VERIFIED MULTI-AGENT ORCHESTRATION: A PLAN-EXECUTE-VERIFY-REPLAN FRAMEWORK FOR COMPLEX QUERY RESOLUTION

Xing Zhang[1] **Yanwei Cui**[1] **Guanghui Wang**[1] **Wei Qiu**[2] **Ziyuan Li**[2]
Fangwei Han[2] **Yajing Huang**[2] **Hengzhi Qiu**[2] **Bing Zhu**[2] **Peiyang He**[1]*

[1]AWS Generative AI Innovation Center [2]HSBC Holdings Plc., HSBC Technology Center, China

ABSTRACT

We present **Verified Multi-Agent Orchestration (VMAO)**, a framework that coordinates specialized LLM-based agents through a verification-driven iterative loop. Given a complex query, our system decomposes it into a directed acyclic graph (DAG) of sub-questions, executes them through domain-specific agents in parallel, verifies result completeness via LLM-based evaluation, and adaptively replans to address gaps. The key contributions are: (1) dependency-aware parallel execution over a DAG of sub-questions with automatic context propagation, (2) verification-driven adaptive replanning that uses an LLM-based verifier as an orchestration-level coordination signal, and (3) configurable stop conditions that balance answer quality against resource usage. On 25 expert-curated market research queries, VMAO improves answer completeness from 3.1 to 4.2 and source quality from 2.6 to 4.1 (1–5 scale) compared to a single-agent baseline, demonstrating that orchestration-level verification is an effective mechanism for multi-agent quality assurance.

1 INTRODUCTION

Large language models (LLMs) have enabled a new generation of multi-agent systems where specialized agents collaborate to solve complex tasks. A central challenge in such systems is *coordination*: given a complex query that requires information from heterogeneous sources and diverse analytical expertise, how should agents be organized and assigned to sub-tasks? How can we ensure result quality without constant human oversight? When should the system stop iterating and synthesize a final answer? These questions are especially acute in domains like *market research*, where analysts gather data from internal databases, public filings, news sources, and competitor reports, then synthesize findings into actionable insights. Information is scattered across heterogeneous sources, analysis requires diverse expertise (financial, operational, competitive), and synthesis demands cross-referencing while resolving contradictions.

Existing multi-agent frameworks fall short of these requirements. Debate-style approaches where agents critique each other's outputs (Du et al., 2023) improve reasoning quality but lack structured task decomposition. Role-playing frameworks where agents assume personas (Li et al., 2023) enable collaboration but provide no mechanism for verifying completeness. More recent systems like AutoGen (Wu et al., 2024) and MetaGPT (Hong et al., 2024) offer flexible interaction patterns, yet still lack principled quality verification and adaptive refinement—critical requirements for production deployment where outputs must be reliable without constant human oversight.

We introduce **Verified Multi-Agent Orchestration (VMAO)**, a framework that addresses these gaps through three key contributions:

1. **DAG-Based Query Decomposition and Execution**: Complex queries are decomposed into sub-questions organized as a directed acyclic graph (DAG), enabling dependency-aware parallel execution with automatic context propagation from upstream results.

*Corresponding author: peiyan@amazon.com

2. **Verification-Driven Replanning**: An LLM-based verifier evaluates result completeness at the orchestration level, triggering adaptive replanning when gaps are identified—providing a principled coordination signal that is decoupled from individual agent implementations.
3. **Configurable Stop Conditions**: Termination decisions are based on completeness thresholds, confidence scores, and resource constraints, enabling explicit quality-cost tradeoffs.

On 25 expert-curated market research queries, VMAO improves answer completeness from 3.1 to 4.2 and source quality from 2.6 to 4.1 (1–5 scale) compared to single-agent and static multi-agent baselines.

