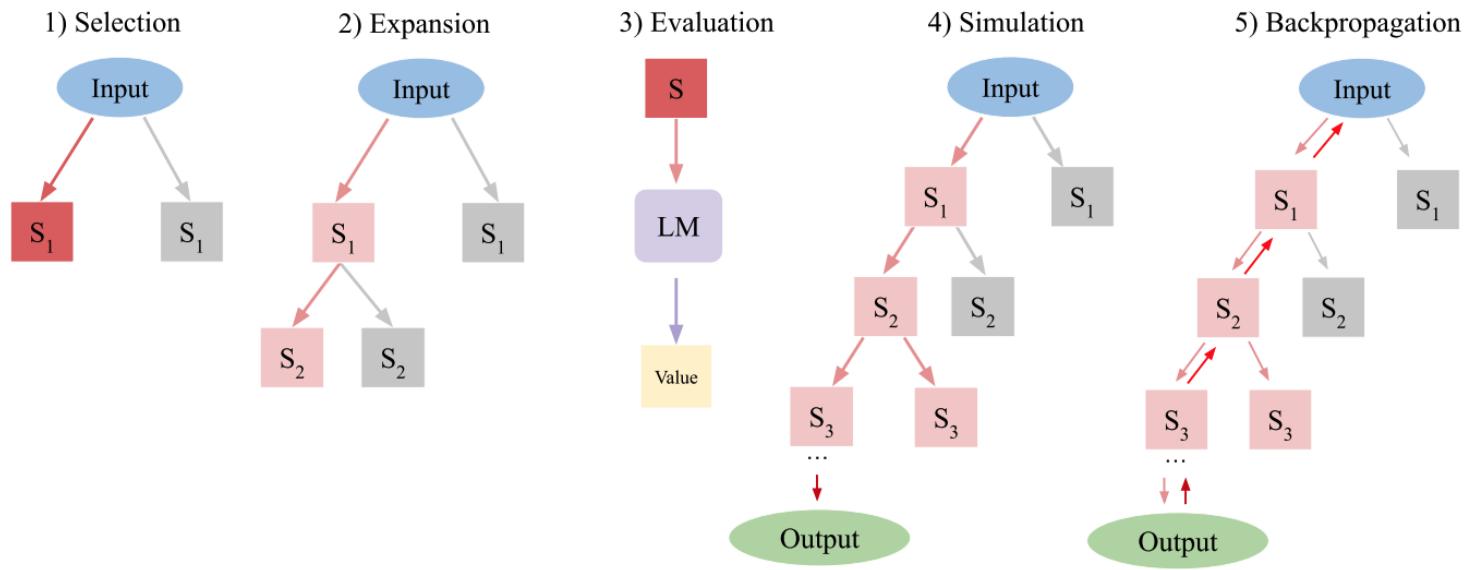


- 

## Simulation:

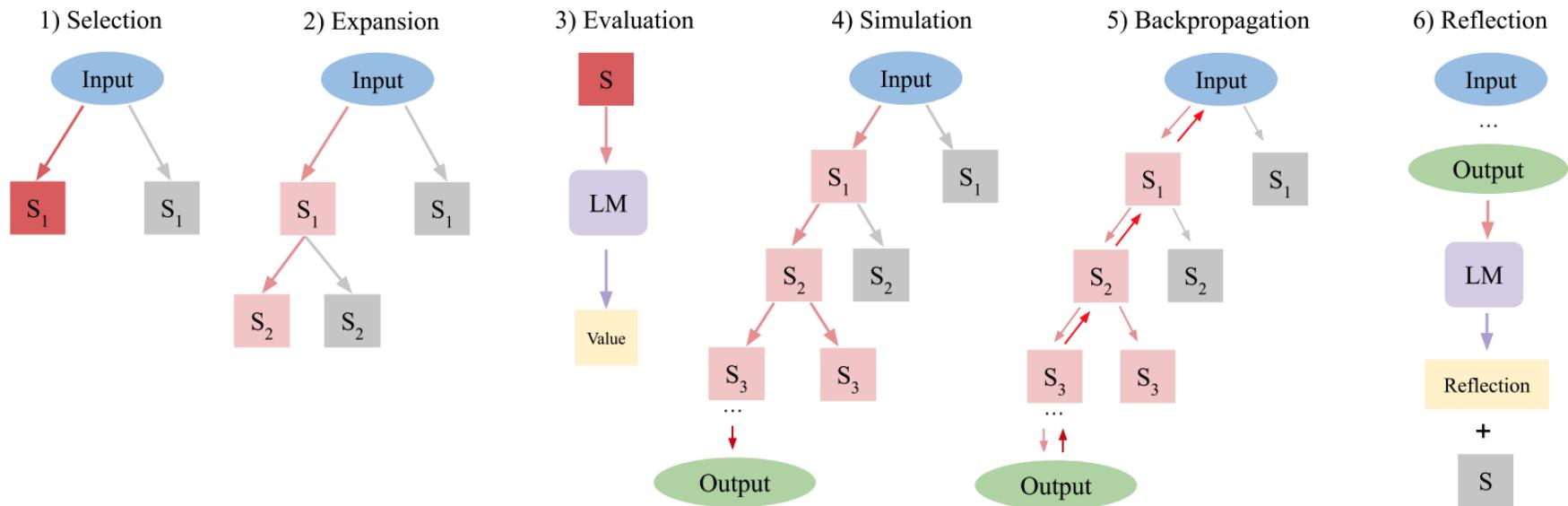
- Repeat expansion/evaluation and prioritize highest value nodes
- Terminal state gives objective feedback on correctness\*\*
- Success is determined by environment (i.e. buying from Amazon, booking flight, etc.)

# LATS - Method Details



- **Backpropagation:**
  - Update values  $V(s)$  of tree based on trajectory outcome
  - For each node, update value based on  $N(s_i) = N(s_{i-1}) + 1$   $\varepsilon V(s_i) = \frac{V(s_{i-1})N(s_{i-1})+r}{N(s_i)}$
  - New values used in UCT for selection of next node

# LATS - Method Details



- **Reflection:**

- When unsuccessful terminal node reached, prompt LLM with trajectory and final reward to self-reflect by summarizing errors in reasoning.
- Store failed trajectories and reflection in memory component
- Integrate in future ICL examples for agent to sidestep need for reinforcement learning

# LATS - Reflexion

	1. Decision making	2. Programming	3. Reasoning
(a) Task	You are in the middle of a room [...] <b>Task:</b> clean some pan and put it in countertop.	<b>Task:</b> You are given a list of two strings [...] of open '(' or close ')' parentheses only [...]	<b>Task:</b> What profession does John Lanchester and Alan Dean Foster have in common?
