

# VOYAGER: An Open-Ended Embodied Agent with Large Language Models

---

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar et al.

Introduction

# Overview

---

- This research introduces “**VOYAGER**”, LLM-powered lifelong learning agent for Minecraft
  - Continuously explores the world, acquires diverse skills, and makes novel discoveries
  - With its **skill library**, it can memorize how to solve various tasks
    - Generalizable; able to apply learned skills in different worlds
    - Interpretable; learned skills can be easily understood by humans

## Introduction

# Why Minecraft?

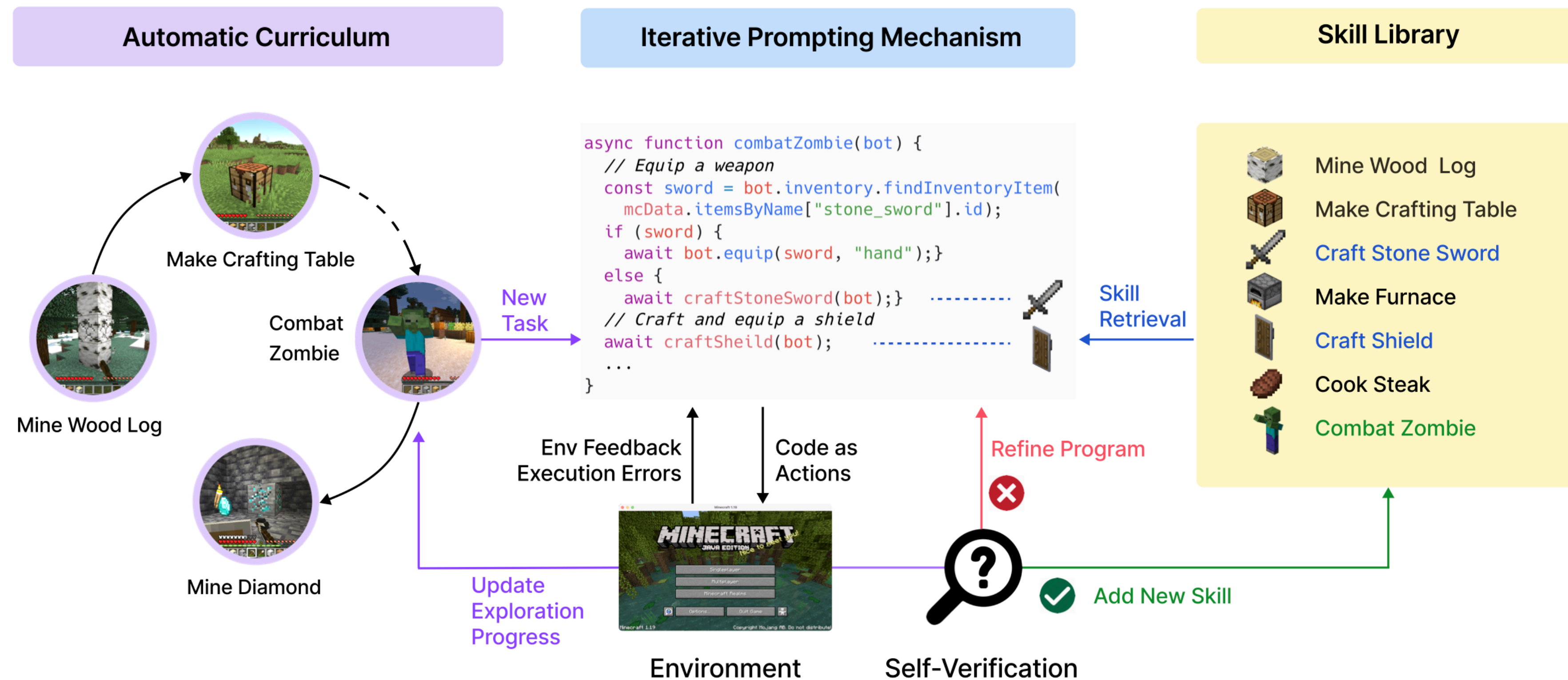
---

- Minecraft does not impose any specific goals on the player
  - It is a sandbox with various structures, items and enemies
  - “Open-ended” environment
- Provides suitable environment to measure how well agents can explore and make new discoveries, without human intervention
  - E.g. mining new ore, crafting new items, building a house, fighting against enemies...
- Classical approaches: Reinforcement Learning, Imitation Learning...
  - Difficulty in systematic exploration, interpretability and generalization

