

How to AI an AI for Almost Anything

Lecture 5 – Multi-Agent Evolution

Paul Liang

Assistant Professor

MIT Media Lab & MIT EECS



<https://pliang279.github.io>

ppliang@mit.edu

 [@pliang279](https://twitter.com/pliang279)



Announcements

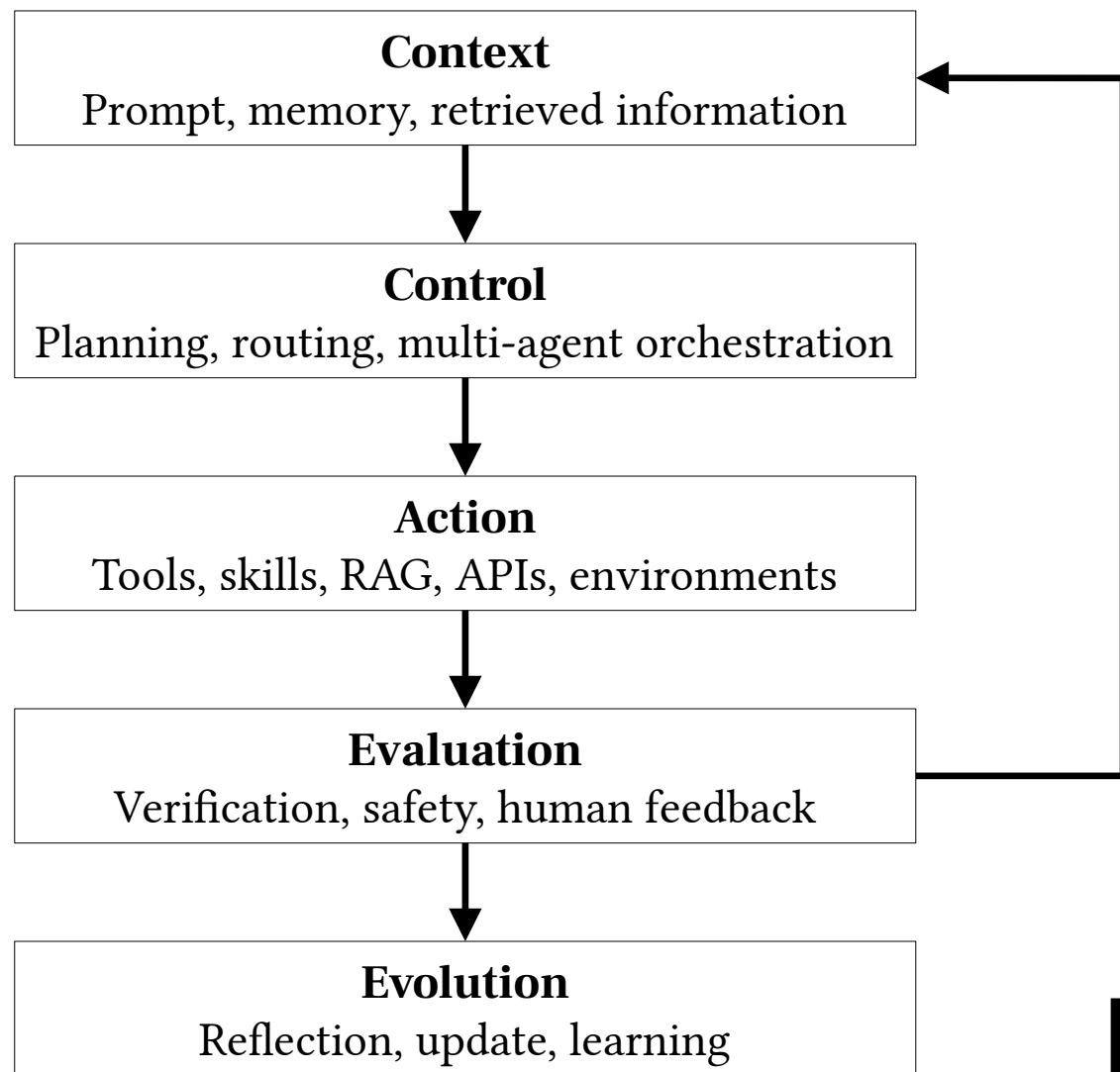
- Homework 1 extended, due Oct 15 11:59pm
- Work on projects, next checkpoint is midterm report + presentation Oct 28
 - Instructions posted on canvas
 - Processed datasets, ran baselines, done error analysis, better proposed idea
 - Bottom-up research. At the same time, continue top-down ideation from readings and literature review.

Today's Lecture

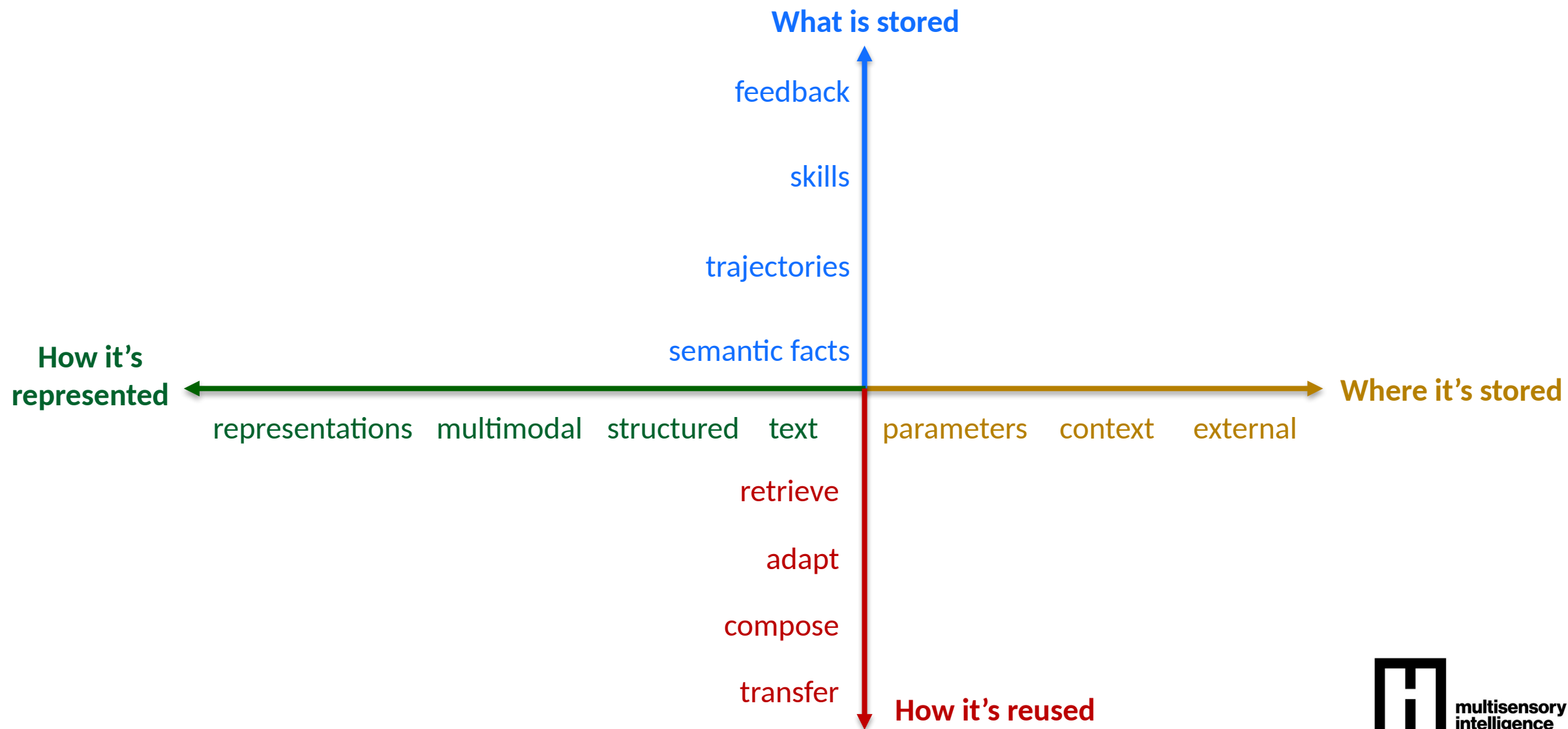
- Planning and task decomposition
- Multi-agent systems
- Multi-agent evolution

Recap of Harnesses

Harness: Code around the model that manages its context and execution flow to complete tasks



Recap of Memory



Recap: Harness Evolution

MEM1: treat memory as internal state in context autonomously updated by LLM

CLM: treat entire context as persistent file autonomously maintained and rewritten by LLM

Meta-Harness: treat entire harness as a file system autonomously modified by LLM

What's common about all these methods?

- Simple, no hand engineering, end-to-end
- New paradigm that generalizes what can be autonomously optimized (memory -> context -> harness)
- Scalable results for zero-shot, in-context prompting, fine-tuning

RLM: more structure in task decomposition and context management with recursive calls to sub-agents. but still highly scalable!

Planning and Task Decomposition

TASK

Buy the lab's new GPU workstation: compare configurations and prices across vendor sites, check compatibility against the cluster requirements on the internal wiki, and file the purchase requisition.

1 **ACTION** goto vendor-a.com → Workstations → spec sheet → GPU comparison ...

15 **ACTION** configure: 2× RTX 6000, 128 GB RAM, 850 W PSU

31 **OBSERVE** wiki: "racks accept 4U chassis; each slot with 1600 W"

context compacted · 128k → 14k tokens.
four vendors searched: only D fits, but has 20-week lead time

201 **ACTION** pick D? look for a fifth vendor? relax a requirement?

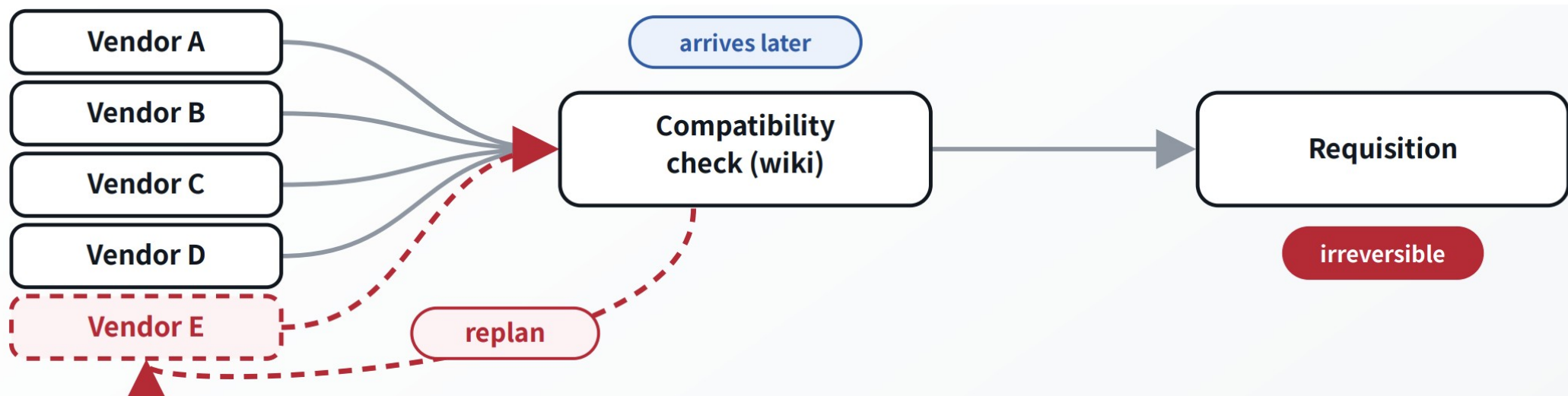
202 **ACTION** goto procurement portal → New requisition → Submit

1 Four independent searches run as one thread in one context.

2 The four sites it chose all come back and none is great.

3 It picks one anyway and files the requisition.

Planning and Task Decomposition



The task has parallel searches, dependencies, information that arrives later in the task, and a step that cannot be undone.

