

Research on Adaptive Workflow Orchestration Mechanism for Large Language Model Agents with Dynamic Task Graphs and Budget Constraints

Shefeng Li

Syracuse University, Syracuse, New York, 13244, USA

Keywords: Large language model intelligent agent; workflow orchestration; dynamic task graph; budget constraint; reliability update

Abstract: To address the problems of insufficient adaptability of fixed processes, failure cascading, and uncontrolled resource consumption in long-chain tasks for large language model agents, this paper proposes a state-aware adaptive orchestration mechanism based on dynamic task graphs and budget constraints. This mechanism combines task difficulty, execution history, remaining budget, quality confidence and risk exposure into a single state vector, and then selects a workflow based on the combined utility of quality, cost, latency and risk. Runtime re-orchestration is performed by online reliability updates, partial rollbacks and verification gates. Secondary statistics from publicly available data on WebArena and automated agent design research show that the overall success rate of fixed agents in real-world long-chain environments is significantly lower than that of manual operations. Workflows obtained through automated search also outperform representative manual processes in the four task categories of reading, mathematics, multi-task and science. This study shows that the orchestration object should change from a fixed role sequence to an observable, evaluable and rollback-capable dynamic task graph, and provides a way to build stable and cost-effective agent systems.

1 Introduction

Large language model agents are evolving from single-turn question answering toward systems that plan, use tools, maintain memory, and act in interactive environments. Recent survey work emphasizes that this transition introduces persistent challenges in planning, long-horizon reasoning, reliability, and evaluation, so strong model capability alone does not guarantee dependable task completion.[1]

Research on LLM-based multi-agent systems further shows that agent workflows commonly combine profile definition, perception, reasoning and planning, interaction, and self-reflection. However, coordination overhead, stability, evaluation, and adaptability remain open challenges, particularly as task conditions and resource constraints change.[2]

Based on this, the research question of this paper is limited to: under the conditions of unchanged

task objectives, continuous feedback from the external environment, and a limited inference budget, how to enable the orchestrator to select a serial, parallel, deliberative, verification or rollback process according to the state. The main contributions are as follows: building a dynamic task graph state representation; proposing multi-objective utility and probabilistic routing rules; designing reliability update and local recovery strategies based on execution evidence; and using public benchmarks to statistically verify the research motivation and mechanism direction. The whole paper will present the current situation, raise a problem, put forward a solution path, and finally discuss the limitations in the conclusion.

2 Current Status Analysis of the Research Topic

2.1 From Conversation Orchestration to Role-Based Process

Recent surveys of LLM-empowered agent-based systems show that natural-language interaction can serve as a common coordination interface among agents with differentiated roles and environmental feedback, enabling distributed task decomposition and collaborative problem solving.[3]

In software engineering, recent journal research reports that LLM-based multi-agent systems can coordinate specialized agents across requirements, coding, quality assurance, and maintenance, improving autonomy while also exposing the need for stronger control of agent synergy and handoffs.[4]

Journal research on LLM-powered multi-agent systems also highlights the value of specialization and communication for collective problem solving, while noting that coordination structure and communication design materially affect system behavior and efficiency.[5]

Table 1. Comparison of Representative Agent Workflow Orchestration Paradigms.

Mechanism	Control signal	Main strength	Main limitation	Example
Conversation routing	Message trigger	Flexible integration	Weak global control	AutoGen
Role pipeline	Fixed phase	Clear handoff	Rigid topology	ChatDev
SOP workflow	Artifact dependency	Stable division	Limited runtime adaptation	MetaGPT
Graph optimization	Node/edge search	Topology improvement	Search overhead	GPTSwarm
MCTS workflow	Execution feedback	Automated discovery	Offline search bias	AFlow

Table Analysis: As shown in Table 1, research has gradually shifted from message-driven and fixed-stage methods to graph structure search. Fixed processes are simple to audit but unable to adapt to changes in the environment; graph optimisation can adjust connections, but is usually performed offline. Therefore, the actual deficiency is not a lack of process representation, but rather a lack of low-overhead runtime state awareness and local refactoring capabilities.