(b) Trajectory	[...] <b>Action:</b> take pan1 from stoveburner1 <b>Obs:</b> Nothing happens. [...] <b>Action:</b> clean pan1 with sinkbasin1 <b>Obs:</b> Nothing happens. [...]	<code>def match_parens(lst):     if s1.count('(') +         s2.count('(') == s1.count(')') +             s2.count(')': [...]     return 'No'</code>	<b>Think:</b> [...] novelist, journalist, critic [...] novelist, screenwriter [...] common is novelist and screenwriter. <b>Action:</b> "novelist, screenwriter"
(c) Evaluation (internal / external)	<b>Rule/LM Heuristic:</b> Hallucination.	<b>Self-generated unit tests fail:</b> <code>assert match_parens(...)</code>	<b>Environment Binary Reward:</b> 0
(d) Reflection	[...] tried to pick up the pan in stoveburner 1 [...] but the pan was not in stoveburner 1. [...]	[...] wrong because it only checks if the total count of open and close parentheses is equal [...] order of the parentheses [...]	[...] failed because I incorrectly assumed that they both had the same multiple professions [...] accurately identifying their professions.
(e) Next Trajectory	[...] <b>Action:</b> take pan 1 from stoveburner 2 [...] <b>Obs:</b> You put the pan 1 in countertop 1.	[...] <code>return 'Yes' if check(S1) or check(S2) else 'No'</code>	<b>Think:</b> [...] So the profession John Lanchester and Alan Dean Foster have in common is novelist. <b>Action:</b> "novelist"