2 Related Work

Multi-Agent Coordination and Tool Use. Recent surveys (Wang et al., 2024; Xi et al., 2023) document the rapid growth of LLM-based multi-agent systems, which vary in coordination strategy: AutoGen (Wu et al., 2024) uses conversational patterns, CAMEL (Li et al., 2023) employs role-playing, MetaGPT (Hong et al., 2024) enforces software engineering workflows, and HuggingGPT (Shen et al., 2023) orchestrates specialized models via a central controller. Orthogonally, work on tool use has focused on single-agent settings: ReAct (Yao et al., 2023b) established the thought-action-observation paradigm, Toolformer (Schick et al., 2023) enables self-supervised tool learning, and ToolLLM (Qin et al., 2023) scales to 16,000+ APIs. These lines of work address coordination and tool use separately, but production systems require both: multiple specialized agents, each with domain-specific tools, working in concert.

Planning, Decomposition, and Verification. Chain-of-Thought (Wei et al., 2022), Tree-of-Thoughts (Yao et al., 2023a), and Least-to-Most prompting (Zhou et al., 2023) decompose complex reasoning into structured steps, but operate within a single LLM rather than distributing sub-tasks across specialized agents. For output quality, Self-Consistency (Wang et al., 2022) aggregates multiple reasoning paths, Self-Refine (Madaan et al., 2023) iterates on single outputs, and Reflexion (Shinn et al., 2023) uses verbal reinforcement—all operating at the individual response level. Missing from prior work is verification at the *orchestration level*: evaluating whether collective results from multiple agents adequately address the original query, and triggering targeted replanning when gaps are detected.

Agentic Search and Deep Research. Recent commercial systems have demonstrated the potential of multi-step agentic research: search-augmented assistants like Perplexity iteratively refine queries to synthesize information from web sources, while deep research features in frontier models (OpenAI, 2025) perform extended multi-step investigation. These systems demonstrate the value of iterative research loops but are closed-source, making their coordination mechanisms difficult to study or reproduce. Our work provides an open, modular framework where the coordination strategy—particularly the verification-driven replanning loop—is explicit and configurable.

Our Approach. VMAO synthesizes these threads into a unified framework for complex query resolution. We decompose queries into a DAG of sub-questions assigned to domain-specific agents, execute them in parallel with dependency-aware scheduling, verify collective completeness via LLM-based evaluation, and adaptively replan to address gaps. We evaluate VMAO on market research tasks, maintaining verifiable output quality through explicit coordination mechanisms.

3 Framework Architecture

3.1 Overview

VMAO operates through five phases: **Plan**, **Execute**, **Verify**, **Replan**, and **Synthesize** (Figure 1a). Given a complex query, the system first decomposes it into sub-questions with assigned agent types and dependencies. It then executes these through specialized agents in parallel where dependencies permit. The verify phase evaluates completeness and identifies gaps. If deficiencies exist, the system replans by generating new sub-questions or marking incomplete ones for retry. This loop continues until stop conditions are met, triggering synthesis of a final answer with proper source attribution.