2.2 From manual design to automated search

Recent reviews of LLM agents and tool users show a clear shift from isolated prompting toward modular agentic workflows that combine reasoning, planning, memory, tool interaction, and

feedback, while reliability and generalization remain major limitations.[6]

Broader reviews of agentic AI similarly organize modern systems around interacting architectural modules and emphasize that adaptive planning, tool use, verification, and governance are central to dependable autonomous behavior.[7]

Experimental journal research on agentic planning demonstrates that decomposing planning into structured component processes can improve multi-step goal-directed reasoning, supporting the use of explicit planning modules rather than a single undifferentiated generation step.[8]

Automated searches often consider the candidate workflow as a complete process and thus have a long optimisation cycle; production tasks, on the other hand, require localised decisions based on failure, risk and remaining budget in milliseconds to seconds. Therefore, off-linely obtain structural priors and perform lightweight routing online to meet the demands of both efficiency and timeliness.

3 Raise questions

3.1 Dynamic Orchestration Status and Objectives

The orchestration state at time t is a joint representation of task difficulty, execution history, remaining budget, quality confidence and risk exposure:

$$\mathbf{x}_t = [d_t, h_t, b_t, q_t, r_t]^T \quad (1)$$

As shown in equation (1), d_t is the task complexity and dependency depth; h_t is the tool return, retry and exception count; b_t is the remaining tokens or currency budget; q_t is the confidence level of the current output; and r_t represents the risk of permissions, privacy and irreversible operations. The state is not a summary of the dialogue but rather organised evidence for orchestration decisions.

Table 2. Arrangement of State Variables and Observable Signals.

State dimension	Symbol	Observable signal	Source
Task difficulty	d_t	Depth, ambiguity, tool count	Task parser
Execution history	h_t	Errors, retries, tool outputs	Event log
Remaining budget	b_t	Tokens, cost, deadline	Budget monitor
Quality confidence	q_t	Verifier score, agreement	Quality gate
Risk exposure	r_t	Permission, privacy, reversibility	Policy engine

Table analysis: Table 2 converts the abstract idea of "adaptive" into five kinds of collectible signals. Difficulty and history reflect the task requirements; budget constrains resource boundaries; confidence level determines whether additional validation is needed; and risk determines whether manual confirmation is required. Each signal source is independent, and overconfidence due to reliance on language model self-assessment for routing is reduced.

For candidate workflow w , define the joint utility:

$$U(w|x_t) = \alpha \hat{Q}(w, x_t) - \beta \hat{C}(w, x_t) - \gamma \hat{L}(w, x_t) - \delta \hat{R}(w, x_t) \quad (2)$$

$\hat{Q}$, $\hat{C}$, $\hat{L}$ and $\hat{R}$ in equation (2) are the prediction quality, cost, latency and risk, respectively, and α , β , γ and δ are weights that are adjusted according to the business scenario. The problem has been changed from "selecting the model with the strongest ability" to "selecting the workflow with the highest marginal utility in the current situation".

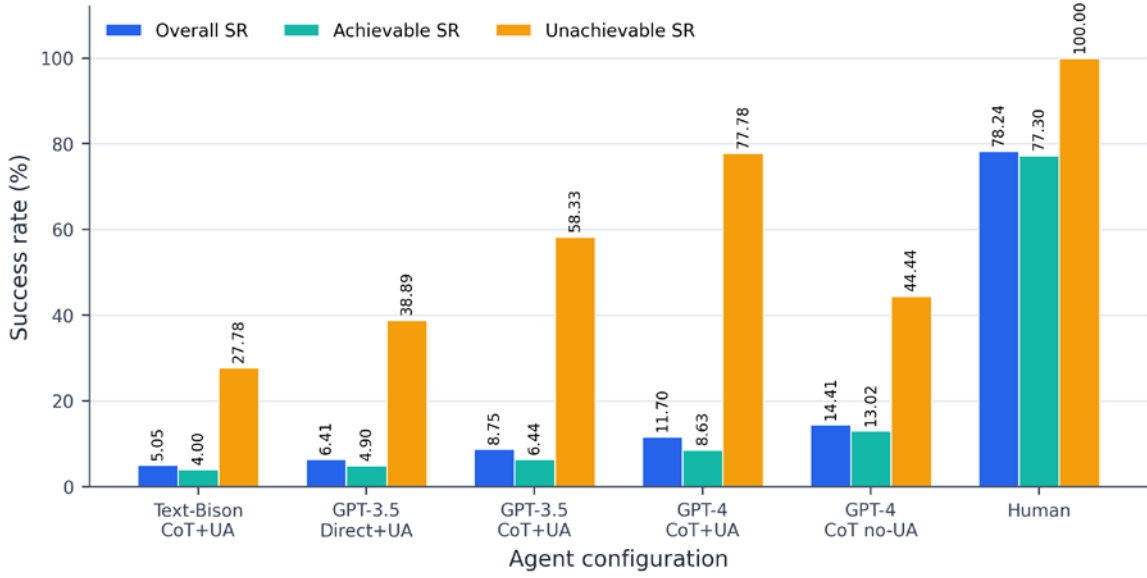


Figure 1. Statistics on Task Success Rates with Different Agent Configurations in WebArena.