The graph itself also changes: when none of the four vendors fits, a fifth search has to be added.

Key questions:

1. When is planning useful and when can it hurt?
2. How to represent plans?
3. When is a plan made and when should it change?

Planning via Prompting

Standard Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain of Thought Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

Planning via Prompting

Enables modularity, specialized models, separates planner from executor.

Stage 1: Decompose Question into Subquestions

Q: It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The water slide closes in 15 minutes. How many times can she slide before it closes?

Language Model

A: To solve "How many times can she slide before it closes?", we need to first solve: "How long does each trip take?"

Stage 2: Sequentially Solve Subquestions

Subquestion 1

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.

Q: How long does each trip take?

Language Model

A: It takes Amy 4 minutes to climb and 1 minute to slide down. $4 + 1 = 5$. So each trip takes 5 minutes.

Append model answer to Subquestion 1

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.

Q: How long does each trip take?

A: It takes Amy 4 minutes to climb and 1 minute to slide down. $4 + 1 = 5$. So each trip takes 5 minutes.

Language Model

A: The water slide closes in 15 minutes. Each trip takes 5 minutes. So Amy can slide $15 \div 5 = 3$ times before it closes.

Subquestion 2

Q: How many times can she slide before it closes?

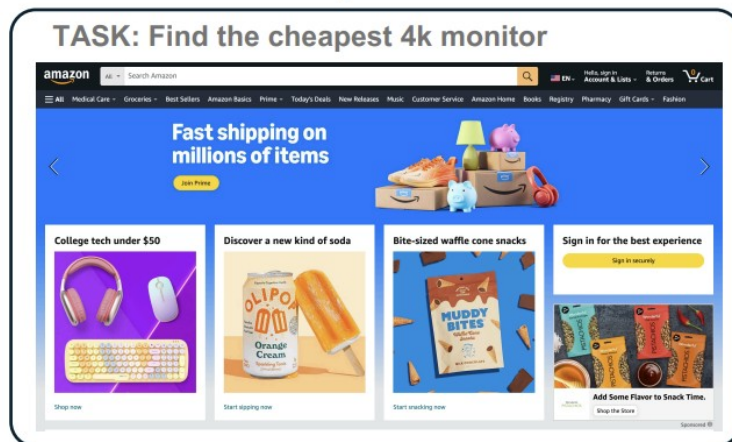
Prompt-based planning works in text-only, action-free, reversible scenarios.

If plans act on the world, changes the state of the agent, or cannot be reversed, then need to plan and re-plan.

Guidance and Orchestration Models

Planning model is larger and visually grounded, but bad at producing low-level actions. Separate small 7B execution model maps plans to coordinate actions.

Vision-Only Observation



Planning



User: Decide the next action for the task.
 Element Description: **The search bar at the top of the page.**
 Action: Type
 Value: 4k monitor

Grounding

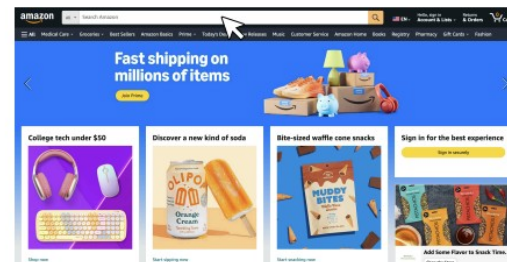


User: What are the pixel coordinates of the element corresponding to **“The search bar at the top of the page”** ?
 (556, 26)

Human-Like Operation

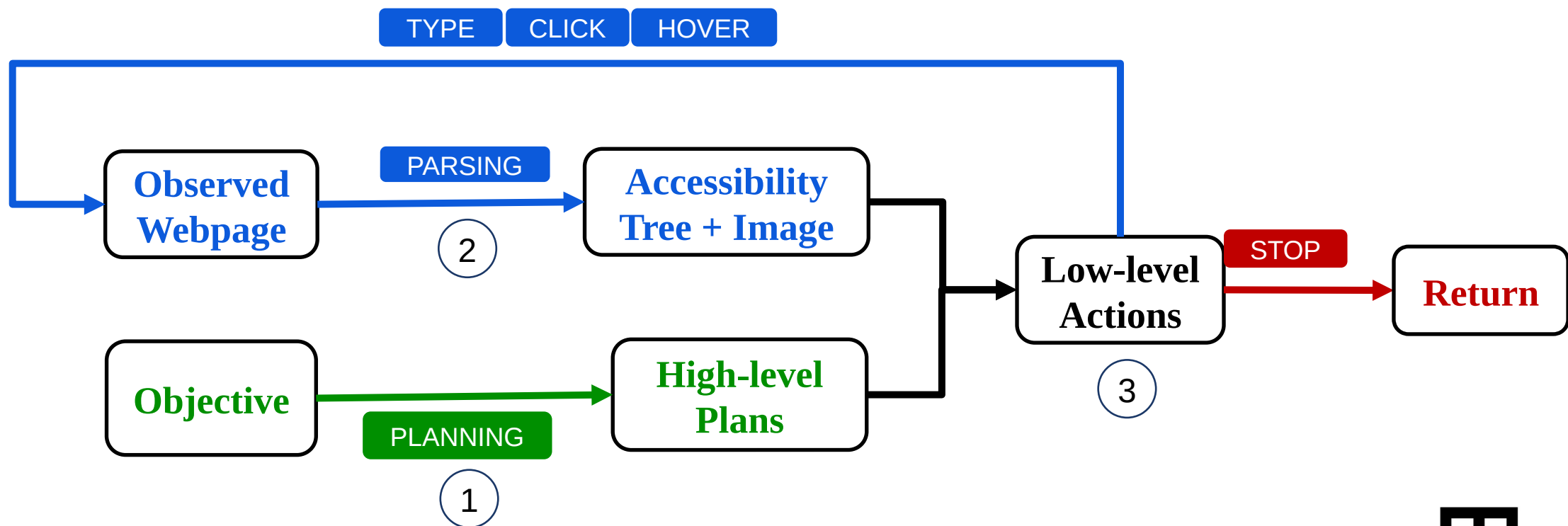


Click (556, 26)
 Type (“4k monitor”)



Planning in Agents

- General architecture of web agents:
 - High-level Reasoning
 - Observation Parsing
 - Low-level Action Generation



Planning with Human-in-the-loop

Asking humans for clarification when it is uncertain about generated plans

What is the price range of wireless earphone in the One Stop Market?

Zero-shot high-level plan

[Search] wireless earphone in the One Stop Market



[Find] the price range of wireless earphone in the One Stop Market

Too ambiguous

Few-shot high-level plan

[Search] wireless earphone in the One Stop Market



[Find] the price of the first wireless earphone



[Read] the price range of wireless earphones in the One Stop Market

Didn't sort

Few-shot high-level plan with human clarification

[Search] wireless earphone in the One Stop Market



[Sort] the price of the wireless earphones from low to high



[find] the lowest priced wireless earphone



[sort] the price of wireless earphones from high to low



[find] the highest priced wireless earphone

Subsequent model generations become correct after clarification

Ground truth high-level plan

[Search] wireless earphones in the One Stop Market.



