Unifying Language Agent Algorithms with Graph-based Orchestration Engine for Reproducible Agent Research

Qianqian Zhang [1], Jiajia Liao [2], Heting Ying [1], Yibo Ma [1], Haozhan Shen [3], Jingcheng Li [1], Peng Liu [1], Lu Zhang [1], Chunxin Fang [2], Kyusong Lee [1,2], Ruochen Xu [1], Tiancheng Zhao[1,2,*]

[1]Om AI Research, [2]Binjiang Institute of Zhejiang University,
[3]College of Computer Science and Technology, Zhejiang University

`tianchez@zju-bj.com`

Abstract

Language agents powered by large language models (LLMs) have demonstrated remarkable capabilities in understanding, reasoning, and executing complex tasks. However, developing robust agents presents significant challenges: substantial engineering overhead, lack of standardized components, and insufficient evaluation frameworks for fair comparison. We introduce Agent Graph-based Orchestration for Reasoning and Assessment (AGORA) [1] , a flexible and extensible framework that addresses these challenges through three key contributions: (1) a modular architecture with a graph-based workflow engine, efficient memory management, and clean component abstraction; (2) a comprehensive suite of reusable agent algorithms implementing state-of-the-art reasoning approaches; and (3) a rigorous evaluation framework enabling systematic comparison across multiple dimensions. Through extensive experiments on mathematical reasoning and multimodal tasks, we evaluate various agent algorithms across different LLMs, revealing important insights about their relative strengths and applicability. Our results demonstrate that while sophisticated reasoning approaches can enhance agent capabilities, simpler methods like Chain-of-Thought often exhibit robust performance with significantly lower computational overhead. AGORA not only simplifies language agent development but also establishes a foundation for reproducible agent research through standardized evaluation protocols.

1 Introduction

Language agents powered by large language models (LLMs) are rapidly transforming how we approach complex computational tasks across diverse domains. Industry adoption of these technologies is accelerating, with projections suggesting that 33% of organizations will implement LLM-based applications by 2025[2]. This growing adoption stems from the unprecedented ability of these systems to integrate natural language understanding with action-oriented capabilities.

Despite their promising trajectory, the practical implementation of language agents remains challenging for researchers and developers. Current frameworks often require substantial custom engineering efforts for each application domain, leading to fragmented implementations and difficulty in comparing different approaches.

To bridge this gap, we present AGORA, a comprehensive framework focused on both practical implementation and scientific evaluation of language agents. AGORA provides an integrated environment where researchers can experiment with various reasoning strategies while developers can build robust applications with minimal engineering overhead. Our framework makes three key contributions that differentiate it from existing approaches: a graph-based workflow orchestration engine that simplifies complex task execution; modular agent algorithm support for diverse reasoning paradigms; and easy-to-use client interfaces for evaluation and interaction.

Through systematic evaluation on mathematical and multimodal reasoning tasks, we demonstrate that AGORA not only facilitates rapid development but also enables rigorous scientific comparison of different agent paradigms. Our results provide actionable insights for researchers and practitioners navigating the growing landscape of language agent technologies.

[1]We made a demo video at: `https://www.youtube.com/watch?v=WRH-F1zegKI`. The comparison of agent algorithms across different LLMs is also available at `https://huggingface.co/spaces/omlab/open-agent-leaderboard`. Source code of AGORA can be found at `https://github.com/om-ai-lab/OmAgent`.