Shinn et al.  
2023

- **Reflection:**

- When unsuccessful terminal node reached, prompt LLM with trajectory and final reward to self-reflect by summarizing errors in reasoning.
- Store failed trajectories and reflection in memory component
- Integrate in future ICL examples for agent to sidestep need for reinforcement learning

# LATS - Experimental Setup

- HotpotQA:
  - Multi-hop question-answering benchmark requiring retrieval 2+ Wikipedia passages
- Programming:
  - HumanEval, MBPP (Mostly Basic Python Problems)
  - Correctness of synthesized programs in Python from natural language docstrings
- WebShop:
  - Online shopping environment with 1.18M real-world products and 12k human instructions
  - Navigate a website via commands to purchase an item matching user specification
- Game of 24:
  - Construct 24 out of a set of numbers and basic operations

# LATS - Results (HotpotQA)

Prompt Method	HotpotQA (EM) ↑
Base LM	0.32
CoT (Wei et al., 2022)	0.34
CoT - SC (Wang et al., 2022)	0.38
ToT (Yao et al., 2023a)	0.55
RAP (Hao et al., 2023)	0.60
RAP ( $n = 10$ )	0.60
LATS (CoT)	<b>0.62</b>

Prompt Method	HotpotQA (EM) ↑
ReAct (Yao et al., 2023b)	0.32
ReAct (best of $k$ )	0.38
Reflexion (Shinn et al., 2023)	0.51
<i>ToT (ReAct)</i>	0.39
<i>RAP (ReAct)</i>	0.54
LATS (ReAct)	0.63
LATS ( $n = 3$ )	0.58
LATS ( $n = 10$ )	0.65
LATS (CoT + ReAct)	<b>0.71</b>

- LATS outperforms all baselines on reasoning/action HotpotQA

# LATS - Results (Programming)

Prompt Method	Model	Pass@1 ↑
CoT (Wei et al., 2022)	GPT-3.5	46.9
ReAct (Yao et al., 2023b)	GPT-3.5	56.9
Reflexion (Shinn et al., 2023)	GPT-3.5	68.1
ToT (Yao et al., 2023a)	GPT-3.5	54.4
RAP (Hao et al., 2023)	GPT-3.5	63.1
LATS (ReAct)	GPT-3.5	<b>83.8</b>
Base LM	GPT-4	80.1
Reflexion	GPT-4	91.0
LATS (ReAct)	GPT-4	<b>92.7</b>

Table 4. GPT-3.5 and GPT-4 Pass@1 accuracy on HumanEval. Prompting with LATS achieves the best performance. We sample 5 solutions during expansion for 8 iterations.

Prompt Method	Pass@1 ↑
CoT (Wei et al., 2022)	54.9
ReAct (Wei et al., 2022)	67.0
Reflexion (Shinn et al., 2023)	70.0
ToT (Yao et al., 2023a)	65.8
RAP (Hao et al., 2023)	71.4
LATS (ReAct)	<b>81.1</b>

Table 5. GPT-3.5 Pass@1 accuracy on MBPP. Prompting with LATS achieves the highest performance. We sample 5 solutions during expansion for 8 iterations.

# LATS - Results (WebShop)

Method	Score ↑	SR ↑
ReAct (Yao et al., 2023b)	53.8	28.0
ReAct (best of k)	59.1	32.0
Reflexion (Shinn et al., 2023)	64.2	35.0
LATS (ReAct)	<b>75.9</b>	<b>38.0</b>
IL (Yao et al., 2022)	59.9	29.1
IL+RL (Yao et al., 2022)	62.4	28.7
Fine-tuning (Furuta et al., 2024)	67.5	45.0
<i>Expert</i>	82.1	59.6

*Table 6.* Score and success rate (SR) on WebShop. Results are organized into prompting, RL-based training, and human performance. For the same number of iterations, LATS improves both score and SR and surpasses RL-based training.

## LATS - Results (Game of 24)

Prompt Method	Game of 24 (Success Rate) ↑
CoT (Wei et al., 2022)	0.08
Reflexion (Shinn et al., 2023)	0.12
ToT (Yao et al., 2023a)	0.20
RAP (Hao et al., 2023)	0.40
LATS (CoT)	<b>0.44</b>

*Table 7.* Results on Game of 24 with GPT-3.5. We sample  $n = 5$  nodes and  $k = 30$  trajectories.