# Architecture of the agent

## Components of VOYAGER

- VOYAGER consists of three key components:



Architecture of the agent

# Automatic Curriculum

---

- An **Automatic Curriculum** presents the agent with the next goal to be achieved
- Leverages the vast knowledge contained in GPT-4
- The input prompt to GPT-4 consists of:
  - The agent's current state:
    - E.g. inventory, equipment, nearby blocks, health, position...
  - Previously completed and failed tasks
  - Additional context
    - Generated by GPT-3.5, based on game knowledge base



**Inventory (5/36):** {'oak\_planks': 3, 'stick': 4, 'crafting\_table': 1, 'stone': 3, 'wooden\_pickaxe': 1}

GPT-4

**Reasoning:** Since you have a wooden pickaxe and some stones, it would be beneficial to upgrade your pickaxe to a stone pickaxe for better efficiency.  
**Task:** Craft 1 stone pickaxe.



**Biome:** river  
**Inventory (4/36):** {'sugar\_cane': 3, 'paper': 3, 'fishing\_rod': 1, 'wooden\_pickaxe': 1}

GPT-4

**Reasoning:** Since you have a fishing rod in your inventory and you are near a river biome, it would be a good opportunity to catch some fish for food and experience.  
**Task:** Catch 1 fish.



**Nearby entities:** pig, cat, villager  
**Health:** 12/20  
**Hunger:** 0/20

GPT-4

**Reasoning:** Your hunger is at 0, which means you need to find food to replenish your hunger. Since there are pigs nearby, you can kill one to obtain raw porkchops.  
**Task:** Kill 1 pig.



**Inventory (6/36):** {'furnace': 1, 'stone\_pickaxe': 1, 'oak\_planks': 7, 'crafting\_table': 1, 'raw\_iron': 4, 'coal': 1}

GPT-4

**Reasoning:** You have raw iron and coal, and you have a furnace. It's time to smelt the iron to make iron ingots, which can be used to craft better tools and armor.  
**Task:** Smelt 4 raw iron.



**Time:** night  
**Nearby entities:** zombie  
**Equipment:** 'stone\_sword', 'shield'

GPT-4

**Reasoning:** Since it's night and there's a zombie nearby, it's a good opportunity to try killing the zombie now that you have a stone sword and a shield equipped.  
**Task:** Kill 1 zombie.