[Sort] the price of the wireless earphones from low to high



[Find] the first item



[Sort] the price of the wireless earphones from high to low



[Find] the first item

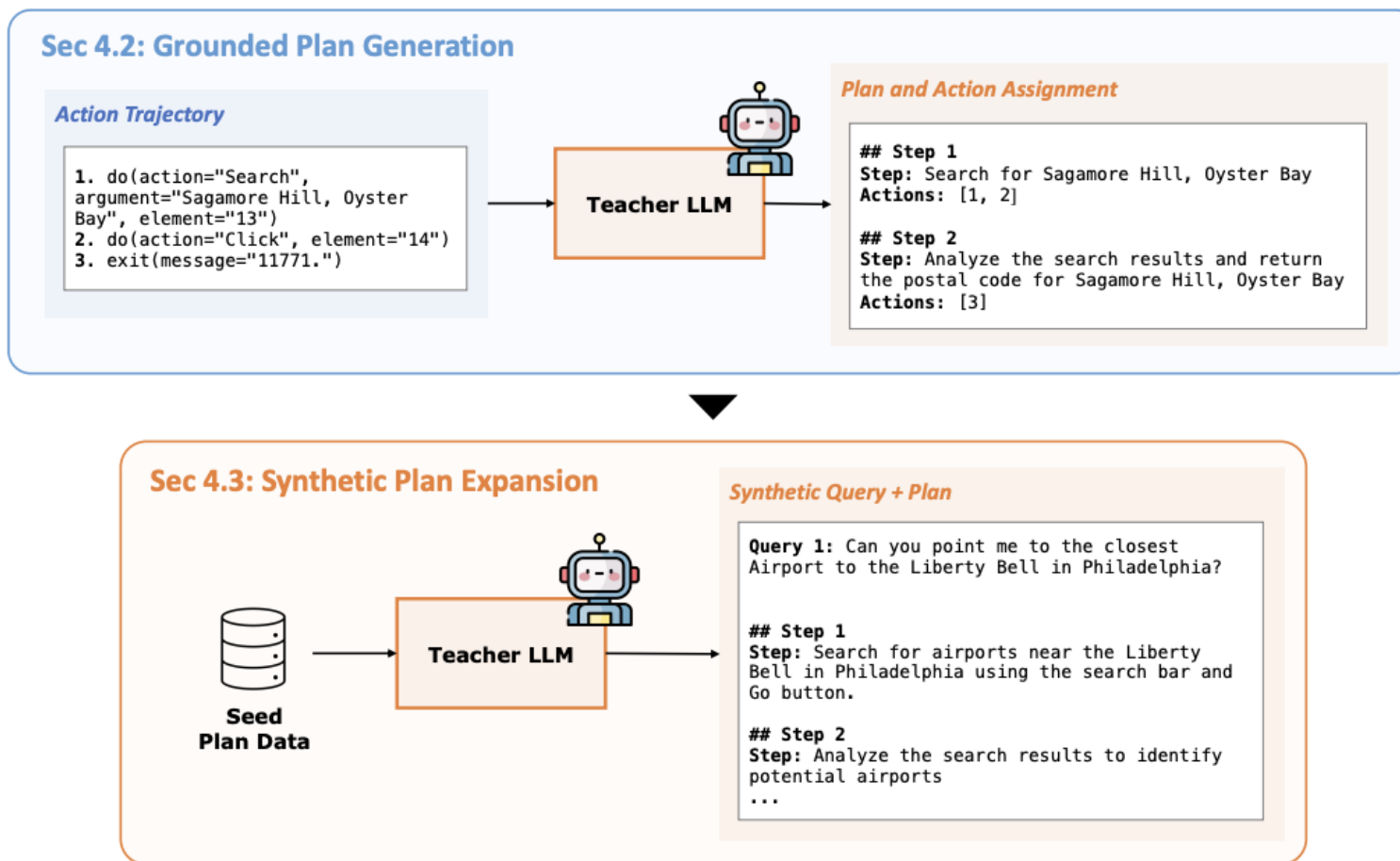


[Identify] the price range of wireless earphones in the One Stop Market



Training the Planner

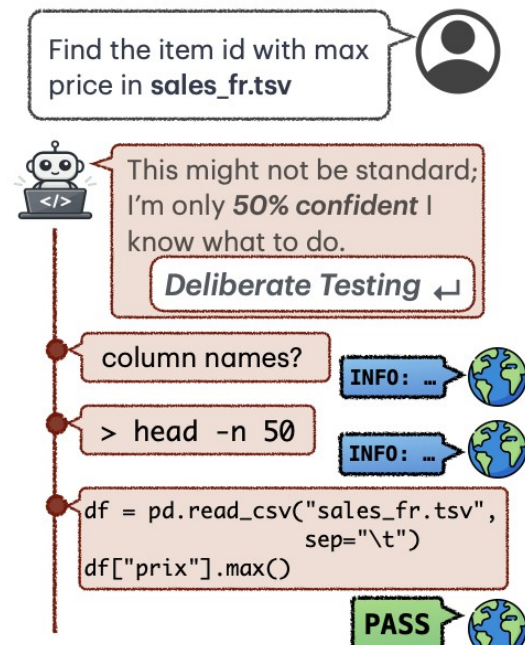
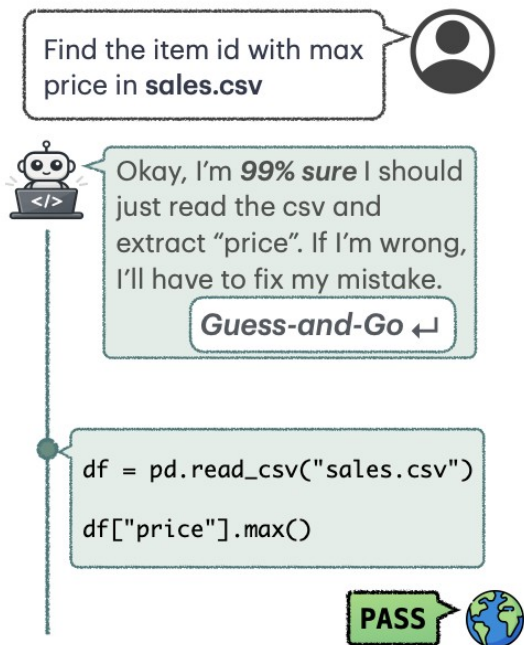
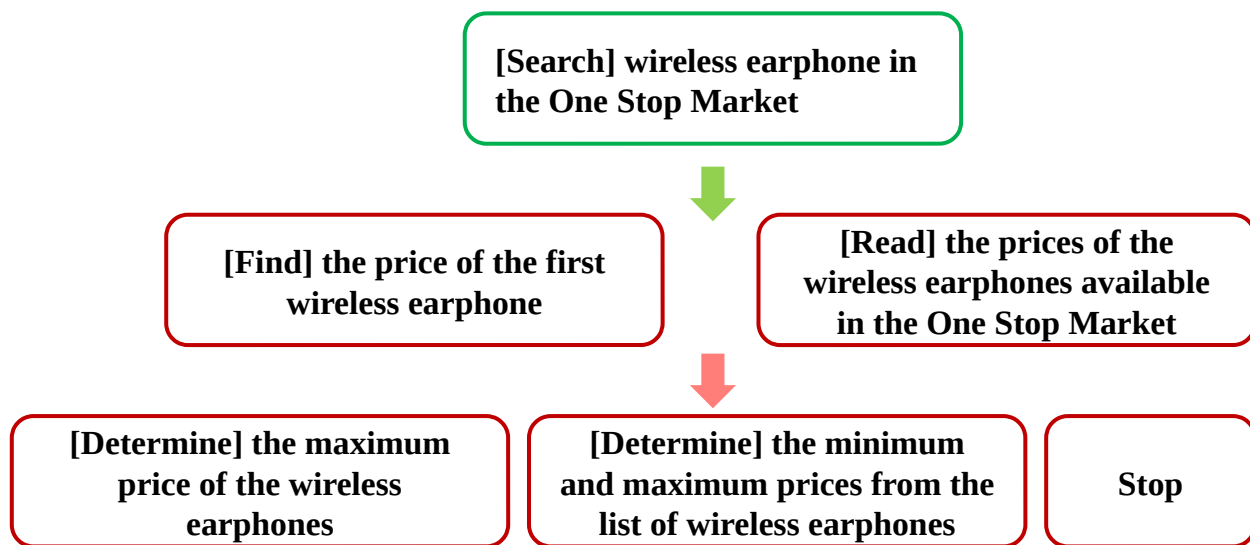
Synthetic plan generation by teacher model + supervised fine-tuning of planner model and execution model.



Calibrated Planning

Sampling high-level plans generated by the model multiple times to estimate uncertainty

High-level Planning

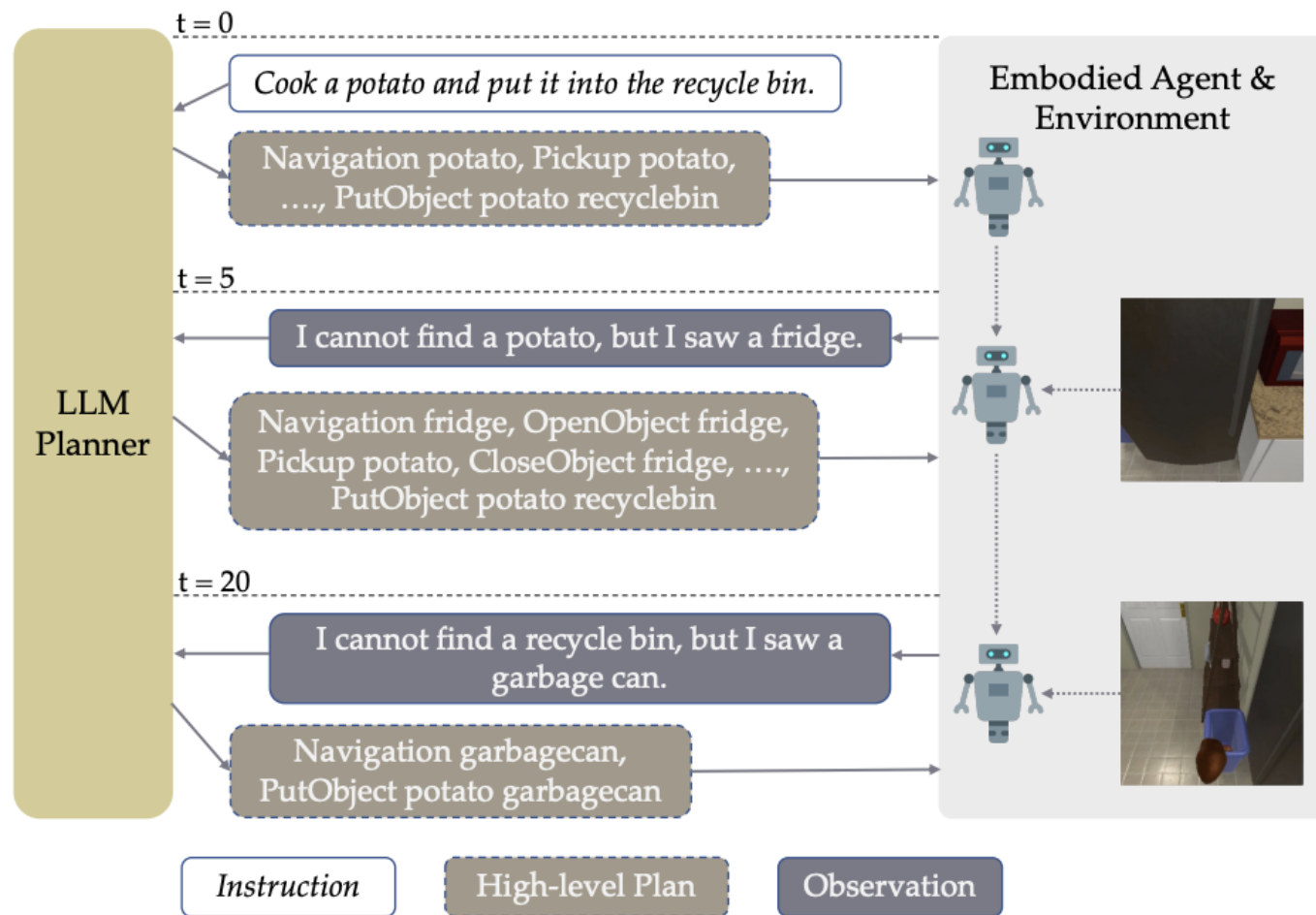


The optimal strategy is a function of model's *success probability* and *environment costs* of different actions. **Can LLMs behave optimally?**

Replanning

Planning, acting in embodied environment, observing feedback, plan again in response to new information or failures.

Main challenge: requires taking safe & non-reversible actions.



Test-Time Planning and Interaction

Not just thinking, but actually executing the action, seeing feedback, and re-trying.

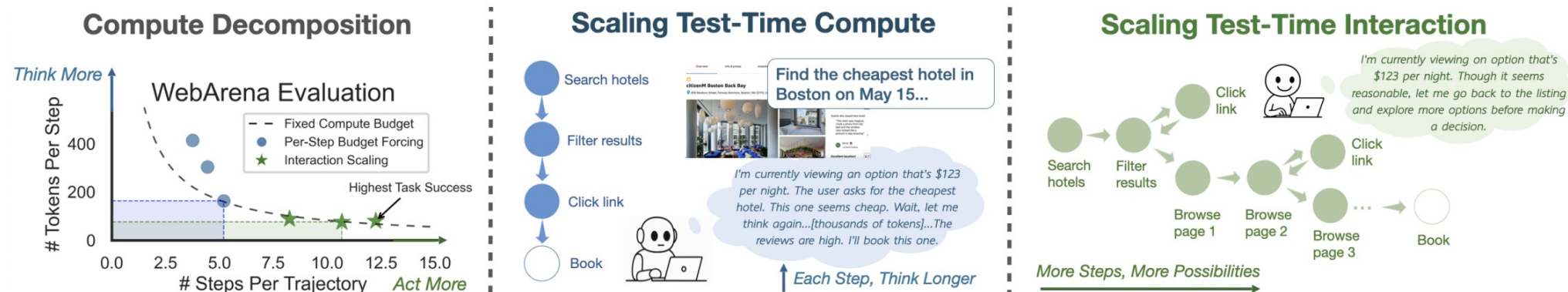
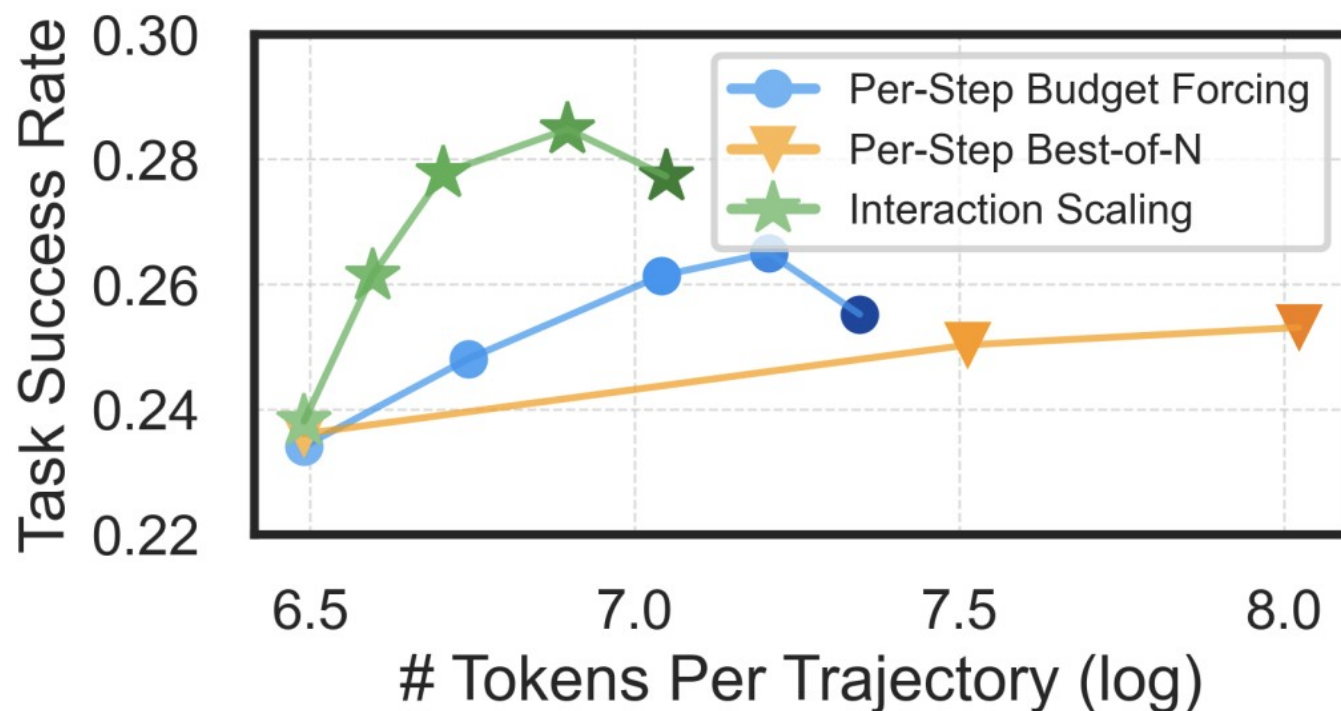