# LATS - Strengths/Weaknesses

- Contributions
  - **Generality:** Supports both reasoning-only and interactive decision-making tasks
  - **Deliberation:** Uses MCTS to systematically balance exploration and exploitation
  - **Adaptability:** Incorporates external feedback and self-reflection to refine search
  - **Efficiency:** Outperforms alternatives with on average less nodes/state expansion
- Limitations
  - Higher computational cost than simpler methods like ReAct/Reflexion
  - Assumes reversible states in the environment — not universally applicable
- Future Work
  - Multi-agent LATS
  - Reduce inference token consumption/computation cost
  - Richer external feedback (multi-modal, etc.)

# LATS - Extra (ReAct)

## (2b) ReAct (Reason + Act)

**Act 1:** Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ... ]

**Act 2:** Go to cabinet 1

**Obs 2:** On the cabinet 1, you see a vase 2.

(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

**Act 6:** Go to countertop 3

**Obs 6:** On the countertop 3, you see a apple 1, a bread 1, ..... a pepper shaker 1, and a vase 3.

**Act 7:** Take pepper shaker 1 from countertop 3

**Obs 7:** You pick up the pepper shaker 1 from the countertop 3.

**Act 8:** Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1. ]

**Act 9:** Go to drawer 1

**Obs 9:** Drawer 1 is closed.

**Act 10:** Open drawer 1

**Obs 10:** You open Drawer 1 ...

**Act 11:** Put pepper shaker 1 in/on drawer 1

**Obs 11:** You put pepper shaker 1 in/on the drawer 1.



# **TravelPlanner: A Benchmark for Real-World Planning with Language Agents**

---

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, Yu Su

*ICML 2024*

# TravelPlanner: Motivation

- Planning is a hallmark of human intelligence
- Capabilities of LLMs
- Existing tasks focus on pared down problems
  - e.g. web tool use, math reasoning
  - single objectives
  - pre-defined sets of actions
- What does human planning look like?

# TravelPlanner: Motivation

- long horizon task
  - a multiple day itinerary
- constraints
  - commonsense:
    - e.g. humans cannot teleport
  - user/hard:
    - e.g. budget, dietary restrictions
  - environment:
    - e.g. no available flights
- agency
  - acquire information from environment
  - reason over acquired information

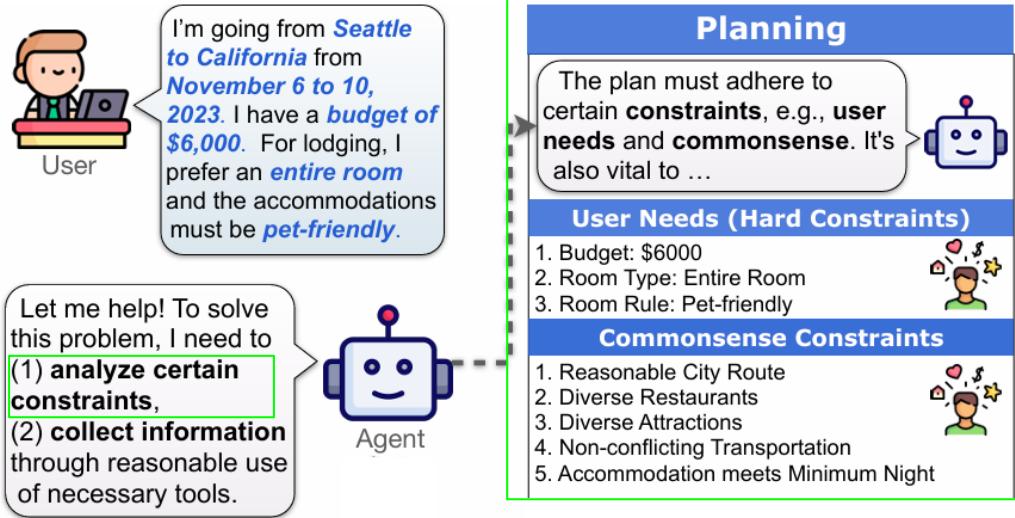
# TravelPlanner: Introduction



I'm going from *Seattle to California* from *November 6 to 10, 2023*. I have a *budget of \$6,000*. For lodging, I prefer an *entire room* and the accommodations must be *pet-friendly*.