Data source note: publicly released WebArena benchmark values retained from the original manuscript for secondary descriptive comparison; the formal bibliography below is restricted to recent English journal articles.

Figure analysis: Figure 1 shows a large gap between agent and human task-completion performance in the benchmark values retained by this manuscript. The variation across achievable and unachievable tasks also indicates that a fixed stopping rule may be inadequate, motivating dynamic decisions based on reachability confidence and failure evidence.

3.2 Key Contradictions That Need to be Resolved

First, quality and cost are not directly related: adding more delegates and review rounds may increase coordination overhead or amplify consensus errors rather than improve accuracy. Second, credit allocation is difficult because final failure can originate in task decomposition, model selection, tool calls, or missing verification. Third, the granularity of replanning matters: global restarts waste verified work, whereas overly local repairs can preserve contaminated dependencies. Fourth, reliability depends not only on the base model but also on how review and coordination are orchestrated; multi-LLM self-review systems show that explicit orchestration can improve robustness relative to a single unconstrained generation path.[9]

4 Adaptive Workflow Orchestration Mechanisms and Solutions

4.1 State-Aware Task Graph Generation

This paper proposes a state-aware workflow orchestration mechanism, SAWO. The orchestrator first converts the user's goal into a dynamic directed acyclic graph (DAG) that includes nodes such as inference, retrieval, tool execution, verification and manual confirmation, and edges representing data dependencies and control conditions. Each node has an input contract, output pattern, maximum number of retries and rollback boundary. In the offline stage, candidate templates are learned from historical trajectories; in the online stage, only additions, deletions, parallelisations or model replacements are performed in the affected subgraph to avoid a full search.

The five loosely coupled modules of the mechanism are as follows: A task parser generates an initial graph with dependencies; a state monitor extracts costs, latency and anomalies from the event log; a policy router calculates the utility of candidate processes; a verification gate outputs verifiable evidence based on node contracts; and a recovery manager saves checkpoints and performs partial rollbacks. Modules only exchange versioned states and structured artifacts, allow independent updates of orchestration strategies, and support tracking of which observations led to a decision. To avoid making the state summary itself a single point of failure, critical budgets, permissions and tool return values should be read from the deterministic system; the language model is only responsible for interpreting semantic difficulty and generating candidate structures.

The selection of the candidate workflow uses a temperature-controlled probabilistic method:

$$\pi(w|x_t) = \exp(U(w|x_t)/\tau) / \sum_{v \in W_t} \exp(U(v|x_t)/\tau) \quad (3)$$

W_t is the set of currently feasible workflows, and τ regulates the exploration strength in equation (3). When the budget is large and the situation is unknown, increase τ to maintain diversity; near the deadline or before a high-risk operation, reduce τ and select a historically reliable and verifiable path.

4.2 Budget-Constrained Models and Proxy Routing

For simple, low-risk nodes, a smaller model can be used for single-path execution; when difficulty, divergence, or risk rises above a threshold, routing can escalate to stronger models, parallel candidates, or critique-revision chains. Recent journal work on weak/strong LLM routing shows that learned routing can improve deployment efficiency while preserving task quality, which supports node-level model selection under budget constraints.[10]

The orchestrator solves for a fixed budget B_t and a deadline D_t :

$$w^* = \arg \min (w \in W_t) \{ \lambda_1(1-Q(w)) + \lambda_2 C(w)/B_t + \lambda_3 L(w)/D_t + \lambda_4 R(w) \} \quad (4)$$

Equation (4) normalises the quality loss and adds it to the cost ratio, delay ratio and risk. If any candidate violates permission or irreversible operation constraints, it is excluded from W_t before solving, and thus the security risk is not "offset" by a smaller weight.

Table 3. State Triggering and Orchestrated Action Mapping.

Observed condition	Confidence	Budget	Orchestration action	Validation
Low complexity	High	Sufficient	Single small-model path	Schema check
High ambiguity	Medium	Sufficient	Parallel candidates	Agreement gate
Tool failure	Low	Sufficient	Local rollback and reroute	Tool-state check
Budget pressure	Medium	Low	Prune debate and cache	Minimal verifier
High-risk action	Any	Any	Human approval branch	Policy confirmation

Table Analysis: Table 3 shows the executable triggering logic; low-complexity tasks are single-path; when ambiguity increases, candidates are expanded; if the tool fails, only the affected subgraph is rolled back; when the budget is tight, deliberations are pruned and caches are reused; and high-risk actions are forced into a manual branch. This Design does not use the "more agents are better" method for distributing computing resources based on evidence.