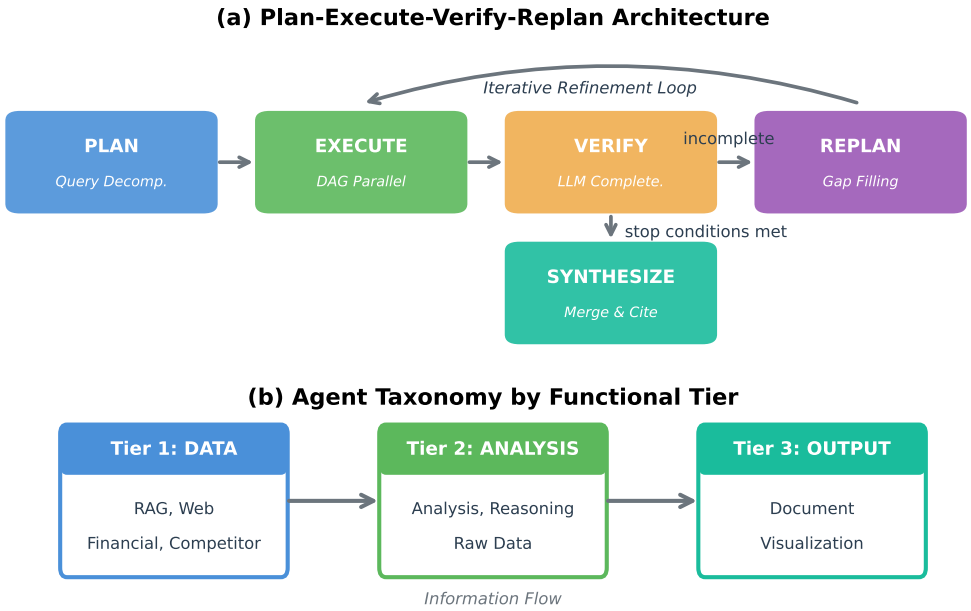

Figure 1: (a) VMAO framework architecture showing the iterative Plan-Execute-Verify-Replan loop. (b) Agent taxonomy organized by functional tier with information flow from data gathering through analysis to output generation.

Table 1: Sub-question structure generated by the QueryPlanner

Field	Description
id	Unique identifier (e.g., sq_001)
question	Specific, answerable question text
agent_type	Agent from taxonomy to handle this question
dependencies	IDs of sub-questions that must complete first
priority	Execution priority (1–10); higher = more important
context_from_deps	Whether to include dependency results in prompt
verification_criteria	Criteria for determining answer completeness

Agents are organized into three functional tiers (Figure 1b): Tier 1 (Data Gathering) agents retrieve information from diverse sources, Tier 2 (Analysis) agents reason over this data, and Tier 3 (Output) agents produce final deliverables. This hierarchy reflects the natural information flow in research tasks and enables principled task assignment by the planner.

3.2 Planning and Execution

The **QueryPlanner** decomposes a complex query into sub-questions organized as a DAG (Table 1). An LLM identifies distinct information requirements, assigns each to an appropriate agent type, establishes dependencies where one sub-question requires another's output, and sets execution priorities.

The **DAGExecutor** then orchestrates execution while respecting dependencies and maximizing parallelism (Algorithm 1). It iteratively identifies ready questions—those whose dependencies have completed—and executes batches in parallel (default $k = 3$). For sub-questions with `context_from_deps` enabled, results from dependencies are prepended to the query. Figure 2a illustrates how independent sub-questions execute concurrently in Wave 1, while dependent questions execute in subsequent waves. Each execution is wrapped with a configurable timeout (default: 600s) and a tool call limiter to prevent infinite loops.

Algorithm 1 DAG-Based Parallel Execution

Require: Execution plan $P = (Q, G)$, max concurrent k
Ensure: Results $R = \{r_1, ..., r_n\}$

$completed \leftarrow \emptyset$
while $|completed| < |Q|$ **do**
  $ready \leftarrow \{q \in Q : deps(q) \subseteq completed \wedge q \notin completed\}$
  $batch \leftarrow$ top-$k(ready, \text{by} = priority)$
  $results \leftarrow$ parallel_execute$(batch)$
  for (q, r) in $results$ **do**
    if $q.context\_from\_deps$ **then**
      $r \leftarrow$ enrich_with_context$(r, \{R[d] : d \in deps(q)\})$
    end if
    $R[q.id] \leftarrow r$; $completed \leftarrow completed \cup \{q\}$
  end for
end while
return R

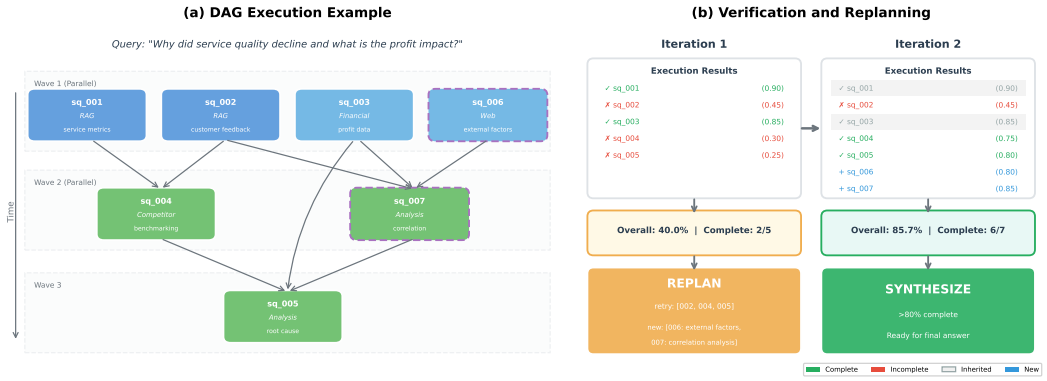

Figure 2: (a) DAG execution: independent sub-questions execute in Wave 1; dependent questions in subsequent waves. (b) Verification-driven iteration: Iteration 1 identifies incomplete results, triggering replanning; Iteration 2 achieves sufficient completeness for synthesis.