- 1,225 diverse user queries and reference plans
- complexity: travel duration
  - 3 days: 1 city
  - 5 days: 1 state, 2 cities
  - 7 days: 1 state, 3 cities

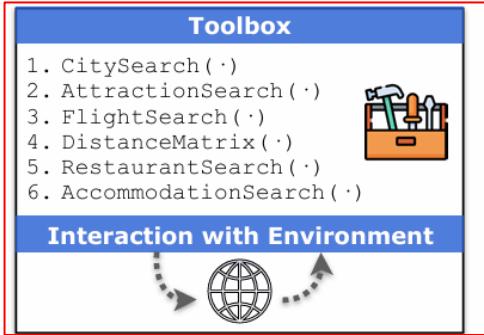
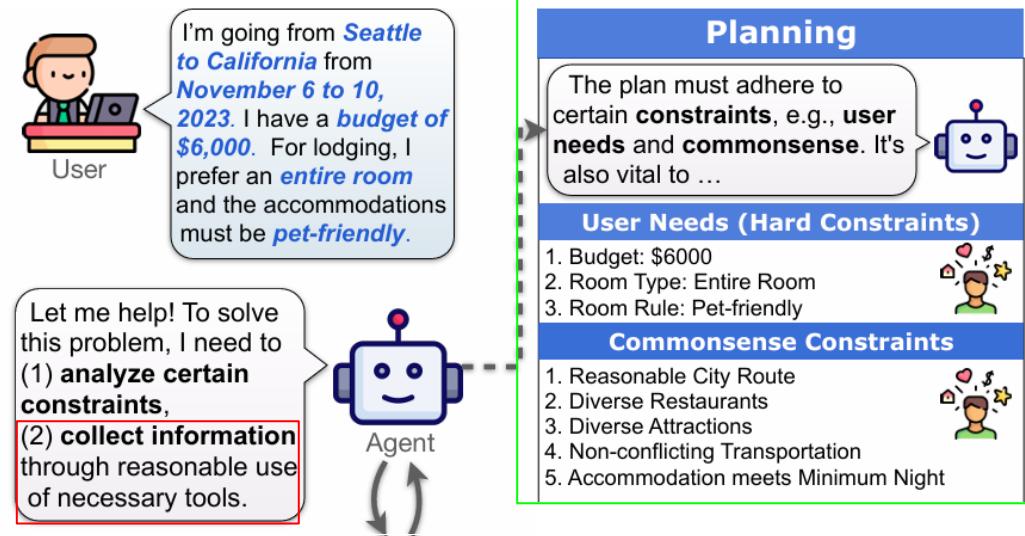
# TravelPlanner: Introduction



constraint complexity:

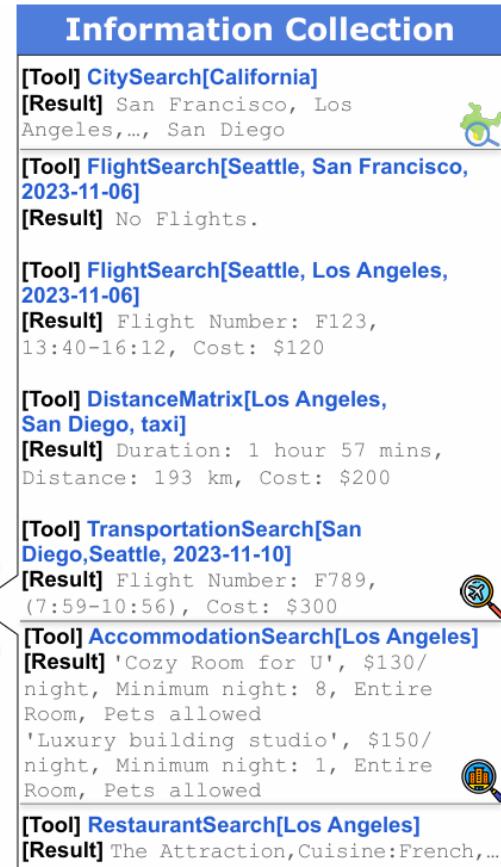
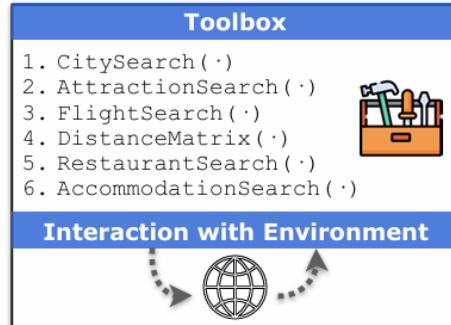
- **easy:**
  - 1 traveller
  - budget
- **medium:**
  - 2-8 travellers
  - +1 hard constraint
- **hard:**
  - +2 hard constraints