Figure 1: We propose a new-axis of test-time scaling for agents: scaling the number of interaction steps. Unlike traditional methods that emphasize longer reasoning per step, we show that acting more helps gain new information from the environment and improve task performance (detailed results of the left plot in Section 4.2).

Test-Time Planning and Interaction

Not just thinking, but actually executing the action, seeing feedback, and re-trying.



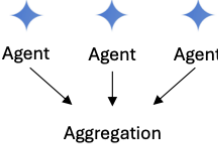
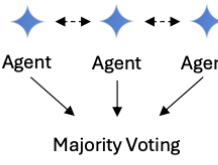
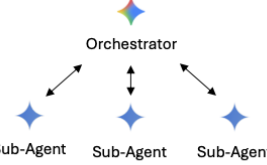
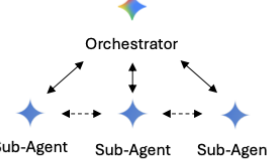
Putting Everything Together: Multi-Agent Systems

Planner/orchestration agents + execution agents.

Decentralized agents with aggregation function.

Multimodal agents each processing different modalities.

Table 2 | Architectural comparison of agent methods with objective complexity metrics. Computational complexity measured in terms of LLM calls, coordination overhead, and parallelization potential.

Characteristic	SAS	MAS (Independent)	MAS (Decentralized)	MAS (Centralized)	MAS (Hybrid)
Interaction Type	 Agent	 Aggregation	 Majority Voting	 Sub-Agent Sub-Agent Sub-Agent	 Sub-Agent Sub-Agent Sub-Agent
LLM Calls	$O(k)$	$O(nk) + O(1)$	$O(dnk) + O(1)$	$O(rnk) + O(r)$	$O(rnk) + O(r) + O(p)$
Sequential Depth	k	k	d	r	r
Comm. Overhead	0	1	$d \cdot n$	$r \cdot n$	$r \cdot n + p \cdot m$
Parallelization Factor	1	n	n	n	n
Memory Complexity	$O(k)$	$O(n \cdot k)$	$O(d \cdot n \cdot k)$	$O(r \cdot n \cdot k)$	$O((r + p) \cdot n \cdot k)$
Coordination	Sequential	Parallel + Synthesis	Sequential Debate	Hierarchical	Hierarchical + Peer
Consensus	-	Synthesis	Debate	Orchestrator	Orchestrator

Decentralized Multi-Agent Systems

Multiple instances of a language model propose answers, critique one another, and debate over several rounds before reaching a final answer.

Question: Carla is downloading a 200 GB file. Normally she can download 2 GB/minute, but 40% of the way through the download, Windows forces a restart to install updates, which takes 20 minutes. Then Carla has to restart the download from the beginning. How long does it take to download the file?

Round 1

ChatGPT: Before the restart, Carla was able to download 40% of the file, which is $0.4 * 200 \text{ GB} = 80 \text{ GB}$. At a rate of 2 GB/minute, it would take her: $\text{time} = 80 \text{ GB} \div 2 \text{ GB/minute} = 40 \text{ minutes}$.

Because of the restart, Carla had to start over. This means she had to download the entire 200 GB file. The time it takes to download the remaining 120 GB is: $\text{time} = 120 \text{ GB} \div 2 \text{ GB/minute} = 60 \text{ minutes}$.

But Carla also had to wait 20 minutes for the Windows update to install, so the total time it took to download the file is: $\text{total time} = 40 \text{ minutes} + 60 \text{ minutes} + 20 \text{ minutes} = \mathbf{120 \text{ minutes}}$. ❌

Bard: Carla is downloading a 200 GB file at a speed of 2 GB/minute. This means that it will take her $200 / 2 = 100 \text{ minutes}$ to download the file.

However, Windows forces a restart to install updates, which takes 20 minutes. This means that Carla will have to restart the download from the beginning. Since Carla has already downloaded 40% of the file, she only needs to download $200 - 40 = 160 \text{ GB}$. At a speed of 2 GB/minute, it will take Carla $160 / 2 = 80 \text{ minutes}$ to download the file.

Adding the time for the Windows restart, it will take Carla $80 + 20 = \mathbf{100 \text{ minutes}}$ to download the file. ❌

Round 2

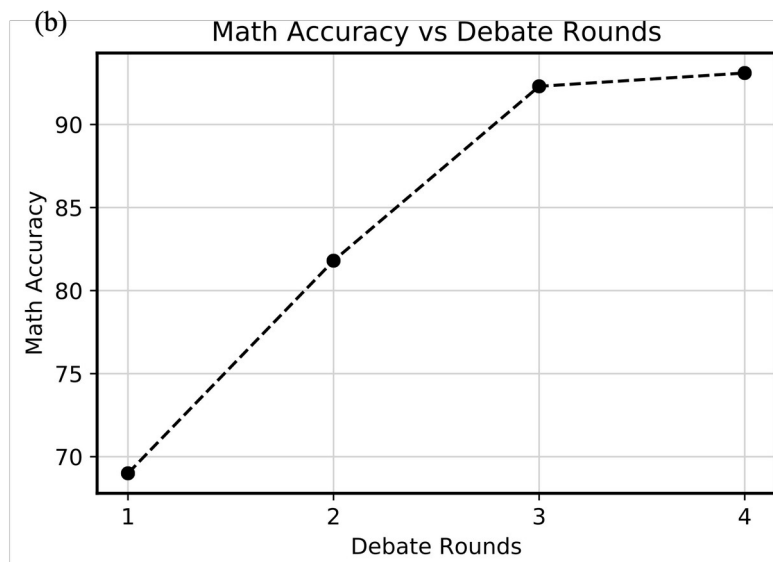
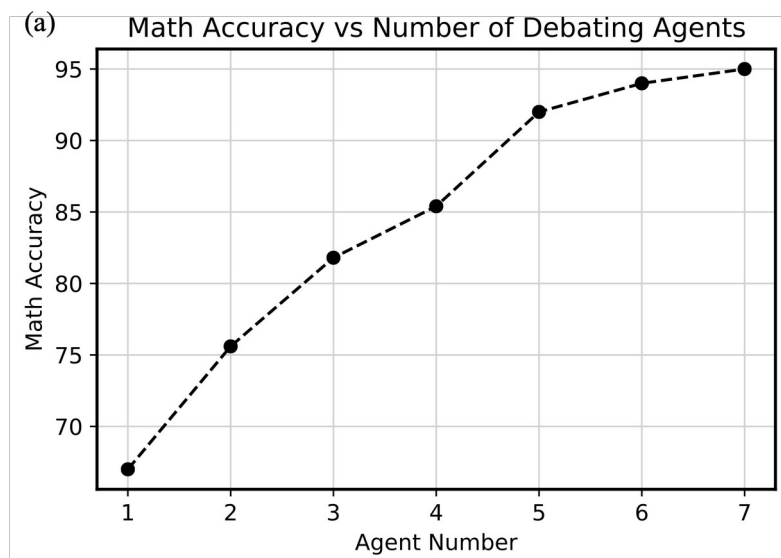
ChatGPT : Based on the calculations provided by the other agent, the correct answer is: Before the restart, Carla was able to download 40% of the file, which is $0.4 * 200 \text{ GB} = 80 \text{ GB}$. At a rate of 2 GB/minute, it would take her: $\text{time} = 80 \text{ GB} \div 2 \text{ GB/minute} = 40 \text{ minutes}$

However, because of the restart, Carla had to start over. This means she had to download the entire 200 GB file. The time it takes to download the entire 200 GB file is: $\text{time} = 200 \text{ GB} \div 2 \text{ GB/minute} = 100 \text{ minutes}$

And since Carla had to wait 20 minutes for the Windows update to install, the total time it took to download the file is: $\text{total time} = 40 \text{ minutes} + 100 \text{ minutes} + 20 \text{ minutes} = \mathbf{160 \text{ minutes}}$. ✅

Decentralized Multi-Agent Systems

Multiple instances of a language model propose answers, critique one another, and debate over several rounds before reaching a final answer.