[2]`https://www.gartner.com/en/articles/intelligent-agent-in-ai?`

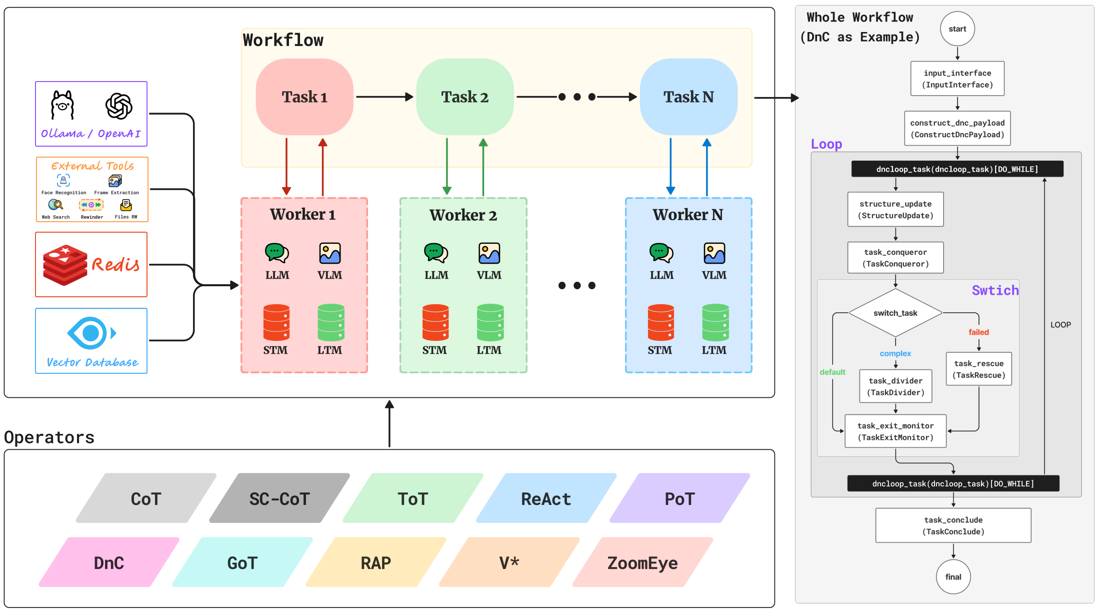


Figure 1: A demonstration of AGORA structure.

2 Related Work

Recent years have seen significant development in LLM agent frameworks and evaluation methodologies. Frameworks like LangChain (Developers, 2022), AutoGPT (Developers, 2023), and AgentVerse (Chen et al., 2024) offer general-purpose infrastructures for agent development, while AutoAgent (Tang et al., 2025) provides zero-code solutions through declarative interfaces. Specialized frameworks address domain-specific applications, including ChemCrow (Bran et al., 2023) for chemistry and OS-Copilot (Wu et al., 2024) for operating systems. For evaluation, comprehensive benchmark suites such as AgentBench (Liu et al., 2023b) and WebArena (Zhou et al., 2023) assess agents across multiple dimensions including reasoning, tool use, and web browsing. Leaderboard platforms like Agent Arena (Yekollu et al., 2024) enable systematic comparison of agents across models, frameworks, and tools through user-driven evaluations. A notable benchmark in this space is the Agent Leaderboard (Bhavsar, 2025), which primarily evaluates LLMs' tool calling and API interaction capabilities. Our work differs by providing a comprehensive evaluation framework that assesses both the underlying LLM capabilities and the effectiveness of different reasoning language agent algorithms, enabling researchers to understand the interplay between model selection and reasoning strategies.

3 AGORA Framework

AGORA is built on top of the OmAgent framework (Zhang et al., 2024), extending it into a flexible and extensible system for building, orchestrating, and evaluating language agents. It abstracts engineering complexity while exposing essential, reusable components—such as LLMs, VLMs, tools, and workflows—needed to construct powerful and research-friendly agents.

Graph-based Workflow Orchestration Engine. At the core of AGORA is a graph-based orchestration engine designed for modularity and scalability. As shown in Figure 1, the system uses a Directed Acyclic Graph (DAG) where each node represents a task. Tasks are either *simple tasks*—developer-defined custom logic—or *logical tasks*—built-in control flows such as branching and looping. Built on the Conductor library, this engine provides visual representations of workflows, making agent behavior intuitive to trace and debug. It also supports asynchronous, distributed execution, which is ideal for managing long-running, complex agent workflows.

Modular Agent Algorithm Support. AGORA includes a diverse set of agent algorithms such as Chain-of-Thought (CoT), Program-of-Thought (PoT), ReAct, Tree-of-Thought (ToT), and more. Each algorithm is implemented as a modular component, allowing developers to reuse common functions like memory access, LLM inference, or tool use. This structure encourages rapid prototyping, easy extensibility, and consistent evaluation across reasoning paradigms.

Client Interfaces for Evaluation and Interaction. After constructing an agent, AGORA provides a suite of Client interfaces tailored to different usage scenarios.

- **WebPageClient:** delivers a web-based chat interface that allows users to directly interact with the agent in real time, making it particularly suitable for qualitative studies such as usability testing or behavioral observation.
- **ProgrammaticClient:** supports automated evaluation using predefined JSON test files, making it ideal for quantitative studies with structured benchmarks—it efficiently runs batch test cases, logs outputs, and summarizes scores.
- **DefaultClient:** offers a lightweight command-line interface, designed for quick testing and debugging of agent logic during development. These clients are plug-and-play and can be easily configured via a configuration file, enabling researchers to seamlessly adapt the interface to different stages of experimentation and evaluation.

These client interfaces are plug-and-play and can be easily configured via a user-friendly config file, enabling seamless switching based on development or evaluation needs.

4 Agent Algorithms

The AGORA framework uses modular, reusable components called **operators** to simplify building and customizing AI systems. Each operator acts as a self-contained unit designed for a specific task, with clear input and output connections that make it easy to integrate into larger workflows.