Tasks proposed by the automatic curriculum

## Architecture of the agent

# Skill Library

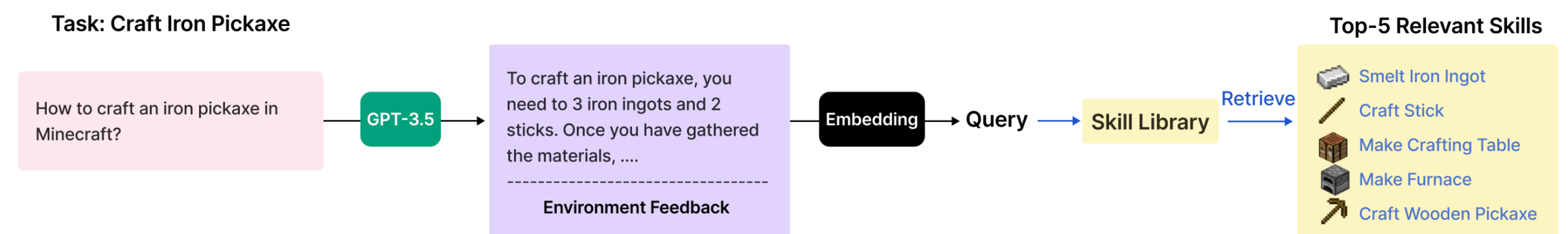
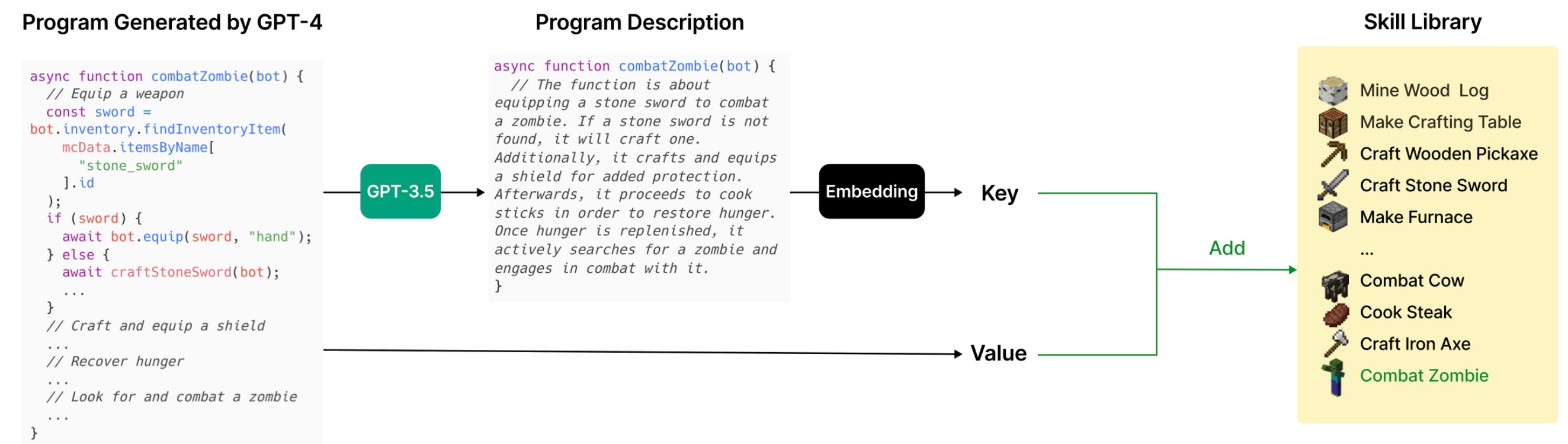
---

- A **Skill Library** stores and retrieves learned skills
  - Each skill is represented with executable JavaScript code
    - Controls player character via MineFlyer API
- Skill generation
  - The input prompt to GPT-4 consists of:
    - Control primitive APIs
    - Relevant skills
    - Generated code from the last round, environment feedback, execution errors
    - Agent's current state

# Architecture of the agent

## Skill Library

- A newly generated skill is added to skill library if it passes self-verification
- Indexed by skill description
  - Generated by GPT-3.5



Top: Adding a new skill / Bottom: Skill retrieval



# Architecture of the agent

## Skill Library

---

```
async function craftWoodenPlanks(bot) {
  const logNames = ["oak_log", "birch_log", "spruce_log", "jungle_log",
"acacia_log", "dark_oak_log", "mangrove_log"];
  const plankNames = ["oak_planks", "birch_planks", "spruce_planks", "jungle_planks",
"acacia_planks", "dark_oak_planks", "mangrove_planks"];
  const logInInventory = logNames.find(logName =>
bot.inventory.count(mcData.itemsByName[logName].id) > 0);
  if (!logInInventory) {
    bot.chat("No wooden log in inventory. Mining a wooden log...");
    await mineWoodLog(bot);
  }
  const logIndex = logNames.indexOf(logInInventory);
  const plankName = plankNames[logIndex];
  bot.chat(`Crafting 4 ${plankName}...`);
  await craftItem(bot, plankName, 1);
  bot.chat(`4 ${plankName} crafted.`);
}
```