3.3 Verification, Replanning, and Synthesis

The **ResultVerifier** evaluates whether execution results adequately answer their sub-questions (Figure 2b). For each result, it produces: status (complete/partial/incomplete), completeness score (0–1), missing aspects, contradictions, and a recommendation (accept/retry/escalate). Results already marked complete are reused to avoid redundant LLM calls.

When verification identifies gaps, the **AdaptiveReplanner** determines corrective actions: *retry* sub-questions with low scores while preserving previous results, introduce *new queries* to address specific missing aspects, or *merge* results from multiple attempts. A key feature is result preservation—previous results are stored and merged with retry attempts, enabling progressive refinement without losing earlier findings.

Determining when to stop iterating is critical for balancing quality and cost. We introduce five configurable stop conditions (Table 2), evaluated after each verification phase: completeness threshold (80% of sub-questions answered), high confidence with partial coverage, diminishing returns (<5% improvement), token budget (1M tokens), and maximum iterations (3). When any condition is met, the system proceeds to synthesis.

For large result sets (>15K characters or 10+ results), direct synthesis would exceed context limits. We address this through hierarchical synthesis: group results by agent type, synthesize within each group to produce condensed summaries, then integrate group summaries into a coherent final answer with proper source attribution.

Table 2: Stop conditions for orchestration termination

Condition	Threshold	Rationale
Ready for Synthesis	80% complete	Sufficient sub-questions answered
High Confidence	75% conf, 50% complete	High reliability despite partial coverage
Diminishing Returns	<5% improvement	Further iteration yields minimal gain
Token Budget	1M tokens	Hard cost limit
Max Iterations	3 iterations	Hard iteration limit

Table 3: Agent taxonomy with tool allocation across MCP servers (42 unique tools total)

Tier	Agent	Tools	Key Capabilities
1: Data	RAG	13	Semantic, keyword, and hybrid retrieval; metadata filtering
	Web Search	4	General and AI-powered search, news retrieval
	Financial	7	Stock quotes, technical indicators, fundamentals
	Competitor	11	Market positioning, benchmarks, competitor news
2: Analysis	Analysis	20	Survey analytics, financial and competitor analysis
	Reasoning	24	Cross-domain reasoning with RAG, web, and financial tools
	Raw Data	1	Python execution (pandas, matplotlib)
3: Output	Document	4	Report generation, tables, source citations
	Visualization	6	Chart generation, statistical summaries

4 IMPLEMENTATION

We implement VMAO using LangGraph for workflow orchestration and the Strands Agent framework for agent execution, integrated with AWS Bedrock. Agent execution uses Claude Sonnet 4.5 as the primary model with Claude Haiku 4.5 as a fallback for graceful degradation; verification and evaluation use Claude Opus 4.5 to provide an independent quality signal. Agents access tools through the Model Context Protocol (MCP), which exposes domain-specific capabilities via independent HTTP microservices. This modular architecture allows adding new tools without modifying agent code.

Table 3 shows the agent taxonomy with tool allocation across eight MCP servers (42 unique tools total). Each server runs independently, enabling horizontal scaling and fault isolation. Agents automatically select appropriate tools based on sub-question requirements.

For production deployment, we implement several safety mechanisms: tool call limiters prevent infinite loops (max 10 consecutive same-tool calls, 50 total per agent), per-execution timeouts enforce bounded latency (default 600s), and phase-level token tracking enables budget enforcement. When the primary model (Sonnet 4.5) is unavailable, the system falls back to Haiku 4.5 with graceful degradation. Real-time observability is provided through Server-Sent Events that stream execution progress to the frontend.