We implemented various agent algorithms as operators and rigorously evaluated their performance in standardized, controlled environments. A description of the implemented agent algorithms is provided in Table 1. In particular, RAP enables more reliable and transparent decision-making processes by transforming complex reasoning tasks into systematic planning problems. The RAP implementation follows a tree-search-based architecture with four main components: selection, expansion, simulation, and backpropagation. In contrast to ToT, RAP enables backpropagation in the search framework, enhancing the efficiency of decision-tree traversal.

4.1 Implemented Agent Algorithms

In addition, We enhaced ReAct to ReAct-pro inspired by the Reflexion (Shinn et al., 2023) implementation. We modified our approach by separating the previously combined Think and Action steps into two distinct model calls, allowing the model to focus more intently on each phase. We also improved PoT by merging short-answer and multiple-choice questions processes into a single workflow consisting of two modules: the program executor and the answer extractor. For GoT, we extend the original GoT implementation into general GoT by allowing it to conduct any tasks other than the predefined tasks like sorting.

Algorithm 1 V*

```
function VSTAR(Image I, Query T)
    VWM ← Init(I, T)
    targets ← LLMIdentify(I, T)
    for each tar in targets do
        patchBox ← getSize(I)
        while true do
            if patchBox $\leq$ minCropSize then break
            end if
            imagePatch ← CropImage(I, patchBox)
                (scores, subImagePatchs, coords, conf) ← VisualSearch(imagePatch, tar)
                if conf $\geq$ thresh then
                    Store(VWM, tar, coords) break
                end if
                searchQueue ← HEAPPUSH(priorityQueue, (score, subImagePatch))
                if priorityQueue not empty then
                    patch ← HEAPPOP(priorityQueue)[1]
                    PatcheBox ← getSize(patch)
                end if
        end while
    end for
    return LLMAnalyze(VWM)
end function
```

5 Evaluation and Leaderboard

5.1 Evaluation Framework

Our experimental evaluation focused on two distinct domains: unimodal mathematical reasoning tasks and multimodal high-resolution image question-answering reasoning tasks. Mathematical reasoning tasks serve as canonical benchmarks for logical inference and problem decomposition, challenging agents to exhibit systematic reasoning and numerical accuracy. These tasks are inherently language-intensive yet require precise step-by-step deduction, making them ideal for evaluat-

Agent Algorithms	Description
Chain of Thought (CoT) (Wei et al., 2022)	Through encourage reasoning in the prompt, CoT enhances LLMs' reasoning by leverages intermediate steps, improving performance in complex tasks like arithmetic and symbolic reasoning. It can be broadly categorized into two types: Zero-shot-CoT and Few-shot-CoT (Kojima et al., 2022).
Self-Consistent CoT (SC-CoT) (Wang et al., 2022)	SC-CoT extends traditional CoT by generating multiple independent reasoning paths for the same problem and aggregating results through majority voting. This approach addresses the inherent variability in LLM reasoning by exploiting the observation that correct answers tend to emerge more consistently across different reasoning attempts than incorrect ones.
Tree of Thoughts (ToT) (Yao et al., 2023)	ToT facilitates advanced decision-making by examining coherent textual units, or "thoughts," as intermediate steps in problem-solving. Unlike traditional token-level approaches, ToT enables LLMs to construct and evaluate a thought tree using methods like Breadth-First Search (BFS) or Depth-First Search (DFS) to derive an optimal chain of thought.
Reasoning and Acting (ReAct) (Yao et al., 2022)	ReAct allows language models to engage with external environments through an iterative cycle of thought, action, and observation. The model reasons about the current state, executes relevant actions, and processes feedback until it gathers sufficient information to deliver a final response.
Program of Thought (PoT) (Chen et al., 2022)	PoT is designed to enhance the reasoning capabilities of language models by integrating programming language statements into their outputs. Unlike CoT, PoT leverages the strengths of language models like Codex to generate both text and executable code.
Divide-and-Conquer (DnC) (Zhang et al., 2024)	DnC enhances problem-solving by decomposing complex issues into manageable sub-problems. In this approach, LLMs alternate between the roles of conqueror, which directly addresses the problem, and divider, which breaks it down into smaller components. The conqueror and the divider operate in an iterative loop until the termination criteria are met.
Graph-of-Thought (GoT) (Besta et al., 2024)	GoT extends the ToT framework by introducing aggregation and refining transformations, enabling advanced graph-based reasoning. This approach decomposes tasks into identical subtasks, processes them independently, and aggregates sub-responses while leveraging internal loops to refine response quality.
Reasoning via Planning (RAP) (Hao et al., 2023)	RAP enhances LLMs by framing complex reasoning tasks as structured planning problems and employing a Monte Carlo Tree Search (MCTS) framework. The RAP implementation follows a tree-search-based architecture with four main components: selection, expansion, simulation, and backpropagation. Selection means intelligently choosing promising paths through the reasoning tree; Expansion breaks down complex questions into manageable sub-questions; Simulation evaluates potential solution paths through systematic exploration; and Backpropagation updates the search strategy based on solutions discovered. In contrast to ToT, RAP enables backpropagation in the search framework, enhancing the efficiency of decision-tree traversal.
V* (Wu and Xie, 2023)	V* introduces a meta-architecture for VLMs, SEAL (Show, sEArch, and TelL), a LLM-guided visual search method that enhances high-resolution image processing through iterative search and contextual reasoning. V* simulates human visual search process and leverages top-down features and contextual guidance to address the limitations of traditional visual encoders. First, V* assesses whether visual search is necessary. If so, the VLM identifies the target object. Subsequently, the LLM-guided search model recursively partitions the image into smaller regions and searches for the target based on the confidence scores derived from contextual cues until the target is located. The information about the identified target is stored in the Visual Working Memory (VWM). Finally, the VLM generates the response using the visual information of all targets stored in the VWM. A implementation of V* is presented in Algorithm 1.
ZoomEye (Shen et al., 2024)	ZoomEye is a training-free agent algorithm that enhances VLM performance on high-resolution images by simulating human zooming behavior. Treating the image as a tree structure, it dynamically explores zoomed-in regions based on visual cues and problem-specific priorities calculated by the VLMs.