Centralized Multi-Agent Systems

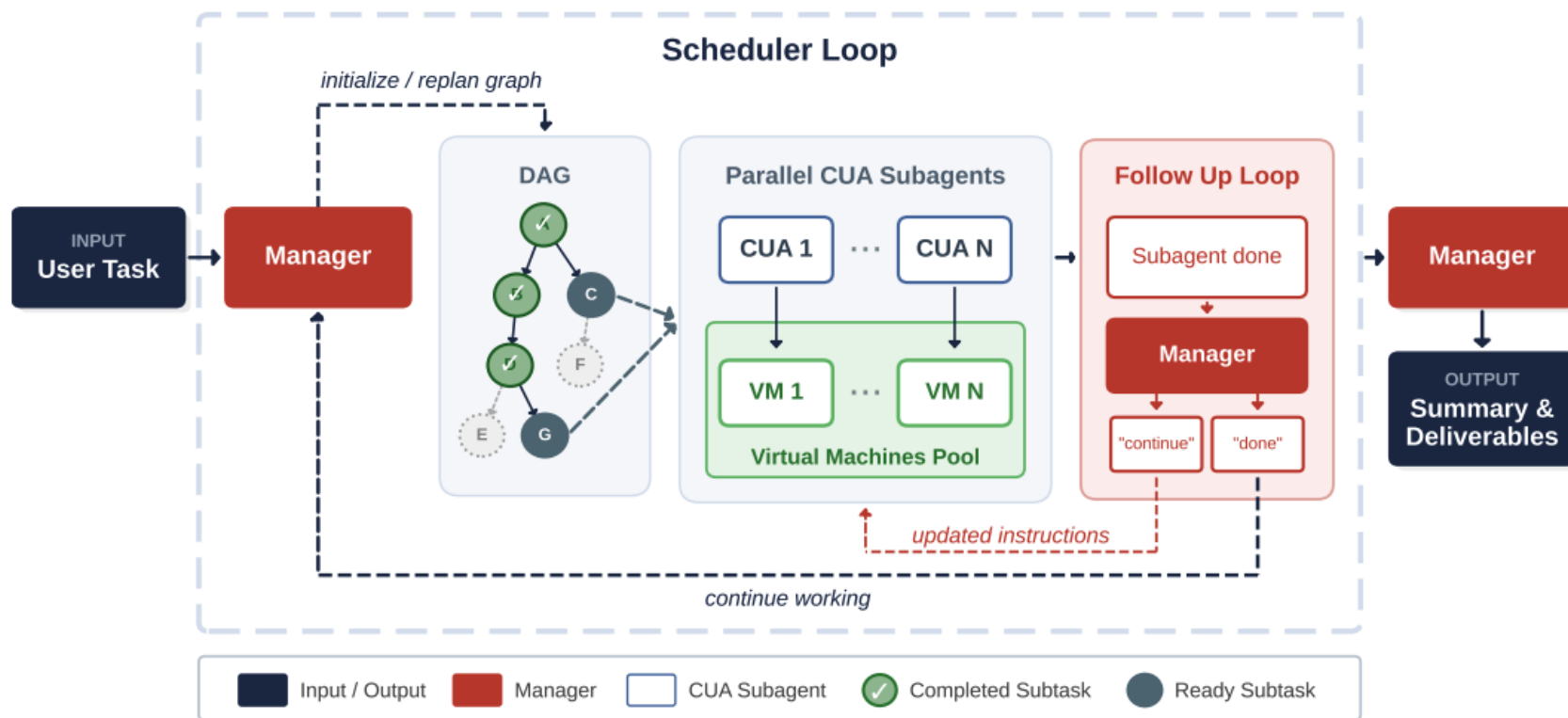
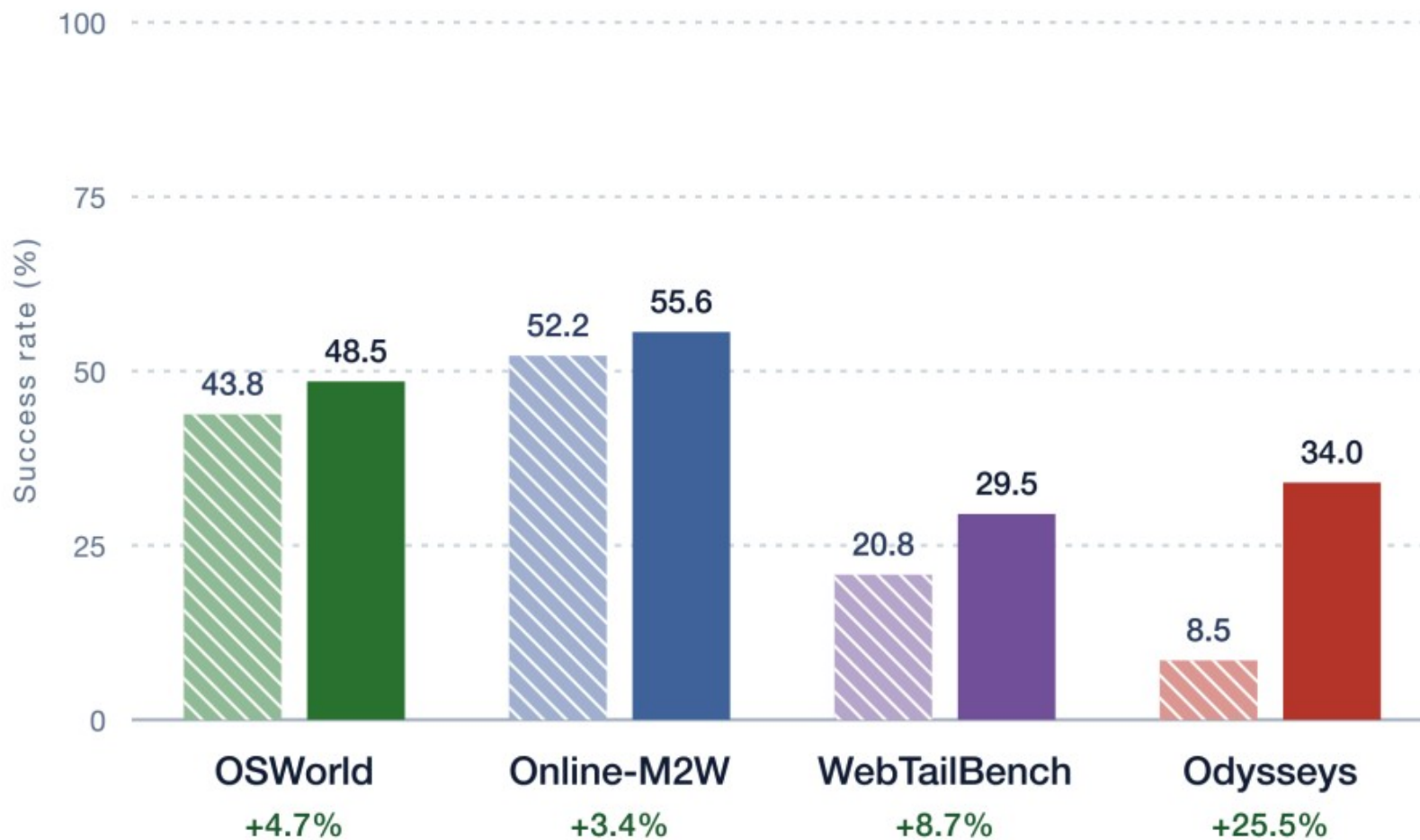


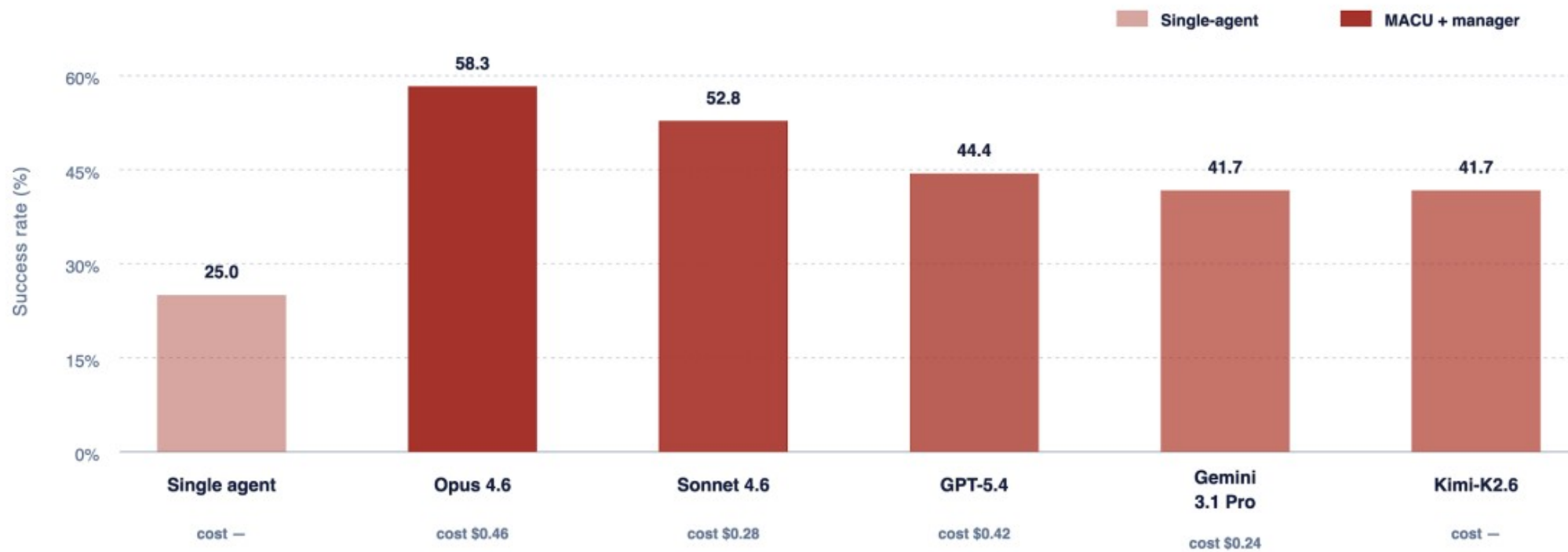
Figure 1: Overview of the setup for multi-agent computer use (MACU). At each iteration, ready subtasks (grey) are launched and handled in parallel by CUA subagents. The manager replans and updates the DAG based on information it receives from completed subagents.

Centralized Multi-Agent Systems



Importance of Strong Orchestrator

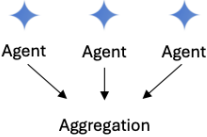
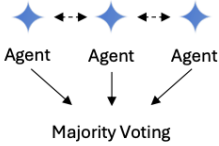
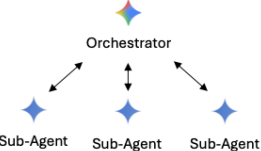
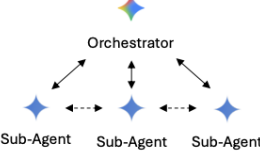
OSWORLD ABLATION · 36 TASKS · QWEN3.5-4B WORKER



When Does Multi-Agent Work?

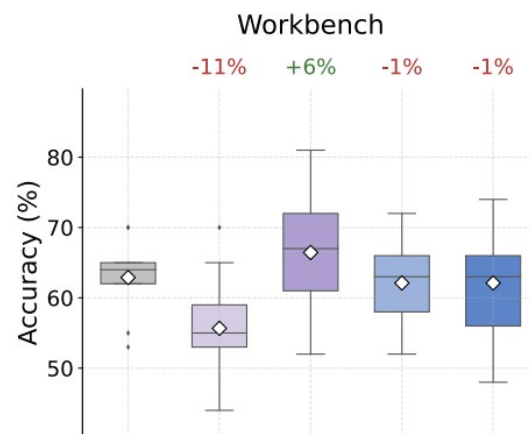
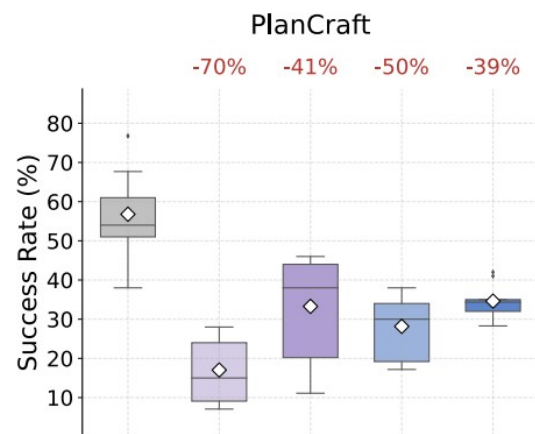
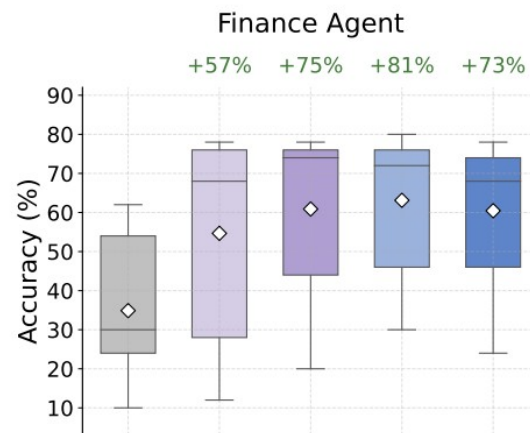
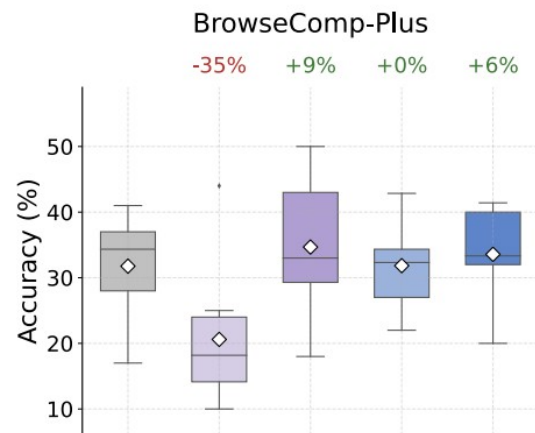
More agents does not automatically improve performance: scaling depends on task structure and multi-agent topology.