# TravelPlanner: Introduction



- 4 millions data entries for 6 tools
- static environment for fair evaluation

# TravelPlanner: Introduction



# TravelPlanner: Evaluation Metrics

- delivery rate:
  - plan delivered within 30 steps
- final pass rate:
  - feasible plans that meet all constraints

# TravelPlanner: Evaluation Metrics

- commonsense constraint pass rate:
  - follow constraints without explicit instruction
- hard constraint pass rate:
  - follow hard constraints

$$\text{Micro Pass Rate} = \frac{\sum_{p \in P} \sum_{c \in C_p} \mathbb{1}_{\text{passed}(c,p)}}{\sum_{p \in P} |C_p|}$$

$$\text{Macro Pass Rate} = \frac{\sum_{p \in P} \mathbb{1}_{\text{passed}(C_p, p)}}{|P|}$$

$P$ : all plans

$C_p$ : all constraints on  $p$

# TravelPlanner: Results

	Validation (#180)						Test (#1,000)					
	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate
		Micro	Macro	Micro	Macro			Micro	Macro	Micro	Macro	
Greedy Search	100	74.4	0	60.8	37.8	0	100	72.0	0	52.4	31.8	0
<i>Two-stage</i>												
Mistral-7B-32K (Jiang et al., 2023)	8.9	5.9	0	0	0	0	7.0	4.8	0	0	0	0
Mixtral-8×7B-MoE (Jiang et al., 2024)	49.4	30.0	0	1.2	0.6	0	51.2	32.2	0.2	0.7	0.4	0
Gemini Pro (G Team et al., 2023)	28.9	18.9	0	0.5	0.6	0	39.1	24.9	0	0.6	0.1	0
GPT-3.5-Turbo (OpenAI, 2022)	86.7	54.0	0	0	0	0	91.8	57.9	0	0.5	0.6	0
GPT-4-Turbo (OpenAI, 2023)	<b>89.4</b>	<b>61.1</b>	<b>2.8</b>	<b>15.2</b>	<b>10.6</b>	<b>0.6</b>	<b>93.1</b>	<b>63.3</b>	<b>2.0</b>	<b>10.5</b>	<b>5.5</b>	<b>0.6</b>

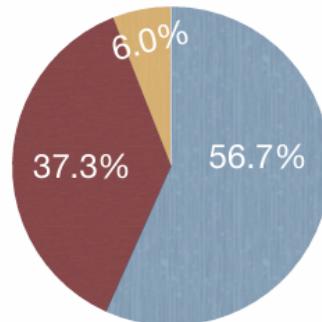
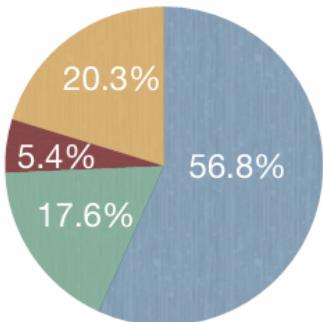
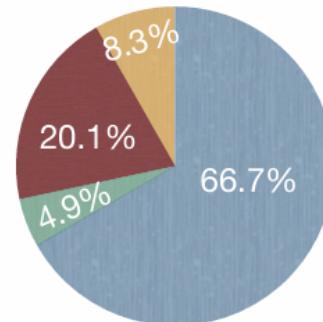
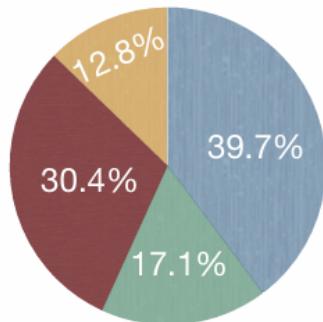
# TravelPlanner: Results