Table 1: Agent algorithms implemented in AGORA.

ing the core reasoning capabilities of LLMs. Meanwhile, multimodal tasks involving high-resolution image understanding address the growing demand for agents to simulate real-world scenarios where contextual reasoning across diverse inputs is essential. Comprehensive experiments were conducted across multiple evaluation metrics, agent algorithms, and LLMs to assess reasoning capabilities in both domains.

To evaluate language agents, this study defines four key metrics: accuracy, cost, token usage, and pass rate. Specially, accuracy assesses the proportion of predictions that exactly match the ground-truth response; cost quantifies the total expenditure incurred measured in US dollar. We used API services for close-sourced models and models with more than 70 billion from SiliconFlow[1] and OpenAI[2]; Token usage measures the number of tokens that a language agent uses to generate predictions, and pass rate measures the proportion of valid predictions among all predictions, where a prediction is considered valid if it is neither empty nor null.

5.2 Experimental Setup

5.2.1 Mathematical Reasoning Tasks

The mathematical reasoning benchmarks include:

GSM8K (Cobbe et al., 2021): A dataset for evaluating language agents' ability to solve elementary math word problems. we conducted the evaluation using 8-shot learning.

AQuA (Ling et al., 2017): This dataset is specifically designed to reason through diverse algebraic problems to assess reasoning abilities. We employed zero-shot learning in the experiments.

[1]SiliconFlow: https://siliconflow.cn/zh-cn/

[2]OpenAI: https://openai.com/

MATH-500 (Hendrycks et al., 2021): A dataset comprising 500 mathematical reasoning problems has been meticulously designed to evaluate the ability of language agents to tackle complex mathematical challenges, where 4-shot learning is applied.

We applied both commercial and open-source models in the experiments.

Commercial Models: In our experiment, GPT-3.5 Turbo and GPT-4o from OpenAI, and Doubao-lite-32k from ByteDance were used as LLM for agent algorithms, and GPT-3.5 Turbo was also used for the extraction of AQuA answers.

Open-source models: We also evaluated open source models like Llama and Qwen for performance and cost effectiveness. We used the following models as the LLMs for Agents: Qwen2.5-72B-Instruct, Qwen2.5-7B-Instruct (Yang et al., 2024b), Qwen2-1.5B-Instruct, Qwen2-0.5B-Instruct (Yang et al., 2024a), Llama-3.3-70B-Instruct, Llama-3.1-8B-Instruct (Grattafiori et al., 2024), InternLM2.5-7B-Chat (Cai et al., 2024), deepseek-r1-1.5B (Guo et al., 2025).

In the experiments, the default setting uses a temperature of 0. More algorithm settings other than default can be found in Appendix A.

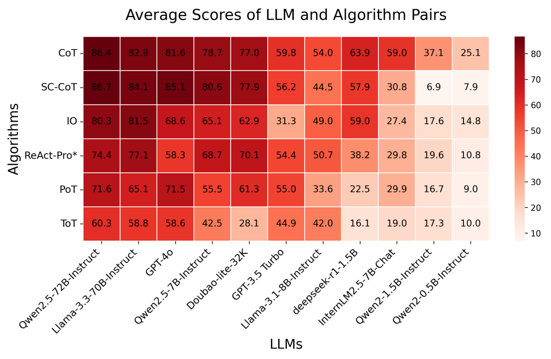
5.2.2 Multimodal Reasoning Tasks

Regarding multimodal reasoning task, we implemented MME-RealWorld (Zhang et al., 2025) as the benchmark. MME-RealWorld aims at solving high-resolution image problems highly relevant to real-world applications. Specifically, we selected images with resolutions between 2K and 4K in the lite version. We implemented V* and ZoomEye in the evaluation, implementation details can be found in Appendix A. Because we only applied open source VLMs and all models used were deployed locally, cost is not involved for evaluation.