Table 2 | Architectural comparison of agent methods with objective complexity metrics. Computational complexity measured in terms of LLM calls, coordination overhead, and parallelization potential.

Characteristic	SAS	MAS (Independent)	MAS (Decentralized)	MAS (Centralized)	MAS (Hybrid)
Interaction Type	 Agent	 Aggregation	 Majority Voting	 Sub-Agent Sub-Agent Sub-Agent	 Sub-Agent Sub-Agent Sub-Agent
LLM Calls	$O(k)$	$O(nk) + O(1)$	$O(dnk) + O(1)$	$O(rnk) + O(r)$	$O(rnk) + O(r) + O(p)$
Sequential Depth	k	k	d	r	r
Comm. Overhead	0	1	$d \cdot n$	$r \cdot n$	$r \cdot n + p \cdot m$
Parallelization Factor	1	n	n	n	n
Memory Complexity	$O(k)$	$O(n \cdot k)$	$O(d \cdot n \cdot k)$	$O(r \cdot n \cdot k)$	$O((r + p) \cdot n \cdot k)$
Coordination	Sequential	Parallel + Synthesis	Sequential Debate	Hierarchical	Hierarchical + Peer
Consensus	-	Synthesis	Debate	Orchestrator	Orchestrator

When Does Multi-Agent Work?

Finding 1: **Multi-agent systems only help for certain types of tasks.**



Ex 1. Finance agent (MAS wins +81%)

-> Task naturally decomposes
(revenue/cost/market)

Ex 2. PlanCraft (MAS loses -70%)

-> Sequential dependency
(action 1 affects action 2)

-> Splitting tasks hurts rather than helps

Guidelines:

Parallelizable tasks -> centralized

Exploration needed -> decentralized

Sequential dependency -> single-agent

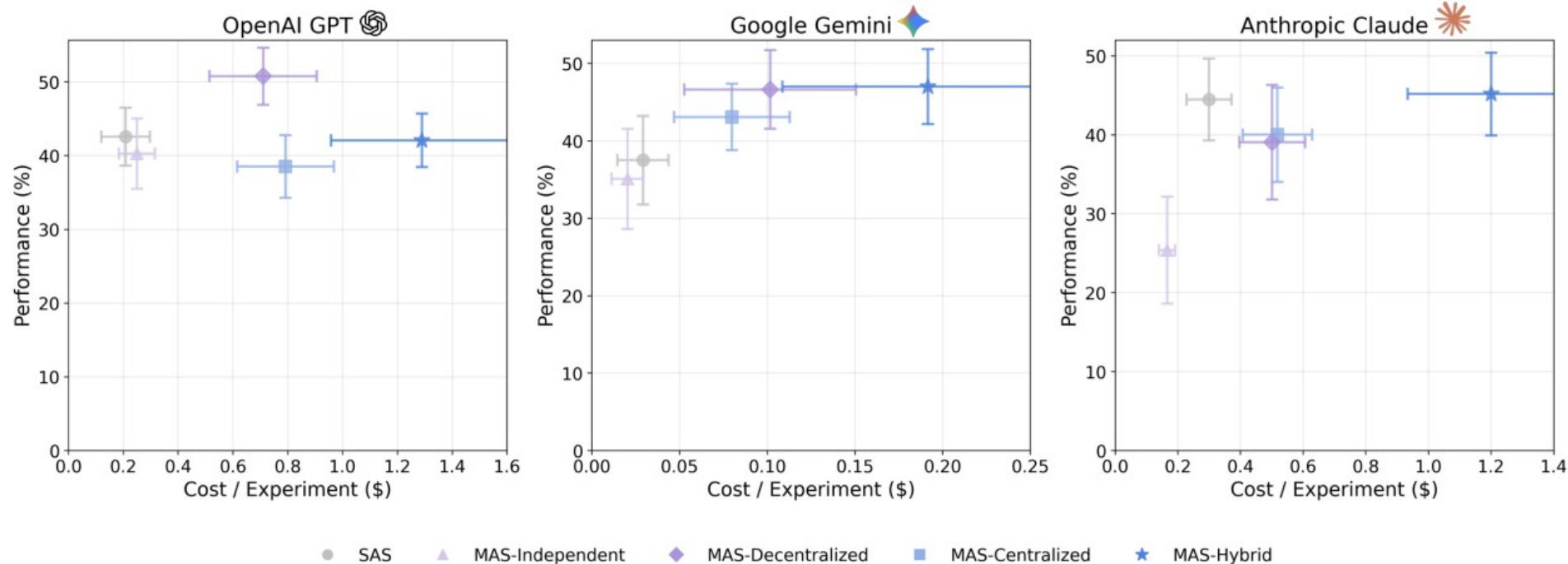
When Does Multi-Agent Work?

Finding 2: **Different base models exhibit different cost-performance tradeoffs.**

OpenAI: performance plateau despite increasing cost.

Google: higher cost, higher performance.

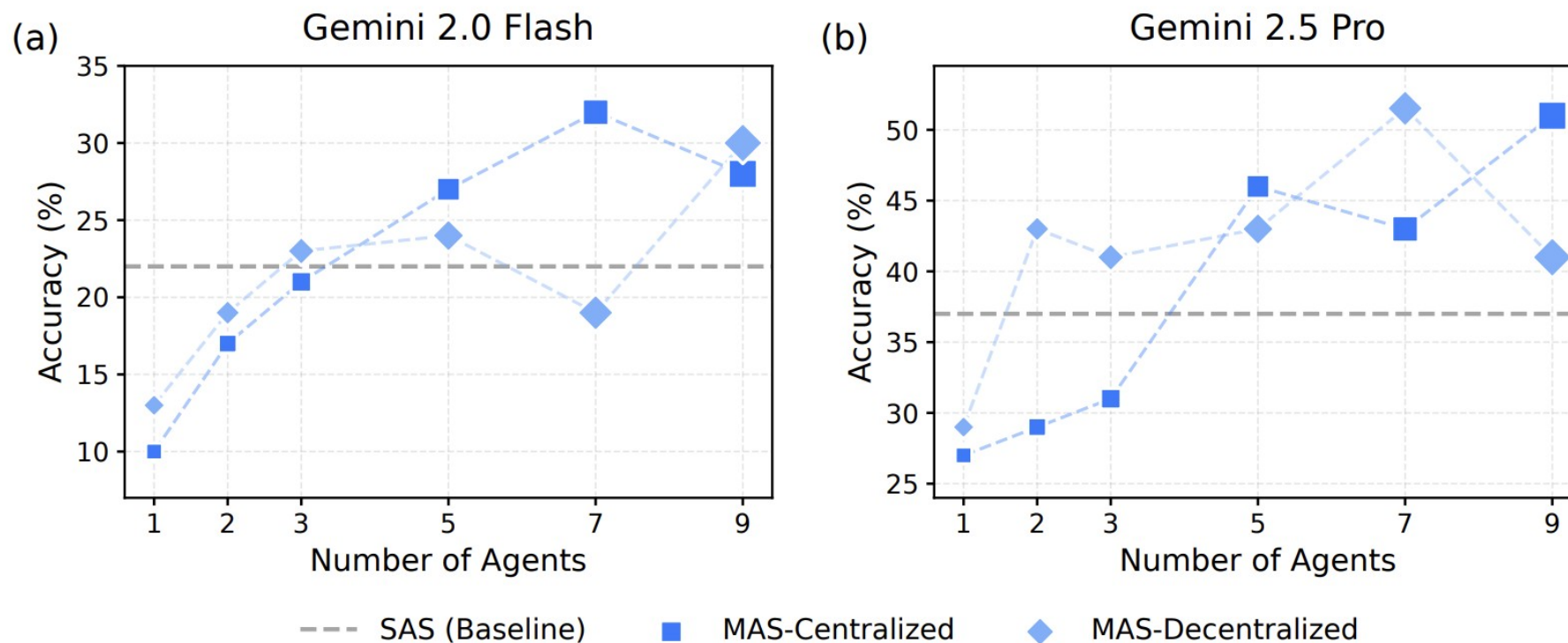
Anthropic: MAS can hurt performance.



When Does Multi-Agent Work?

Finding 3: **Increasing number of agents does not always improve performance.**

Tool-heavy tasks (e.g., 16-tool workflows) suffer from multi-agent coordination overhead, with penalties compounding as environment complexity increases.

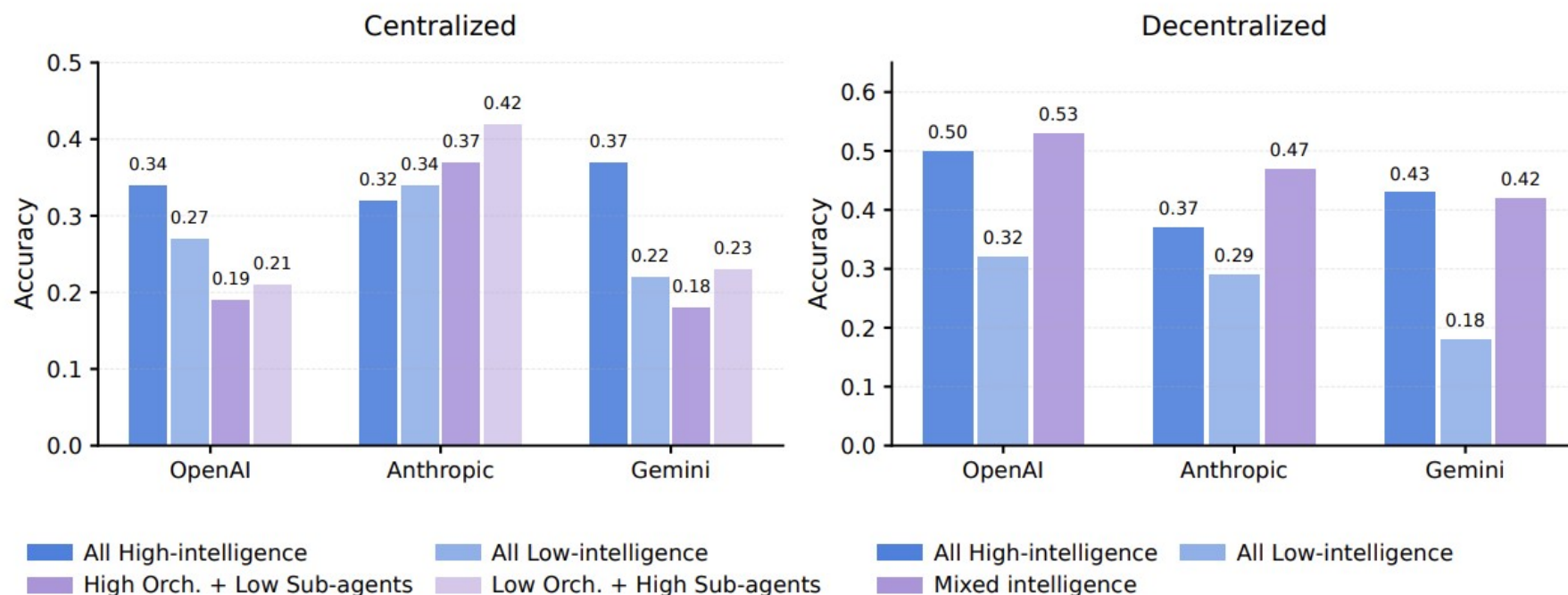


When Does Multi-Agent Work?