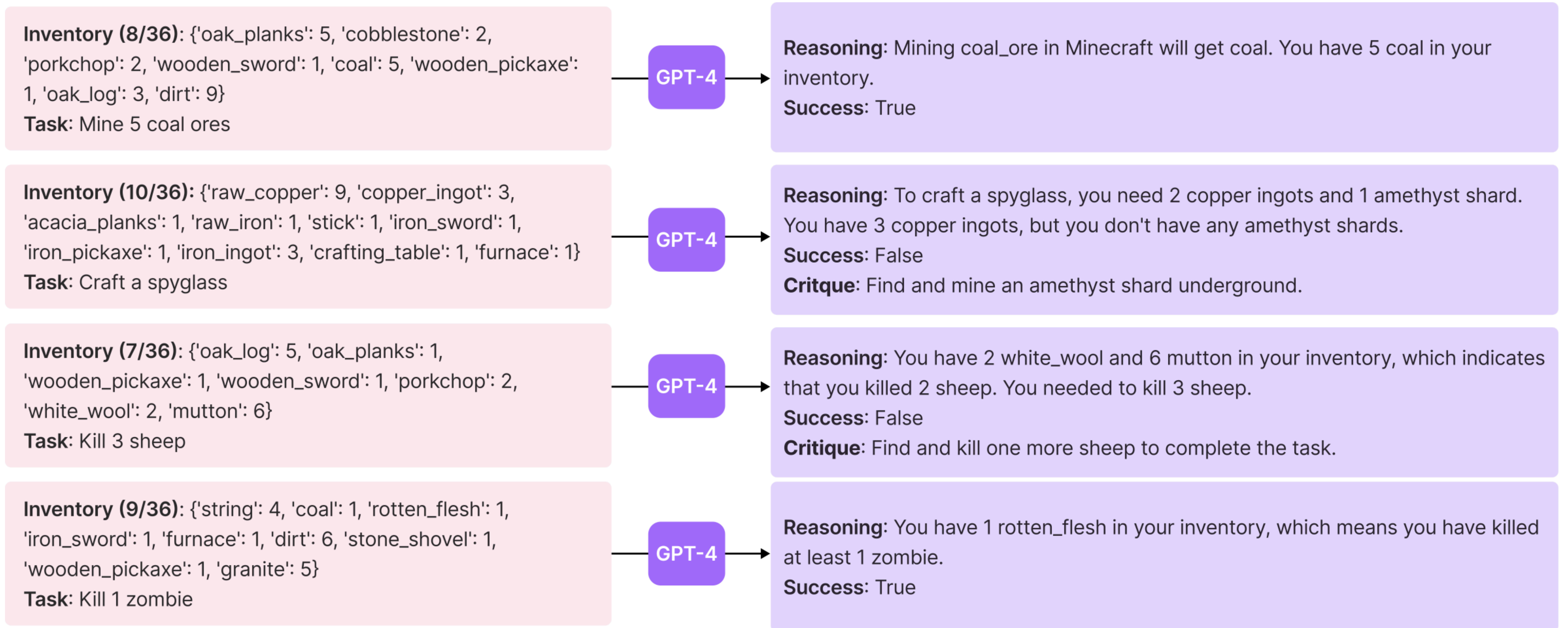
Skill library example 1: craftWoodenPlanks.

Architecture of the agent

# Iterative Prompting Mechanism

---

- LLMs struggle to produce the correct code in one shot
  - E.g. Hallucination
- An **Iterative Prompting Mechanism** tries to solve this problem
  1. Execute the generated program and obtain feedbacks from the game and execution errors (if any)
  2. Incorporate the feedback into GPT-4's prompt for code refinement
  3. Repeats 1 and 2 until a **self-verification** module confirms the task completion
  4. Add the program to the skill library



Self-verification examples.

## Experiments

# Baselines

---

- They managed to compare VOYAGER with the following methods. These methods were originally designed for NLP tasks:
  - ReAct
    - *ReAct: Synergizing Reasoning and Acting in Language Models*
  - Reflexion
    - *Reflexion: Language Agents with Verbal Reinforcement Learning*
  - AutoGPT
    - *Significant-Gravitas/AutoGPT: An experimental open-source attempt to make GPT-4 fully autonomous.*