5 EXPERIMENTS

5.1 DATASET: MARKET RESEARCH QUERIES

We evaluate VMAO on **market research tasks**—a domain where traditional research typically requires 2–4 weeks of human effort. These tasks are challenging because relevant data is scattered across heterogeneous sources, answering questions requires diverse expertise (financial, operational, competitive), and synthesis demands cross-referencing while resolving contradictions. We curated 25 queries from domain experts spanning four categories:

- **Performance Analysis** (8 queries): Operational metrics, trends, and causal factors. *Example*: "What factors explain the year-over-year change in customer satisfaction?"
- **Competitive Intelligence** (7 queries): Comparison with industry peers and market positioning. *Example*: "How does our market share compare to regional competitors?"
- **Financial Investigation** (5 queries): Financial metrics combined with operational context. *Example*: "What is driving the change in revenue per customer?"
- **Strategic Assessment** (5 queries): Open-ended synthesis across multiple dimensions. *Example*: "What are the key risks and opportunities for geographic expansion?"

Query complexity varies from simpler queries (3–5 sub-questions, 2–3 agent types) to complex ones (8–12 sub-questions, 5+ agent types with multi-level dependencies). Each query consumes 500K–1.1M tokens and requires 10–20 minutes of execution plus domain expert review, making 25 queries a practical yet meaningful evaluation set.

5.2 Baselines and Configuration

We compare three configurations:

- **Single-Agent**: One reasoning agent with access to all tools, relying on internal reasoning to determine tool invocation order.
- **Static Pipeline**: Predefined agent sequence (RAG $\rightarrow$ Web $\rightarrow$ Financial $\rightarrow$ Analysis $\rightarrow$ Synthesis) without verification or replanning.
- **VMAO**: Full framework with dynamic decomposition, parallel execution, verification-driven replanning, and stop conditions.

All configurations use Claude Sonnet 4.5 for agent execution and the same tool set. We evaluate *Completeness* (how thoroughly all query aspects are addressed, 1–5 scale) and *Source Quality* (proper citation and traceability, 1–5 scale). Evaluation follows a two-stage process: an LLM judge (Claude Opus 4.5) first scores each response using structured rubrics, then human domain experts review and adjust scores where the LLM assessment appears inconsistent or misses domain-specific nuances. We deliberately use a different, more capable model for evaluation than for execution to reduce self-evaluation bias, though both models belong to the same family. In practice, human reviewers adjusted fewer than 15% of LLM scores, typically by ± 0.5 points, indicating reasonable LLM-human alignment on these metrics.

We evaluate *Completeness* rather than accuracy because deep research queries have no single ground truth—a question like "What factors explain declining satisfaction?" admits multiple valid answers. Completeness measures whether all relevant aspects are addressed with supporting evidence, better capturing the exploratory nature of research. Source Quality ensures answers are grounded in verifiable sources.

5.3 Results

Table 4 presents the main results across all 25 queries. VMAO achieves substantially higher completeness (+35%) and source quality (+58%) compared to Single-Agent. The Static Pipeline improves over Single-Agent but cannot adapt when initial agents return insufficient results. VMAO's verification-driven approach identifies gaps and adaptively replans, leading to more complete answers with better source attribution. The increased resource usage reflects verification overhead, justified by quality improvements.

Figure 3(a) shows a typical token distribution across orchestration phases: execution dominates (61%) as agents invoke tools and process results, while verification and synthesis remain efficient. VMAO demonstrates consistent improvements across all query categories (Figure 3(b)), with the largest gains on Strategic Assessment queries (+53% completeness), which require synthesizing information across multiple dimensions. Performance Analysis queries show more modest gains, as these often have well-defined data sources that even single agents can locate.