	Validation (#180)						Test (#1,000)					
	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate
		Micro	Macro	Micro	Macro			Micro	Macro	Micro	Macro	
Greedy Search	100	74.4	0	60.8	37.8	0	100	72.0	0	52.4	31.8	0
<i>Two-stage</i>												
Mistral-7B-32K (Jiang et al., 2023)	8.9	5.9	0	0	0	0	7.0	4.8	0	0	0	0
Mixtral-8x7B-MoE (Jiang et al., 2024)	49.4	30.0	0	1.2	0.6	0	51.2	32.2	0.2	0.7	0.4	0
Gemini Pro (G Team et al., 2023)	28.9	18.9	0	0.5	0.6	0	39.1	24.9	0	0.6	0.1	0
GPT-3.5-Turbo (OpenAI, 2022)	86.7	54.0	0	0	0	0	91.8	57.9	0	0.5	0.6	0
GPT-4-Turbo (OpenAI, 2023)	<b>89.4</b>	<b>61.1</b>	<b>2.8</b>	<b>15.2</b>	<b>10.6</b>	<b>0.6</b>	<b>93.1</b>	<b>63.3</b>	<b>2.0</b>	<b>10.5</b>	<b>5.5</b>	<b>0.6</b>
<i>Sole-planning</i>												
Direct <sub>GPT-3.5-Turbo</sub>	100	60.2	4.4	11.0	2.8	0	100	59.5	2.7	9.5	4.4	0.6
CoT <sub>GPT-3.5-Turbo</sub>	100	66.3	3.3	11.9	5.0	0	100	64.4	2.3	9.8	3.8	0.4
ReAct <sub>GPT-3.5-Turbo</sub>	82.2	47.6	3.9	11.4	6.7	0.6	81.6	45.9	2.5	10.7	3.1	0.7
Reflexion <sub>GPT-3.5-Turbo</sub>	93.9	53.8	2.8	11.0	2.8	0	92.1	52.1	2.2	9.9	3.8	0.6
Direct <sub>Mixtral-8x7B-MoE</sub>	100	68.1	5.0	3.3	1.1	0	99.3	67.0	3.7	3.9	1.6	0.7
Direct <sub>Gemini Pro</sub>	93.9	65.0	8.3	9.3	4.4	0.6	93.7	64.7	7.9	10.6	4.7	2.1
Direct <sub>GPT-4-Turbo</sub>	<b>100</b>	<b>80.4</b>	<b>17.2</b>	<b>47.1</b>	<b>22.2</b>	<b>4.4</b>	<b>100</b>	<b>80.6</b>	<b>15.2</b>	<b>44.3</b>	<b>23.1</b>	<b>4.4</b>

# TravelPlanner: Analysis

- Incorrect tool use
  - e.g. malformed arguments
- Dead loops
  - e.g. invalid actions, repetitive loops

● Max Step Limit ● Argument Error ● Invalid Action Dead Loop ● Same Action Dead Loop



Mixtral-7x8B-MoE

Gemini Pro

GPT-3.5-Turbo

GPT-4-Turbo

# TravelPlanner: Analysis

Constraint Type	Two-stage			Sole-planning		
	Easy	Medium	Hard	Easy	Medium	Hard
<i>Commonsense Constraint</i>						
Within Sandbox	37.4	31.2	33.9	76.4	71.5	79.3
Complete Information	53.4	52.9	58.0	94.5	96.4	96.2
Within Current City	69.3	67.3	68.3	89.1	80.8	82.4
Reasonable City Route	44.5	45.6	54.9	99.4	99.7	99.1
Diverse Restaurants	85.1	81.4	86.8	91.1	89.8	87.8
Diverse Attractions	94.3	90.4	94.0	100.0	100.0	100.0
Non-conf. Transportation	70.1	73.3	83.1	60.1	56.5	87.5
Minimum Nights Stay	46.8	46.2	51.1	37.4	28.8	30.1
<i>Hard Constraint</i>						
Budget	10.1	8.4	4.4	37.4	35.1	25.1
Room Rule	-	5.6	11.3	-	31.5	43.6
Cuisine	-	10.8	11.4	-	57.5	46.7
Room Type	-	12.4	13.8	-	45.7	56.7
Transportation	-	-	18.6	-	-	77.5
<i>Final</i>						
Final Pass Rate	1.1	0.3	0.3	8.0	2.7	2.2