# Experiments

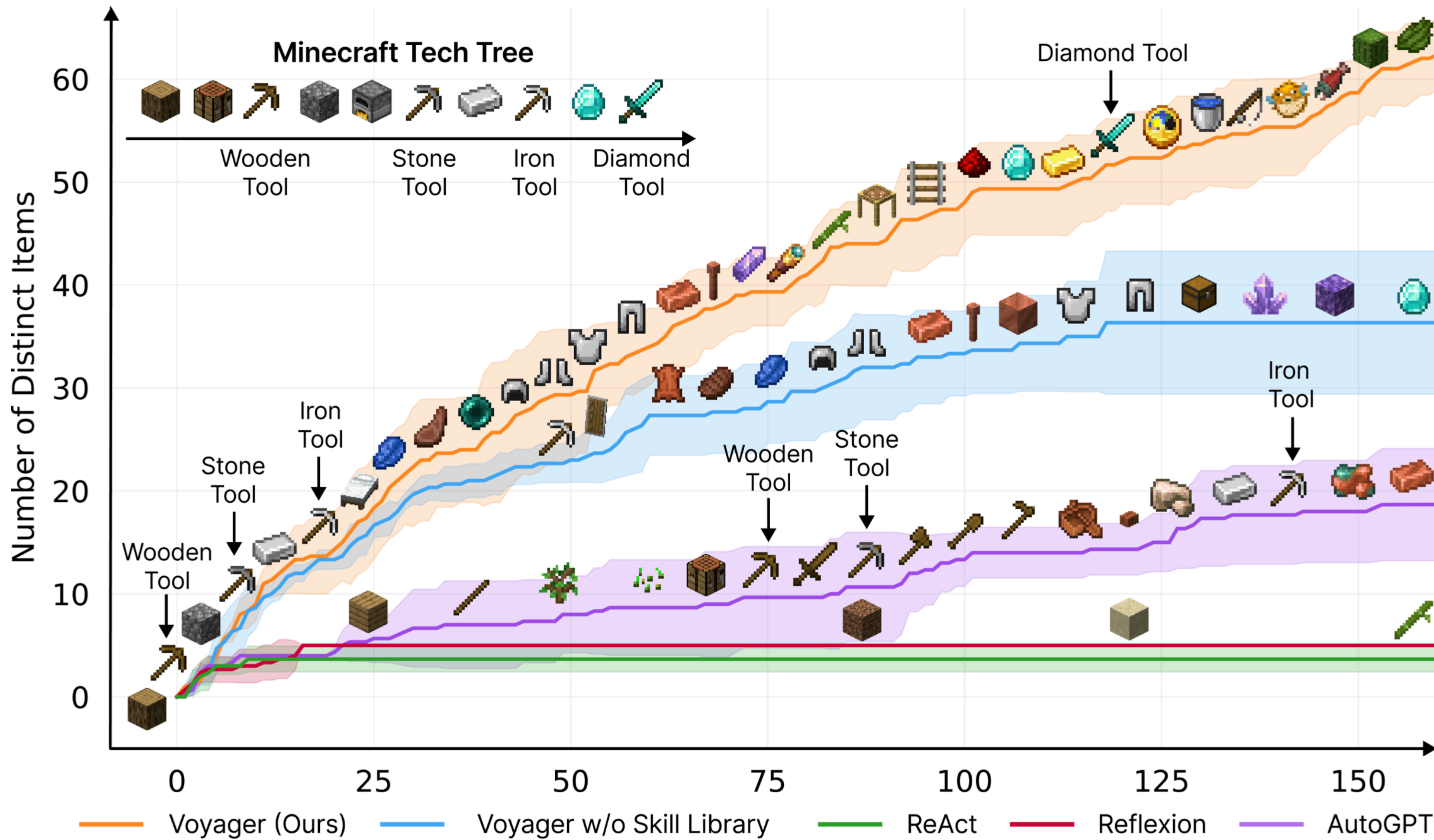
## Results against baselines

---

Method	Wooden Tool	Stone Tool	Iron Tool	Diamond Tool
ReAct [29]	N/A (0/3)	N/A (0/3)	N/A (0/3)	N/A (0/3)
Reflexion [30]	N/A (0/3)	N/A (0/3)	N/A (0/3)	N/A (0/3)
AutoGPT [28]	92 ± 72 ( <b>3/3</b> )	94 ± 72 ( <b>3/3</b> )	135 ± 103 ( <b>3/3</b> )	N/A (0/3)
VOYAGER w/o Skill Library	<b>7 ± 2</b> ( <b>3/3</b> )	<b>9 ± 4</b> ( <b>3/3</b> )	29 ± 11 ( <b>3/3</b> )	N/A (0/3)
VOYAGER (Ours)	<b>6 ± 2</b> ( <b>3/3</b> )	<b>11 ± 2</b> ( <b>3/3</b> )	<b>21 ± 7</b> ( <b>3/3</b> )	<b>102</b> ( <b>1/3</b> )

---

- The average number of iterations to reach each milestone
  - Fractions show how many times the agent could reach the milestone
- VOYAGER consistently progressed through tech tree with fewer iterations