In our experiments, most queries ($>$75%) terminate via resource-based conditions (diminishing returns, max iterations, or token budget), reflecting conservative thresholds that prioritize thoroughness

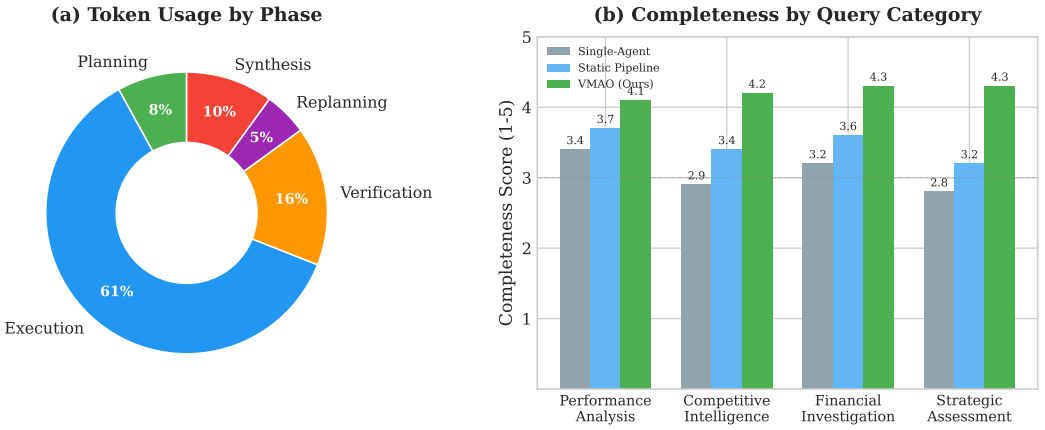

Figure 3: (a) Token usage breakdown by orchestration phase for a typical query. Execution dominates at 61%, while verification and synthesis remain efficient. (b) Completeness scores by query category across methods. VMAO shows consistent improvements, with largest gains on Strategic Assessment (+53%).

Table 4: Comparison of orchestration methods on market research tasks. Completeness and Source Quality are co-scored by LLM and human evaluators (1–5 scale, higher is better).

Method	Completeness	Source Quality	Avg Tokens	Avg Time (s)
Single-Agent	3.1	2.6	100K	165
Static Pipeline	3.5	3.2	350K	420
VMAO (Ours)	**4.2**	**4.1**	850K	900

over speed. These parameters are configurable for deployments requiring faster responses or lower costs.

Evaluation Limitations. We acknowledge three caveats: (1) 25 queries is a modest evaluation set without reported confidence intervals, (2) the LLM judge (Opus 4.5) belongs to the same model family as the execution model (Sonnet 4.5), potentially introducing shared biases despite human review, and (3) the Static Pipeline baseline tests verification and replanning jointly without a component-level ablation. We view the current evaluation as a meaningful signal of the framework's potential, while acknowledging that larger-scale evaluation with independent judges would strengthen the conclusions.

6 Discussion

Unlike skill-based systems (*e.g.*, AutoGPT plugins) that invoke capabilities sequentially within a single agent, VMAO offers explicit DAG decomposition for interpretable plans, parallel execution reducing latency, verification-driven iteration for progressive refinement, and cross-agent synthesis with source attribution. The LLM-based verification serves as a principled coordination signal—assessing whether collective results satisfy the query—decoupling coordination from agent implementation.

When Does Verification Help Most? The largest gains from verification-driven replanning appear on open-ended, multi-dimensional queries (Strategic Assessment: +53% completeness) where initial decomposition inevitably misses relevant aspects. For narrower queries with well-defined data sources (Performance Analysis), single agents already locate most relevant information, and the marginal benefit of replanning is smaller. This suggests verification is most valuable when the query space is difficult to fully characterize upfront—precisely the setting where static pipelines fail. We also observe that the majority of replanning actions are *retries* of incomplete sub-questions rather than introduction of entirely new ones, indicating that agent execution variance (tool failures, insufficient search results) is a larger contributor to gaps than poor initial decomposition.

Limitations. Our framework has several limitations beyond the evaluation caveats noted in Section 5.3. LLM-based verification may miss subtle factual errors or hallucinations, as it evaluates completeness rather than accuracy—the verifier can confirm that a claim is present and sourced, but cannot independently establish its truth. Poor query decomposition can propagate errors downstream: if the planner misframes a sub-question, the verifier may accept a well-sourced but irrelevant answer. The system's 8.5× token cost relative to a single agent (850K vs. 100K tokens) may be prohibitive for latency-sensitive or cost-constrained settings. Finally, all experiments use a single model family (Claude); the framework's effectiveness with other LLM families remains untested.