Finding 4: **Mixtures of heterogeneous agents can outperform better base agents.**

We don't need expensive models everywhere. If single agent baseline > 45%, MAS hurts.

Cheap orchestrator + expensive execution sub-agents seems to work well.



When Does Multi-Agent Work?

Finding 5: **Agent-system performance can be predicted.**

A scaling law using models, tasks, MAS setup as parameters predicts the best architecture 87% of the time. From heuristics ("try a setup") to more principled science of agent design.

The complete functional form is:

$$\begin{aligned}
 P = & \beta_0 + \beta_1(I - \bar{I}) + \beta_2(I - \bar{I})^2 + \beta_3 \log(1 + T) + \beta_4 \log(1 + n_a) \\
 & + \beta_5 \log(1 + O\%) + \beta_6 c + \beta_7 R + \beta_8 E_c + \beta_9 \log(1 + A_e^{\text{trace}}) \\
 & + \beta_{10} P_{SA} + \beta_{11}(I \times E_c) + \beta_{12}(A_e^{\text{trace}} \times P_{SA}) \\
 & + \beta_{13}(O\% \times T) + \beta_{14}(R \times n_a) + \beta_{15}(c \times I) \\
 & + \beta_{16}(E_c \times T) + \beta_{17}(P_{SA} \times \log(1 + n_a)) \\
 & + \beta_{18}(I \times \log(1 + T)) + \beta_{19}(A_e^{\text{trace}} \times T) + \varepsilon,
 \end{aligned}$$

■ $p < 0.001$ ■ Significant ($p < 0.05$) ■ Non-significant ($p > 0.05$)

- Model capability (I)
- System configuration (n_a)
- Task properties (T, P_{SA})
- Coordination metrics ($E_c, O\%, A_e, c, R$)

Base model intelligence.

Num agents.

Tool count, single agent baseline.

Efficiency, overhead, trace error,
message density, redundancy.

Multi-Agent Evolution

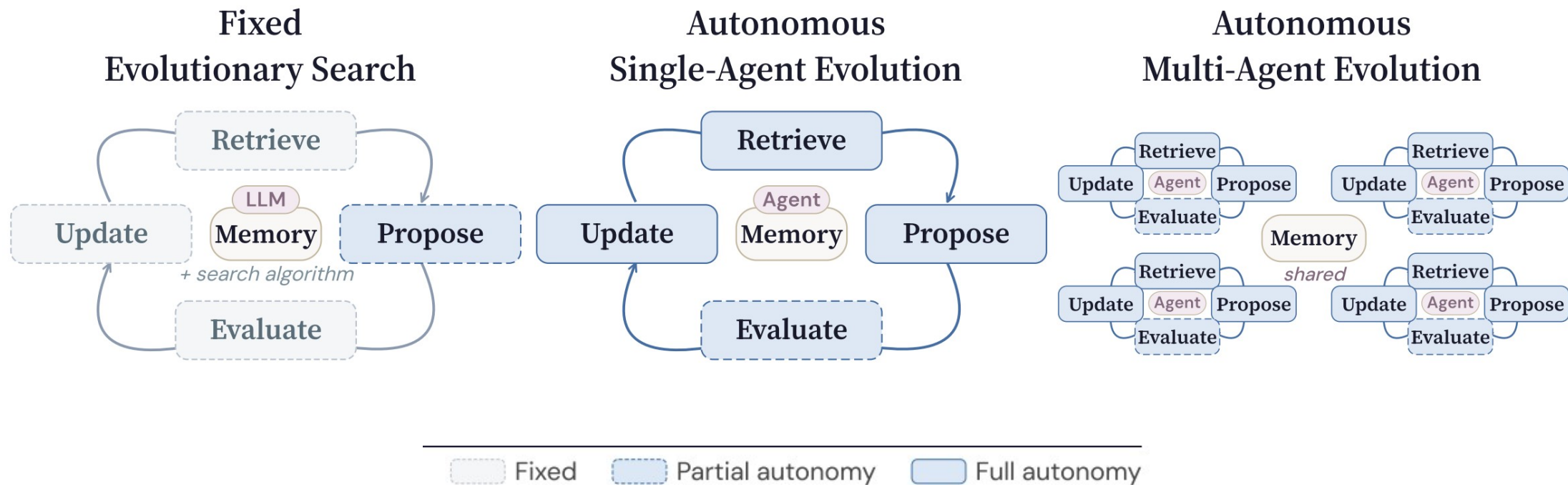
Given multi-agent systems don't always work and depend on critical hyperparameters (e.g., task, base models, topologies), there have been recent attempts at automating the multi-agent design space, instead of relying on hand engineering.

i.e., Multi-agent evolution/optimization.

What can be evolved?

- Individual models and agents (i.e., models, context, harness)
- Multi-agent system (i.e., agent roles, communication, topologies)
- Shared information (i.e., common memory, tools, skills)

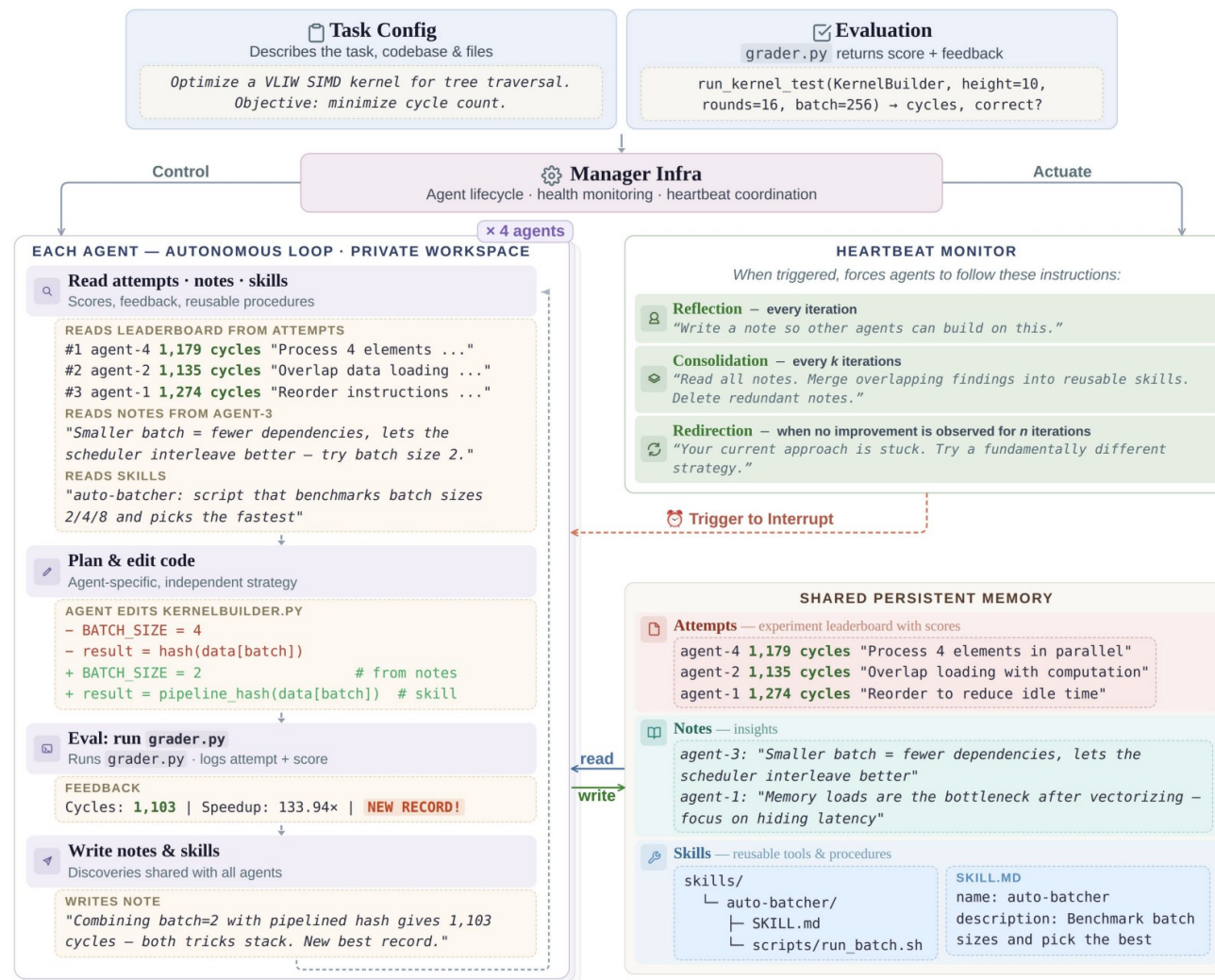
CORAL for Multi-Agent Evolution



CORAL: Multi-Agent Evolution <https://github.com/Human-Agent-Society/CORAL>

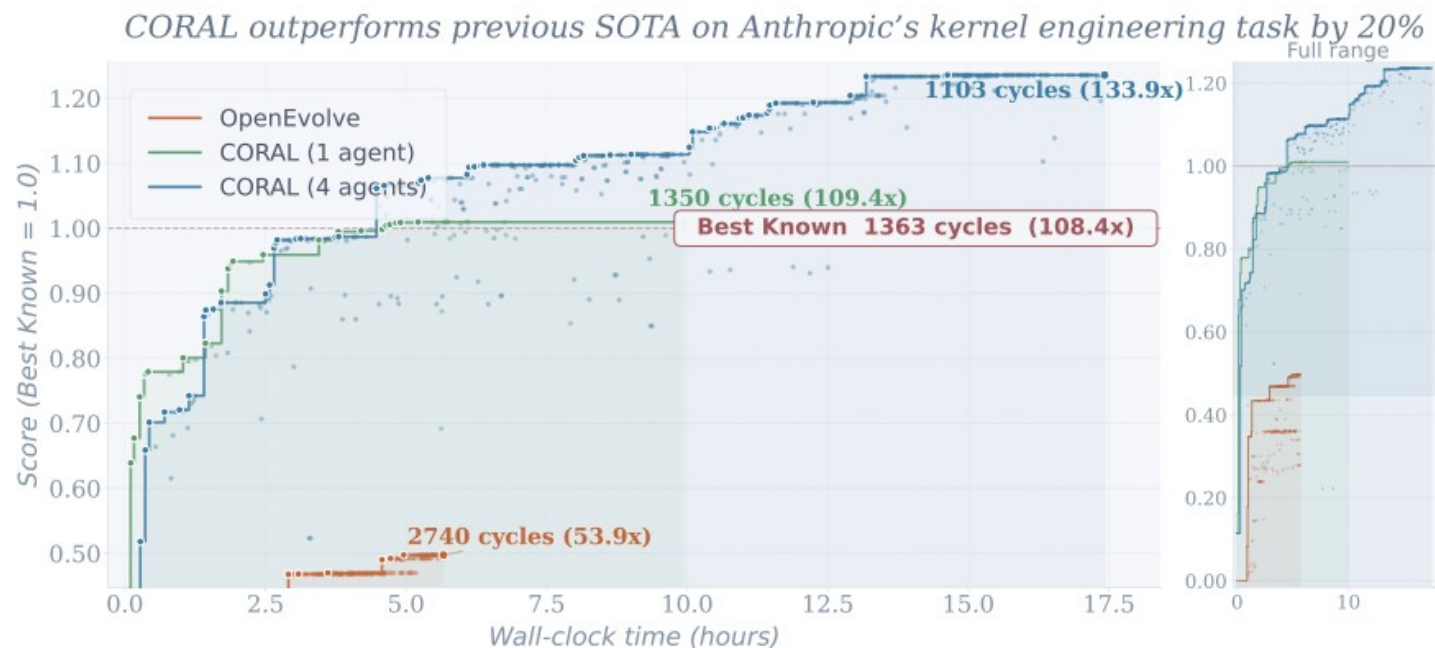
Key ideas:

- Multi-agent specialization
- Increased autonomy
- Shared memory
- Heartbeat mechanism



Kernel Engineering

<https://github.com/Human-Agent-Society/CORAL>



Operations research

Systems engineering

Scientific discovery

	Model	Evals ↓	Best Cycles ↓	Speedup ↑	Imp. Rate ↑
• Best Known	Anthropic	—	1363	108.4x	—
• OpenEvolve	Opus 4.6	363	2740	53.9x	3% (12/363)
• CORAL (1 agent)	Opus 4.6	<u>56</u>	<u>1350</u>	<u>109.4x</u>	<u>43% (24/56)</u>
• CORAL (4 agents)	Opus 4.6	596	<u>1103</u>	<u>133.9x</u>	9% (54/596)

The task is to optimize a GPU kernel to minimize execution cycles from a 147,734-cycle baseline. Score = baseline / cycles, normalized so the previous best known result (1,363 cycles) equals 1.0. All methods use Claude Opus 4.6.

More Results

<https://github.com/Human-Agent-Society/CORAL>

New SOTA on 8 tasks spanning math, algorithms, and system optimization

	Task	Final Score					Impr. Rate (%)				# Evals			
		SOTA	OE	SE	EvoX	CORAL	OE	SE	EvoX	CORAL	OE	SE	EvoX	CORAL
Math	Circle-Pack. ↑	2.6359	2.6293	2.6001	2.6320	2.6360	7.0	19.4	27.1	100.0	100	62	48	11
	Signal Proc. ↑	0.7429	0.6420	0.8171	0.7306	0.8229	11.8	8.6	21.1	30.3	85	58	71	56
	Erdős Over. ↓	0.38088	0.38188	0.38156	0.38125	0.38089	9.5	15.7	27.8	36.8	84	89	36	19
	MMD-16-2 ↓	12.89	12.92	12.89	12.96	12.89	8.2	9.8	33.3	83.3	97	82	18	6
	MMD-14-3 ↓	4.16	4.21	4.46	4.46	4.16	25.0	9.4	24.5	75.0	12	96	53	8
	3rd-Autocorr. ↓	1.4557	1.4731	1.4812	1.5552	1.4557	6.2	32.4	12.0	60.0	97	37	75	5
System	EPLB ↑	0.145	0.127	0.129	0.146	0.149	8.3	12.6	10.0	78.9	60	87	100	19
	PRISM ↑	26.26	26.26	26.26	26.26	26.26	31.6	18.8	25.0	100.0	19	32	16	3
	LLM-SQL ↑	0.730	0.716	0.724	0.726	0.731	6.7	23.8	6.0	53.3	100	21	83	15
	Txn Sched. ↑	4348	3774	3802	3984	4566	10.1	16.0	16.0	27.3	99	25	94	22
	Cloudcast ↓	632.7	627.2	627.8	623.5	618.4	7.2	20.8	12.3	33.3	69	24	65	9

More Results

<https://github.com/Human-Agent-Society/CORAL>

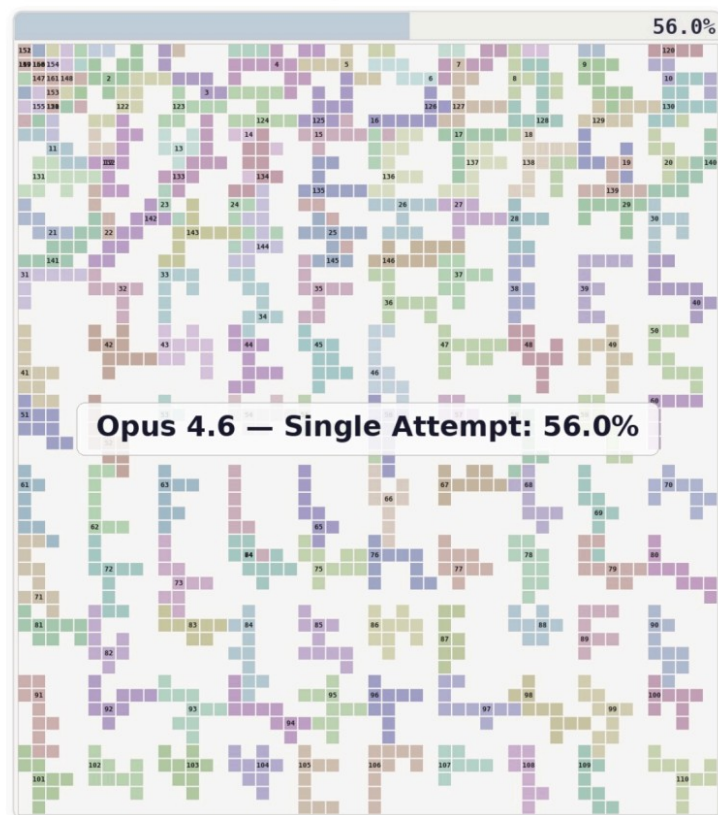
Polyomino Packing

Pack all polyominoes as tightly as possible into the grid. Minimize unused area.



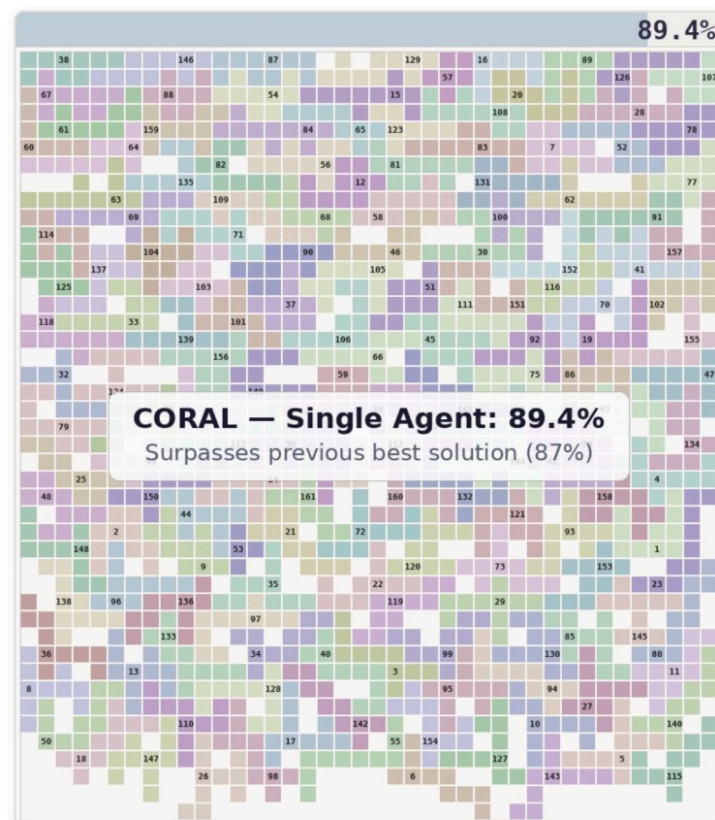
Opus 4.6

Single attempt



CORAL

Autonomous Self-Evolution



Multi-Agent Improves Over Single Agent

Multi-agent evolution gives gains ranging from 4% to 20%

	Runtime	Task	Final Score				Impr. Rate (%)		# Evals	
			SOTA	1-Agent	4-Agent	Gain (%)	1-Agent	4-Agent	1-Agent	4-Agent
Stress	Claude Code + Opus 4.6	Polynominoes ↑	87.0	80.2	84.2	4.99	42.4	19.4	33	67
		Kernel Eng. ↓	1363	1350	1103	18.30	43.0	9.0	56	596
Math	OpenCode + MiniMax M2.5	Circle-Pack. ↑	2.6359	2.3531	2.5391	7.90	20.7	28.3	29	46
		Signal Proc. ↑	0.7429	0.7174	0.7383	2.91	30.5	19.8	59	253
		Erdős Over. ↓	0.38088	0.39237	0.38311	2.36	16.7	6.9	42	58
		MMD-16-2 ↓	12.89	12.91	12.89	0.15	32.4	11.5	34	103
		MMD-14-3 ↓	4.16	4.53	4.19	7.51	63.6	15.0	11	80
		3rd-Autocorr ↓	1.4557	1.5337	1.4931	2.65	42.9	22.1	7	59
System	OpenCode + MiniMax M2.5	EPLB ↑	0.145	0.128	0.129	0.78	50.0	65.4	6	26
		PRISM ↑	26.26	25.85	26.26	1.59	32.6	31.7	46	82
		LLM-SQL ↑	0.730	0.693	0.730	5.34	83.3	71.7	6	46
		Txn Sched. ↑	4348	3704	3774	1.89	50.0	66.7	16	24
		Cloudcast ↓	632.7	849.4	672.8	20.80	72.7	66.7	11	12

CORAL Analysis

<https://github.com/Human-Agent-Society/CORAL>

1. Importance of knowledge accumulation using memory of notes and skills.

Knowledge Accumulation (1-Agent)		
Task	w/ Know.	w/o Know.
Kernel Eng. ↓	1350	1601
Polyominoes ↑	80.2	77.3
Txn Sched. ↑	4566	4444

2. Importance of multi-agent co-evolution, versus multiple independent agents.

Co-evolution (4-Agent)		
Task	Co-evol.	Indep. Best
Kernel Eng. ↓	1103	1180
Polyominoes ↑	84.2	80.8
Txn Sched. ↑	4694	4629

CORAL Analysis

<https://github.com/Human-Agent-Society/CORAL>

1. **Cross-agent information transfer:** On kernel engineering, 36% of attempts use another agent as their parent, these improve at 17% compared to 9% for all attempts.
2. **Exploration diversity:** Agent's vocabulary overlap by 0.3-0.4 Jaccard similarity, so more than half their explorations are unique.
3. **Contribution balance:** All agents perform roughly similar number of evolutionary attempts with similar success rates.

Lecture Summary

- Planning and task decomposition is increasingly important for efficient agents.
- Multi-agent systems bridge planning + execution, decentralized exploration, multimodal agents. But they don't always work better!
- Multi-agent evolution can help turn hand engineering into autonomous optimization of multi-agent systems.