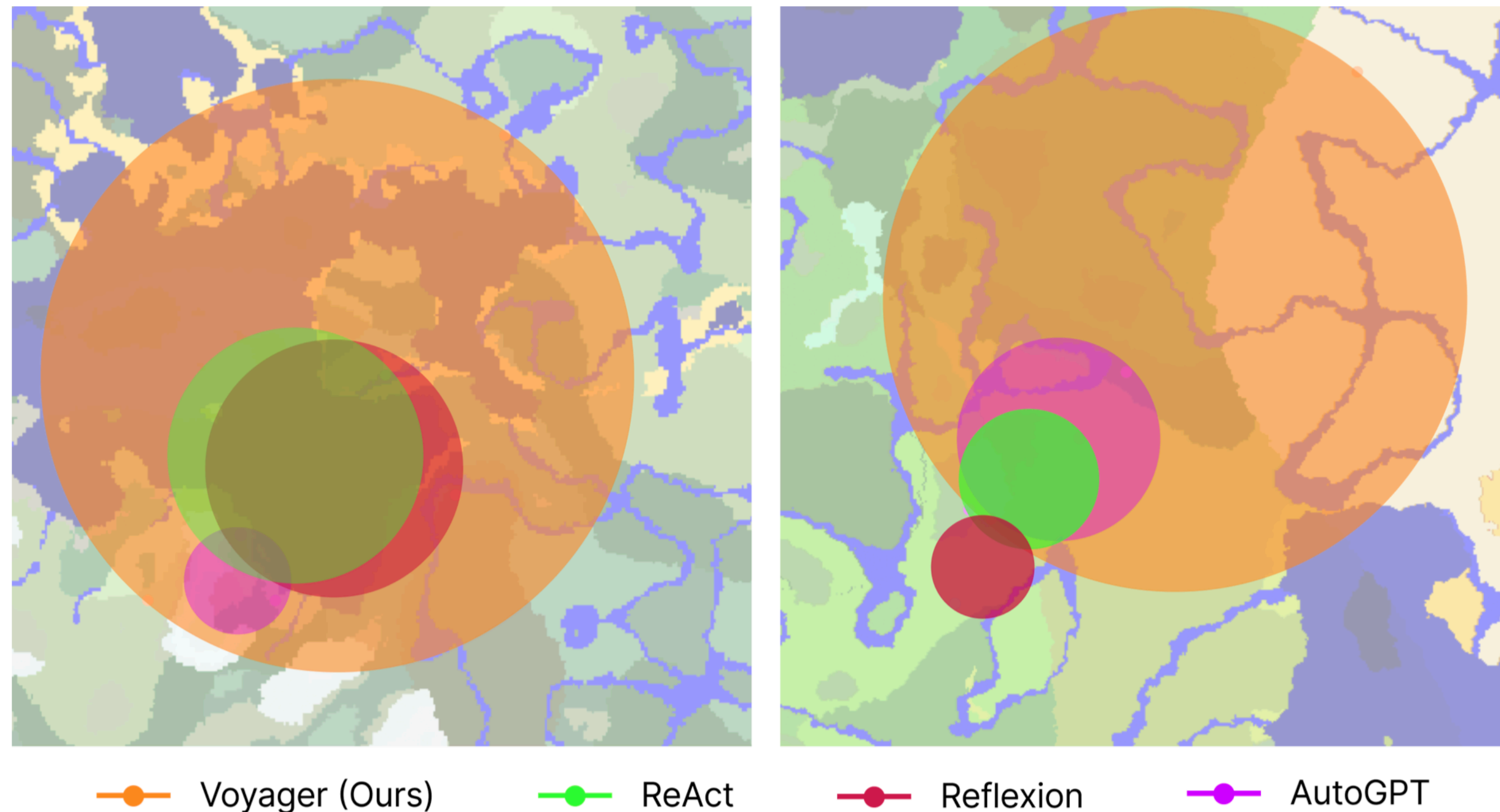


*VOYAGER: An Open-Ended Embodied Agent with Large Language Models*

# Experiments

## Results against baselines

---



- Shows how far the agents could traverse
- VOYAGER could navigate 2.3x longer distance compared to baselines