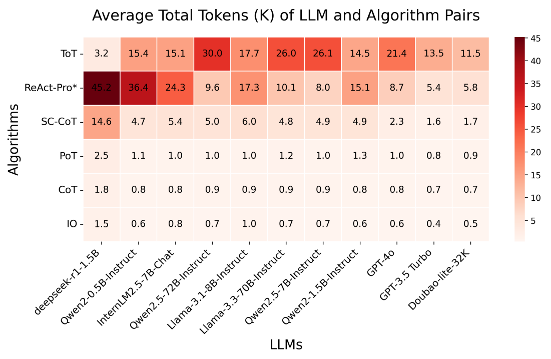
5.3 Mathematical Reasoning Results

5.3.1 Performance Comparison

The average scores and average token consumptions of LLM and algorithm pairs are illustrated in Figure 2, where the average token consumption is calculated by first summing the input and output tokens per sample for each dataset, then computing the overall mean across all benchmarks. The comparison details can be found at Open Agent leaderboard (Lab, 2025). Furthermore, we performed a score versus cost analysis for different LLM agent algorithms, as depicted in Figure 3. The dashed line in the plot represents an ideal trend line, which



(a) Average scores.



(b) Average input and output token consumptions.

Figure 2: LLMs and agent algorithms average scores and average token consumptions on mathematical reasoning tasks.

serves as a visual benchmark, illustrating the optimal balance between cost and performance. Points on the top-left corner indicate agent-LLM pairs that offer the best possible trade-off between task accuracy and computational cost. Models smaller than 7B parameters were self-hosted locally, thus their cost metrics are not shown. It should be mentioned that GoT, RAP and DnC were excluded from the comparison. GoT is specifically designed to decompose complex tasks into several identical sub-tasks, such as sorting and keyword counting. RAP and DnC was not included due to its high token consumption.

Open-source models with 70 billion parameters have demonstrated exceptional performance compared to other models. Also, Qwen2.5-7B-Instruct surpasses GPT-3.5 Turbo in this task. Surprisingly, deepseek-r1-1.5B, with only 1.5 billion parameters, exhibits remarkable performance by outperforming the InternLM2.5-7B-Chat model. When considering different agent algorithms, the simplest CoT approach also outperforms other agent algorithms while utilizing the least number of tokens.

5.3.2 Key Findings

Simple agent algorithms show robust performance. CoT and SC-CoT algorithm has demonstrated remarkable performance despite their simplicity. Utilizing the Doubao-lite-32k model, CoT achieved an accuracy of 89.31% on the GSM8K dataset, with a token cost of only $0.0558. However, SC-CoT encounters challenges with smaller models, which struggle to strictly adhere to instructions, resulting in difficulties parsing the output. Notably, more advanced algorithms, such as PoT and TOT, which incorporate external tools, perform worse on mathematical problems compared to the simpler algorithms. We observed that PoT's reliance on the code generation and parsing capabilities of LLMs does not lead to significant improvements compared to other agent algorithms. In fact, it can have negative effects, particularly with smaller LLM models due to the code generation quality. Moreover, the thinking generation and state evaluation for ToT does not significantly reduce the difficulty of reasoning, but rather significantly increases its token usage, which leads to exhibiting poorer performance.

This phenomenon prompts a reflection on the value of algorithmic complexity. The advantage of simpler methods is primarily reflected in the reduction of error accumulation. Complex agent algorithms often involve multiple steps, each potentially introducing errors, whereas a single reasoning chain significantly reduces the risk of error propagation. CoT's simple prompts are easier to adjust and optimize, making the reasoning process more transparent, easier to understand, and improved. In terms of cost-effectiveness, CoT's advantages are even more apparent. Lower token consumption translates to reduced operational costs, and faster reasoning speeds enhance system responsiveness. Additionally, the straightforward implementation reduces development and maintenance costs. These findings offer important practical insights. When designing intelligent systems, we should prioritize simple and direct solutions, introducing complexity only when necessary. It is advisable to start with a basic CoT implementation and gradually optimize based on the specific task characteristics, while carefully evaluating the actual benefits of each added complexity.