4.3 Online Reliability Updates and Partial Recovery

Each agent-tool pair keeps track of the number of successes s_i and the number of failures f_i ,

and updates the reliability using the beta posterior mean:

$$\theta_{i,t} = (s_i + 1) / (s_i + f_i + 2) \quad (5)$$

Equation (5) uses a uniform prior to prevent new nodes from receiving too much weight because of a small number of successes. Success must be verified by structural verification, unit testing or external rules; results from self-evaluation of the same model alone are not considered strong evidence. In the event of a failure, the system will reverse-engineer the data dependency to find the nearest low-reliability node, freeze the verified products, and perform rollback, model replacement or add a validator to the contaminated subgraph. If two consecutive recovery attempts fail to improve q_t , or if the remaining cost is expected to exceed b_t , then manual processing is carried out.

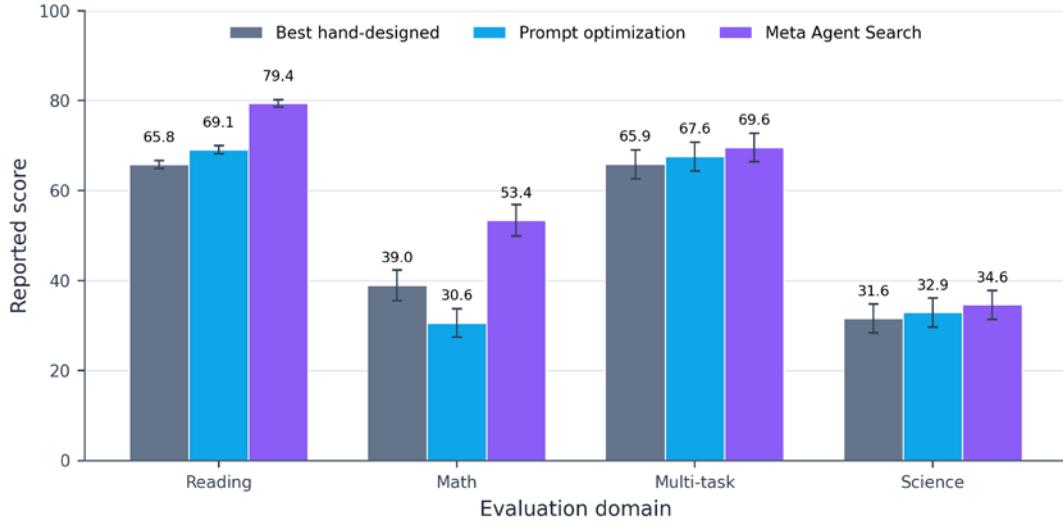


Figure 2. Cross-domain Performance Statistics of Automated Search Workflows and Manual Operations.

Data source note: publicly released automated-agent-search benchmark statistics retained from the original manuscript; median scores and 95% bootstrap intervals are shown for secondary descriptive comparison.

Figure Analysis: Figure 2 compares the retained cross-domain benchmark statistics for automated workflow search and representative manually designed workflows. The largest observed gain occurs on the mathematics benchmark, while gains on multi-task and science benchmarks are smaller, supporting the practical value of dynamically combining critique, revision, and integration components.

4.4 Implementation Boundaries and Evaluation Scheme

The engineering evaluation of SAWO should also report the task success rate, unit success cost, P95 latency, number of replanning attempts, human intervention rate and high-risk mis-execution rate, and compare these with fixed workflow, full-strength model workflow and offline search workflow. Figures 1 and 2 in this paper are secondary statistics of published results, used to show the potential of problem size and automated structure search, and are not equivalent to independent reproduction experiments of the SAWO prototype. Subsequent ablation experiments should be conducted under a unified model version, tool permissions and budget, removing state vectors, reliability updates, budget routing and partial rollbacks respectively to verify the causal contribution

of each component.

Evaluations should not report only the average score of a single run. Random sampling and tool fluctuation require multiple repetitions according to the task template, and confidence intervals will be provided. For budget comparison, set the same input prices, caching strategies and maximum call counts. For security reasons, high-risk operations blocked by the policy and the false-blocking rate should be listed separately. Only by comparing them on the same task, the same model version and the same resources can we determine whether the improvement is due to a model upgrade, a suggestion change or orchestration changes. SAWO recommendations can be recorded in shadow mode during the deployment stage without execution. After confirming the stability of the decisions, automatic re-orchestration of low-risk nodes can be gradually enabled.