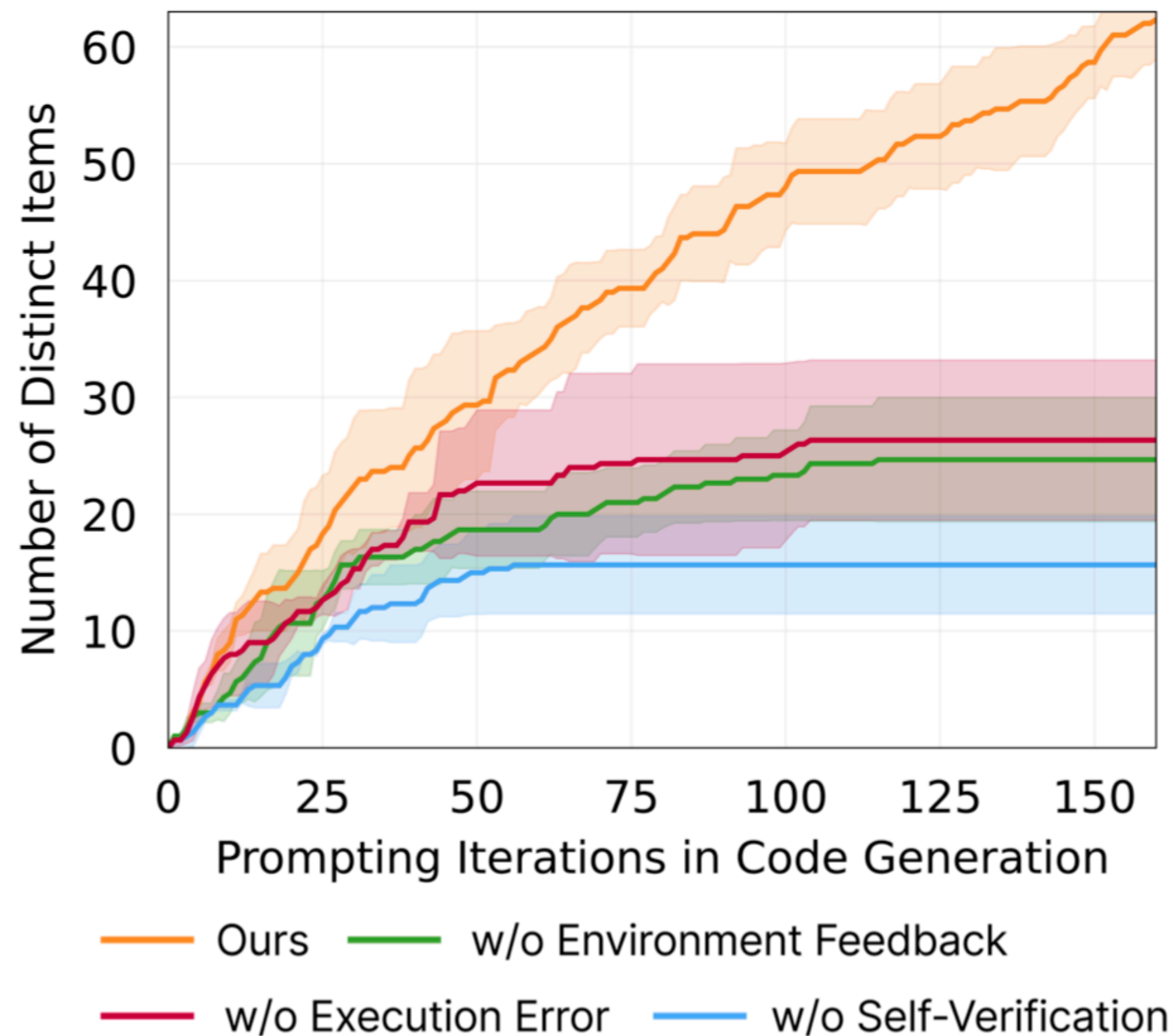
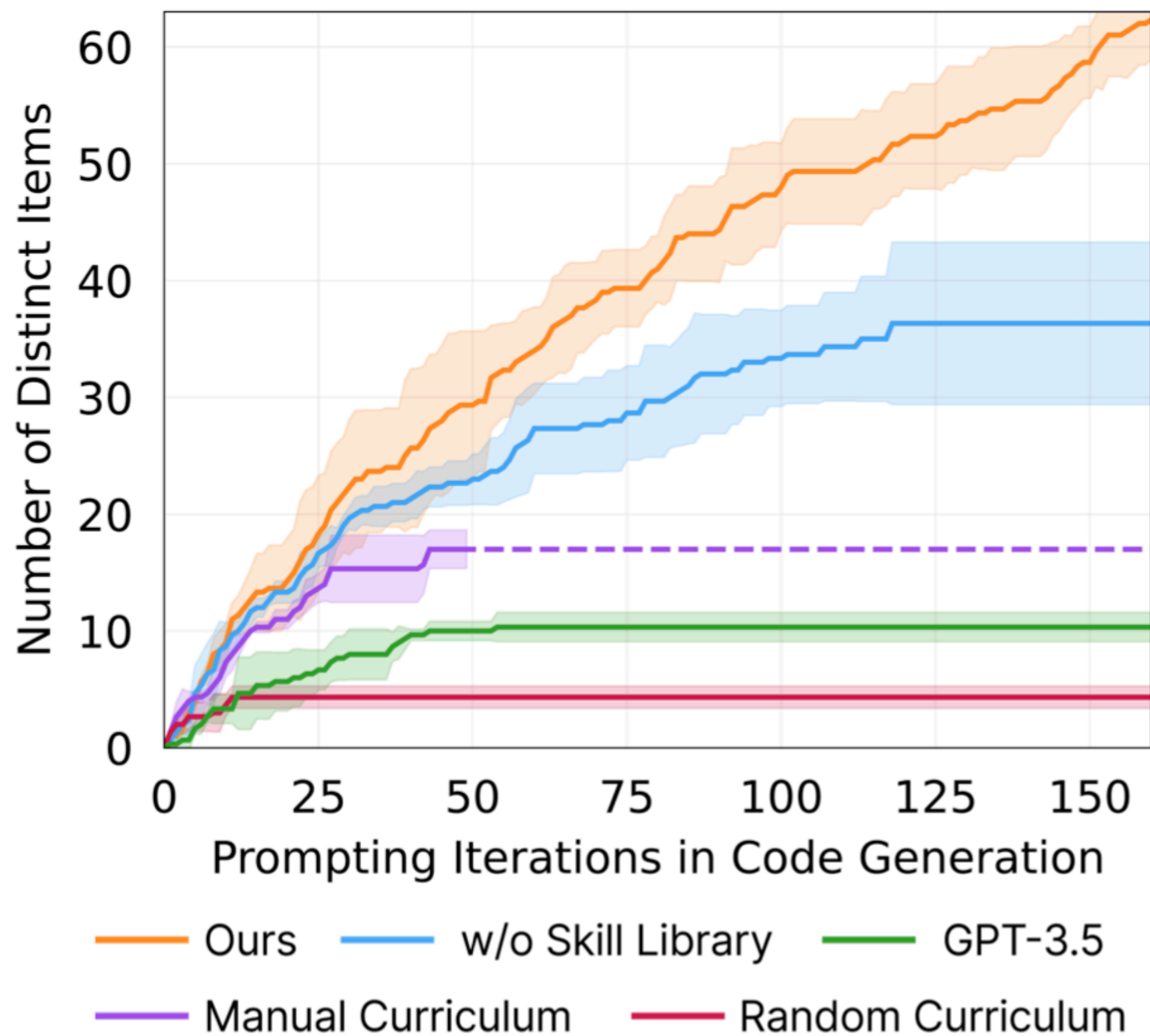
## Experiments

# Ablation Studies

---

- They removed each component of VOYAGER to study their impact on performance
- As a result, it turned out
  - Automatic curriculum is crucial for consistent progress
    - VOYAGER fails to discover new items
  - VOYAGER without skill library tends to stagnate in later stages
  - Self-verification is the most important among all the feedback types
  - GPT-4 significantly outperforms GPT-3.5 in code generation





Left: Ablation studies for the automatic curriculum, skill library, and GPT-4  
 Right: Ablation studies for the iterative prompting mechanism

## Conclusions

# Possibilities and limitations

---

- The proposed LLM-empowered agent could explore the world continuously, develop novel skills, and make new discoveries without human invention
- Limitations:
  - Cost
    - GPT-4 API costs 15x more than GPT-3.5
  - Inaccuracies
    - The iterative prompting mechanism sometimes fails to generate the correct skill
  - Hallucinations
    - The automatic curriculum occasionally proposes unachievable tasks

**Thank you for your attention**