Agent algorithms can be sensitive to prompts. We also noticed the importance of prompt design. As shown in Table 2, the base ReAct achieved a baseline performance of 34.25% on the AQuA dataset. Inspired by the Reflexion implementation, we prompt ReAct to ReAct-Pro by separating the previously combined Think and Action steps into two distinct model calls, allowing the model to focus more intently on each phase. This modification alone boosted accuracy to 40.16%. The real breakthrough came from a remarkably simple addition by including the sentence: "You can take as many steps as needed" in the prompt, we observed an extraordinary increase in accuracy to 64.57%, an almost 90% improvement over the baseline. This simple prompt fundamentally transformed the model's behavior patterns.

Agent Algorithm	Dataset	LLM	Score
ReAct	GSM8K	GPT-3.5 Turbo	38.13
ReAct-Pro	GSM8K	GPT-3.5 Turbo	74.91
ReAct	AQuA	GPT-3.5 Turbo	34.25
ReAct-Pro	AQuA	GPT-3.5 Turbo	64.57

Table 2: Comparison of ReAct and ReAct-Pro on different datasets.

Open-source models are competitive with commercial ones. Open-source models at the 70B level, such as Llama-3.3-70B-Instruct and Qwen2.5-72B-Instruct, have shown outputs that exceed those of the closed-source GPT-4o. However, the enhancement brought by agent frameworks to top-tier large models (such as GPT and models above 70B) is relatively limited. In some cases, complex agents like ReAct may even lead to a decline in performance.

Small models perform better with simple agent algorithms. For smaller models, such as Qwen2.5-7B-Instruct, CoT demonstrates a marked improvement, while PoT shows limited enhancement. This limitation is primarily attributed to the bottleneck in code generation capabilities.

5.4 Multimodal Reasoning Results

5.4.1 Performance Comparison

We compared IO, V*, and ZoomEye using various models. The detailed comparison results are shown in Table 3 in Appendix C. It is important to note that due to the specific nature of the V* models, we were unable to obtain their token usage data. Overall, the final scores of the same models improved after using the ZoomEye framework, particularly the Qwen2.5-VL-7B-Instruct model, which even outperformed the Qwen2.5-VL-72B-Instruct IO. After applying the agent algorithms, both the input and output token usage increased significantly. Notably, the Qwen2.5-VL models (7B and 72B) demonstrated identical token consumption patterns in IO, which can be attributed to their strong instruction adherence capabilities and the multiple-choice format of the benchmark questions. Moreover, the V* framework received one of the lowest scores, primarily due to its low pass rate.

5.4.2 Key Findings

In our experiments, we found that the performance of the models was generally improved after using a multimodal agent workflow like ZoomEye, especially the 7B model outperformed the 72B model. This phenomenon suggests that adopting multimodal agent can effectively provide more visual details in final answer, thus helping the model to generate more accurate answers. Therefore, if computational resources are sufficient, it is recommended to prioritize models with larger parameters to fully leverage their potential. However, if computational resources are limited, smaller models combined with efficient agent workflows can still achieve comparable results.

6 Discussion

In the design of agent systems, it is crucial to prioritize straightforward and direct solutions, incorporating complexity only when necessary. It is recommended to begin with a fundamental CoT that achieves a balance between performance and cost. Complexity can be progressively increased based on task requirements (e.g., using ToT for hierarchical planning when CoT proves insufficient) , ensuring a systematic trade-off between efficiency and task complexity. For the selection of LLMs, we recommend utilizing models with at least 7 billion parameters or employing reasoning models such as deepseek-r1. This recommendation is primarily due to the tendency of smaller models to exhibit issues with instruction adherence. Furthermore, we noticed multimodal agent algorithms like ZoomEye can enhance agent performance by providing valuable visual details. Although larger models should be prioritized when resources allow, smaller models can still yield competitive outcomes.

7 Conclusions

In this paper, we present AGORA , a comprehensive framework for building and evaluating language agent algorithms that addresses critical challenges of engineering overhead, fragmented implementations, and insufficient evaluation standards. Our graph-based workflow orchestration engine (built on DAGs) enables dynamic task decomposition and asynchronous distributed execution. Meanwhile, its modular design standardizes agent algorithms (e.g., CoT, V*) for plug-and-play integration. The multi-client evaluation interfaces also facilitate both qualitative user studies and quantitative benchmarking, enabling rigorous cross-algorithm comparisons across LLMs and tasks. We systematically integrated 10 state-of-the-art agent algorithms spanning from CoT to V*, under a unified modular architecture, which reduces engineering overhead.

Our evaluation across mathematical and multimodal tasks revealed several important insights. First, simpler reasoning approaches like CoT often demonstrate robust performance and consume less cost than more complex alternatives. Second, the effectiveness of different agent algorithms varies substantially across different model sizes. Third, for multimodal tasks, specialized agent algorithms like ZoomEye can substantially enhance model performance on high-resolution images, highlighting the value of reasoning strategies using VLMs.