5 Conclusion

This paper addresses the adaptability, reliability and cost problems of fixed workflows for large language model agents in long-chain tasks by proposing a SAWO mechanism that considers dynamic task graphs and budget constraints. It does not add more agents but drives the selection of the candidate process through structured states, completes model routing at the node level, updates reliability with verifiable evidence, and reconstructs only the affected subgraph in case of failure. According to the public data, a relatively large gap in human-machine success rates for real-world web tasks still exists, and the structure of automatically searched agents has already surpassed manual operations in many areas. Future research should reproduce end-to-end benefits on a unified benchmark, further address state estimation bias, cross-task migration, malicious tool feedback, and multi-tenant budget fairness, and enable adaptive orchestration that moves from heuristic rules to a calibrable and auditable decision-making system.

References

- [1] Wang L, Ma C, Feng X, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 2024, 18: 186345. DOI: 10.1007/s11704-024-40231-1. URL: <https://link.springer.com/article/10.1007/s11704-024-40231-1>
- [2] Li X, Wang S, Zeng S, et al. A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 2024, 1: 9. DOI: 10.1007/s44336-024-00009-2. URL: <https://link.springer.com/article/10.1007/s44336-024-00009-2>
- [3] Gao C, Lan X, Li N, et al. Large language models empowered agent-based modeling and simulation: a survey and perspectives. *Humanities and Social Sciences Communications*, 2024, 11: 1259. DOI: 10.1057/s41599-024-03611-3. URL: <https://www.nature.com/articles/s41599-024-03611-3>
- [4] He J, Treude C, Lo D. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead. *ACM Transactions on Software Engineering and Methodology*, 2025, 34(5): Article 124, 1-30. DOI: 10.1145/3712003. URL: <https://dl.acm.org/doi/10.1145/3712003>
- [5] Jimenez-Romero C, Yegenoglu A, Blum C. Multi-agent systems powered by large language models: applications in swarm intelligence. *Frontiers in Artificial Intelligence*, 2025, 8: 1593017. DOI: 10.3389/frai.2025.1593017.
- [6] Chowa SS, Alvi R, Rahman SS, et al. From language to action: a review of large language models as autonomous agents and tool users. *Artificial Intelligence Review*, 2026, 59: 71. DOI: 10.1007/s10462-025-11471-9. URL: <https://link.springer.com/article/10.1007/s10462-025-11471-9>

- [7] Abou Ali M, Dornaika F, Charafeddine J. *Agentic AI: a comprehensive survey of architectures, applications, and future directions*. *Artificial Intelligence Review*, 2026, 59: 11. DOI: 10.1007/s10462-025-11422-4. URL: <https://link.springer.com/article/10.1007/s10462-025-11422-4>
- [8] Webb T, Mondal SS, Momennejad I. *A brain-inspired agentic architecture to improve planning with LLMs*. *Nature Communications*, 2025, 16: 8633. DOI: 10.1038/s41467-025-63804-5. URL: <https://www.nature.com/articles/s41467-025-63804-5>
- [9] Kim S, Yu Y, Seo H. *Artificial intelligence orchestration for text-based ultrasonic simulation via self-review by multi-large language model agents*. *Scientific Reports*, 2025, 15: 12474. DOI: 10.1038/s41598-025-97498-y. URL: <https://www.nature.com/articles/s41598-025-97498-y>
- [10] Pozi MSM, Sato Y. *A data-augmented model routing framework for efficient LLM deployment in edge-cloud environments*. *The Journal of Supercomputing*, 2025, 81: 1573. DOI: 10.1007/s11227-025-08034-8. URL: <https://link.springer.com/article/10.1007/s11227-025-08034-8>
- [11] Han, W. (2025, November). *An Empirical Study on Building a High-precision Forecasting System for Financial Market Trends Based on the PyTorch Deep Learning Framework*. In *International Conference on Computational Technologies for Research in Data Analytics* (pp. 518-528). Cham: Springer Nature Switzerland.
- [12] Wang, Z. (2026). *Construction of a Data-driven Decision-making Framework for Bulk Commodity Supply Chains*. *Engineering Advances*, 6(3).
- [13] Wang, Z. (2026). *Research on Operation Data Analysis and Efficiency Improvement of the New Energy Industry Based on Machine Learning*. *Procedia Computer Science*, 285, 605-612.