# TravelPlanner: Analysis

Average	Agent			Reference		
	3-day	5-day	7-day	3-day	5-day	7-day
FlightSearch	0.7	1.2	0.8	2.0	3.0	4.0
DistanceMatrix	0.3	0.6	1.2	2.0	3.0	4.0
RestaurantSearch	0.9	1.5	2.4	1.0	2.0	3.0
AttractionSearch	0.9	1.7	2.4	1.0	2.0	3.0
AccommodationSearch	0.9	1.6	2.5	1.0	2.0	3.0

# TravelPlanner: Analysis

Tool-use	Direct	Reflexion
<p><b>Query:</b> Please plan a travel itinerary for me. I'm departing from Cincinnati and heading to Norfolk for three days. The dates of travel are from March 10th to <b>March 12th, 2022</b>. I have a budget of \$1,400 for this trip.</p> <p><b>Trajectory:</b></p> <p>Action 1: FlightSearch[Cincinnati, Norfolk, <b>2023-03-10</b>] Observation 1: There is no flight from Cincinnati to Norfolk on <b>2023-03-10</b>.</p> <p>Action 2: FlightSearch[Cincinnati, Norfolk, <b>2023-03-09</b>]</p> <p>Action 3: FlightSearch[Cincinnati, Norfolk, <b>2023-03-11</b>]</p> <p>Action 4: FlightSearch[Cincinnati, Richmond, <b>2023-03-10</b>] ...</p> <p>Action 8: FlightSearch[Cincinnati, Charlotte, 2023-03-10]</p> <p>Action 9: <b>I'm sorry, but as an AI language model, I am unable to execute actions or observe their outcomes.</b></p>	<p><b>Query:</b> Please create a travel itinerary for a solo traveler departing from Jacksonville and heading to Los Angeles for a period of 3 days, from March 25th to March 27th, 2022. The budget for this trip is now set at \$2,400.</p> <p><b>Plan:</b></p> <p>Day 1: transportation: <b>Flight Number: F3765485, from Jacksonville to Los Angeles</b> ...</p> <p>accommodation: Midtown East 2 Bed/2 Bath Elevator Building Apt (7-day minimum night)</p> <p>Day 2: ...</p> <p>accommodation: Midtown East 2 Bed/2 Bath Elevator Building Apt (7-day minimum night)</p> <p>Day 3: transportation: <b>Flight Number: F3765485, from Los Angeles to Jacksonville</b> ...</p>	<p><b>Query:</b> Can you help construct a travel plan that begins in Philadelphia and includes visits to 3 cities in Virginia? The trip duration is for 7 days, from March 15th to March 21st, 2022, with a total budget of \$1,800.</p> <p><b>Trajectory:</b></p> <p>...</p> <p>Observation 3: <b>The cost of your current plan is 316.0 dollars.</b></p> <p>Thought 4: <b>I need to select a more budget-friendly accommodation</b> to keep the overall trip within the \$1,800 budget.</p> <p>Action 4: CostEnquiry[{"transportation": "-", "breakfast": "-", "attraction": "Virginia Museum", "lunch": "Guang Zhou Chinese Restaurant, Richmond", "dinner": "Sandpiper Restaurant &amp; Lounge, Richmond", "accommodation": "2 bd 2 bathroom Apartment in Upper East Side, Richmond"}]</p> <p>Observation 4: <b>The cost of your plan is 389.0 dollars.</b></p>

# TravelPlanner: Conclusions

- LLMs still lack many of the skills necessary to plan in real world scenarios
- In particular, they struggle with
  - multiple constraints
  - multiple components of a task
  - large amounts of information
  - aligning their actions with their reasoning
  - proper formatting for tool use