As the field continues to evolve, we believe this framework will serve as a valuable foundation for exploring increasingly sophisticated agent architectures and reasoning approaches. Future work of AGORA should focus on: (1) expanding the evaluation framework to encompass broader complex real-world tasks (e.g., tool utilization and web interaction scenarios); (2) developing adaptive agents that dynamically select optimal reasoning strategies based on task characteristics; and (3) prioritizing seamless integration of emerging LLMs via extensions to AGORA's modular architecture.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Pratik Bhavsar. 2025. Agent leaderboard. https://huggingface.co/spaces/galileo-ai/agent-leaderboard.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. Internlm2 technical report. *Preprint*, arXiv:2403.17297.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Wenhu Chen et al. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yimin Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, Conghui He, Botian Shi, Xingcheng Zhang, Han Lv, Yi Wang, Wenqi Shao, Pei Chu, Zhongying Tu, Tong He, Zhiyong Wu, Huipeng Deng, Jiaye Ge, Kai Chen, Kaipeng Zhang, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. 2025. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *Preprint*, arXiv:2412.05271.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

AutoGPT Developers. 2023. Autogpt.

LangChain Developers. 2022. Langchain: Framework for developing applications powered by language models.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Om AI Lab. 2025. Open agent leaderboard. https://github.com/om-ai-lab/open-agent-leaderboard.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. *Preprint*, arXiv:2304.08485.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023b. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv: 2308.03688*.

Haozhan Shen, Kangjia Zhao, Tiancheng Zhao, Ruochen Xu, Zilun Zhang, Mingwei Zhu, and Jianwei Yin. 2024. Zoomeye: Enhancing multi-modal llms with human-like zooming capabilities through tree-based image exploration. *Preprint*, arXiv:2411.16044.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Jiabin Tang, Tianyu Fan, and Chao Huang. 2025. Autoagent: A fully-automated and zero-code framework for llm agents. *arXiv e-prints*, pages arXiv–2502.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*. Published at ICLR 2023.

Jason Wei et al. 2022. Chain of thought prompting elicits reasoning in large language models. In *NeurIPS*.

Penghao Wu and Saining Xie. 2023. V*: Guided visual search as a core mechanism in multimodal llms. *Preprint*, arXiv:2312.14135.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. 2024. Os-copilot: Towards generalist computer agents with self-improvement. *arXiv preprint arXiv:2402.07456*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024b. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*. Published at ICLR 2023.

Shunyu Yao et al. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*.

Nithik Yekollu, Arth Bohra, Ashwin Chirumamilla, Kai Wen, Sai Kolasani Wei-Lin Chiang, Anastasios Angelopoulos, Joseph E. Gonzalez, Ion Stoica, and Shishir G. Patil. 2024. Agent arena.

Lu Zhang, Tiancheng Zhao, Heting Ying, Yibo Ma, and Kyusong Lee. 2024. Omagent: A multi-modal agent framework for complex video understanding with task divide-and-conquer. *arXiv preprint arXiv:2406.16620*.

Yi-Fan Zhang, Huanyu Zhang, Haochen Tian, Chaoyou Fu, Shuangqing Zhang, Junfei Wu, Feng Li, Kun Wang, Qingsong Wen, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. 2025. Mme-realworld: Could your multimodal llm challenge high-resolution real-world scenarios that are difficult for humans? *Preprint*, arXiv:2408.13257.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Agent Algorithm Parameter Settings

In the experiments of this paper, the default setting for LLMs uses a temperature of 0. For ReAct-Pro, the parameter is set with a maximum number of steps equal to 10. For SC-CoT, the temperature is 1 and the number of paths is 5; For TOT, we use bfs as the search method, with b as 1, max depth and a max steps are both setted as 6, and the number of evaluations is 3.

The mulitmodal model configration is described as follows:

V*: The SEAL structure uses specific models trained on llava-7b, including seal_vqa_7b and seal_vsm_7b. seal_vqa is responsible for identifying and providing the target objects needed for the search from question, as well as utilizing the data in the VWM(visual working memory) to answer the relevant questions. seal_vsm combines the common sense knowledge with the context of the image to locate the target object and records its information into VWM. Due to the specificity of the model, parameters such as temperature and max_tokens were not configured. As for the visual search parameters such as the confidence threshold, we use the same parameters as the original settings: confidence maximum 0.5, minimum 0.3, target cue threshold 6.0, target cue threshold decay 0.7, target cue threshold minimum 3.0. In addition we set 10 as the maximum search steps for each target. The reason for this is that the minimum image size of Vstar is 224×224, which can take an hour or even longer when searching for high-resolution images (e.g., 4K images) if we do not limit the number of search steps.