Transferability and Future Work. The core components—DAG decomposition, verification, and replanning—are domain-agnostic and should transfer to domains like legal discovery or scientific literature review with appropriate agent and tool configuration. Future directions include learning-based stop conditions trained on execution traces, component-level ablation studies to isolate the contribution of each framework element, evaluation with diverse model families, and human-in-the-loop verification for high-stakes queries.

7 Conclusion

We presented VMAO, a framework that coordinates specialized LLM agents through a Plan-Execute-Verify-Replan loop. On 25 market research queries, VMAO improves answer completeness from 3.1 to 4.2 and source quality from 2.6 to 4.1 (1–5 scale) compared to single-agent baselines, with the largest gains on open-ended queries that require multi-dimensional synthesis. Our results suggest that orchestration-level verification—where an independent model evaluates whether collective agent results satisfy the original query—is an effective coordination mechanism for multi-agent systems. Key open questions remain around component-level contributions, generalization across model families and domains, and scalable evaluation methodology. We will release the implementation upon publication.

References

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2023.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

OpenAI. Introducing deep research. *OpenAI Blog*, 2025. URL `https://openai.com/index/introducing-deep-research/`.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

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

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2023b.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

A Prompt Templates

We provide simplified versions of the core prompts. Each follows a structured format with input specifications, decision rules, and JSON output schemas.

Planning Prompt

You are a query planner. Decompose complex queries into sub-questions for specialized agents.

Input: Original query, conversation context, available agents

Planning Rules:
- RAG First: Always search internal knowledge base first or in parallel
- Maximize Parallelism: Execute independent questions simultaneously
- Minimize Dependencies: Only when results feed into other questions
- Be Specific: Clear, answerable scope for each question

Sub-question Fields: id, question, agent_type, dependencies, priority, context_from_deps, verification_criteria

Output: JSON with sub_questions array and explanation

Verification Prompt

Verify if the sub-question has been adequately answered with proper metadata.

Input: Sub-question, verification criteria, result, dependency results

Evaluation Criteria:

– Completeness: All aspects of question addressed?
– Evidence Quality: Multiple sources? Cross-referenced?
– Metadata: Source attribution (filename/URL/date) present?
– Specificity: Concrete facts/numbers vs vague claims?
– Contradictions: Conflicts between sources?

Output: JSON with verification_status (complete/partial/incomplete), completeness_score (0–1), missing_aspects, confidence, recommendation (accept/retry/escalate)

Replanning Prompt

Determine next actions based on verification results.

Input: Original query, execution plan, completed/incomplete results, iteration count

Critical Rule: MUST include ALL incomplete sub-question IDs in retry list.

Decision Logic:

– completeness $>$ 0.8: Proceed to synthesis (done)
– Incomplete results exist: Add ALL to retry_sub_questions
– completeness 0.5–0.8: Add new_sub_questions to fill gaps
– Contradictions found: Add queries targeting different sources
– iterations $\geq$ max: Return empty lists (done)

Output: JSON with retry_sub_questions, new_sub_questions, explanation

Synthesis Prompt

Synthesize results from multiple agents into a concise, well-cited answer.

Input: Original query, sub-question results, verification summary

Required Structure:

1. Executive Summary (2–3 sentences with key metrics)
2. Key Findings (5–8 bullets with source citations)
3. Analysis (2–3 paragraphs connecting insights)
4. Conclusions (confidence level and limitations)

Citation Format: [source - section/URL, metadata]

Output: JSON with answer, key_findings, confidence, sources, gaps

B CONFIGURATION PARAMETERS

Table 5 lists the default configuration parameters used in our experiments. These can be tuned for different quality-latency tradeoffs.

Table 5: Configuration parameters for VMAO orchestration

Parameter	Default	Description
max_iterations	3	Maximum replanning iterations
token_budget	1M	Maximum tokens before stopping
ready_threshold	0.8	Completeness ratio for synthesis
high_confidence	0.75	Confidence threshold for early stop
diminishing_returns	0.05	Minimum improvement to continue
max_concurrent	3	Parallel agent executions
agent_timeout	600s	Per-agent timeout