ZoomEye: As a more generalized agent visual search framework, we apply and evaluate a variety of mainstream open-source multimodal models, including Llava-v1.5-7B (Liu et al., 2023a), InternVL2.5-8B (Chen et al., 2025), Qwen2.5-VL-7B-Instruct (Bai et al., 2025), and Qwen2.5-VL-72B-Instruct, which support a wide range of complex multimodal visual questioning tasks. For these VLMs, we set temperature to 0.0 and max_tokens to 2048. We also set the same parameters as the ZoomEye original settings:

- Answering Confidence Threshold:
  - Maximum: 0.4
  - Minimum: 0
- Smallest Patch Size: 384
- Depth Limit: 5
- Number of Intervals: 2
- Threshold Decrease: [0.1, 0.1, 0.2]

B Score Versus Cost Analysis on Mathematical Reasoning

C Performance Comparison on Multimodal Reasoning

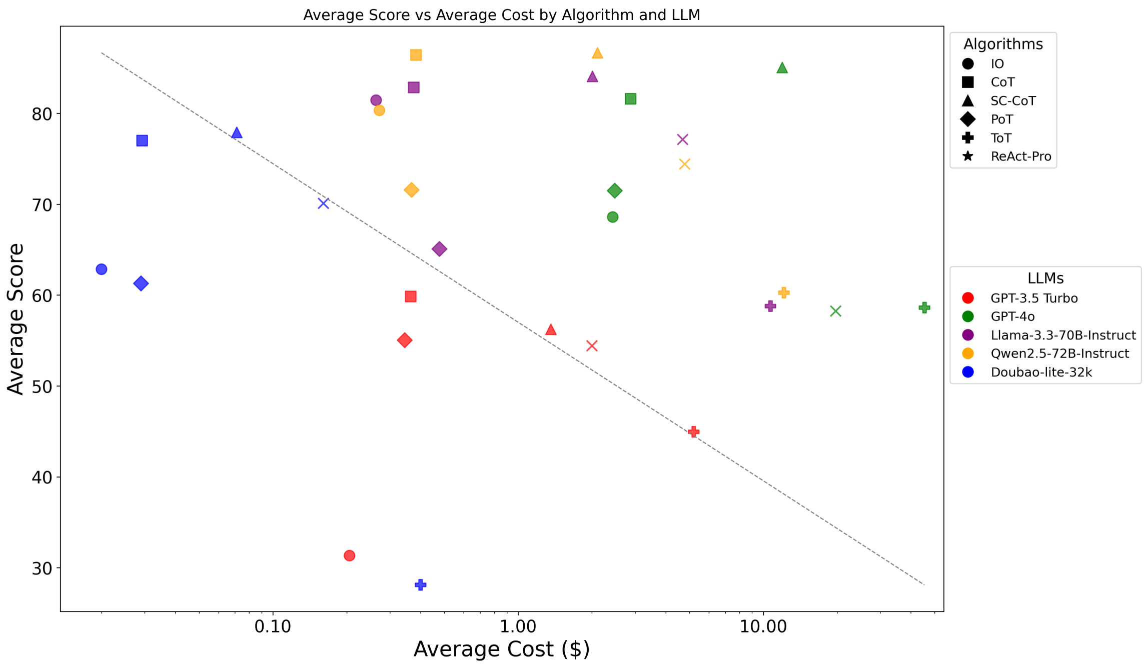


Figure 3: Score versus cost analysis for different LLM agent algorithms. The ideal models appear in the top-left corner with high performance and low cost. Models smaller than 7B parameters were self-hosted locally, thus their cost metrics are not shown.

Agent	VLMs	Score	Pass Rate	Total Input Tokens	Total Output Tokens	All Tokens
ZoomEye	Qwen2.5-VL-72B-Instruct	51.56	99.81	76,808,965	1,276,460	78,085,425
ZoomEye	Qwen2.5-VL-7B-Instruct	48.06	96.50	94,418,593	1,472,836	95,891,429
IO	Qwen2.5-VL-72B-Instruct	44.47	100.00	6,174,490	2,114	6,176,604
ZoomEye	InternVL2.5-8B	43.42	99.34	153,857,588	2,017,170	155,874,758
IO	InternVL2.5-8B	42.95	100.00	2,779,778	2,335	2,782,113
IO	Qwen2.5-VL-7B-Instruct	42.86	100.00	6,174,490	2,114	6,176,604
ZoomEye	Llava-v1.5-7B	31.60	98.86	113,073,261	1,368,724	114,441,985
IO	Llava-v1.5-7B	24.79	100.00	734,868	17,036	751,904
V*	seal_vqa & seal_vsm	15.14	72.37	-	-	-

Table 3: Performance comparison of different agents and VLMs on MME-RealWorld.