WHAT MATTERS FOR AGENT REPLACEMENT? LOW-ORDER REPLACEABILITY IN MULTI-AGENT WORKFLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Replacing an agent in an established multi-agent workflow is inherently context-dependent: the same agent may help in one workflow but hurt in another. This seemingly suggests that predicting the effect of a replacement requires modeling the workflow context in detail. We ask whether this complexity is actually necessary. Specifically, how much information is actually needed to predict the effect of an unseen agent replacement before execution? We find that Agent Replacement exhibits *Low-Order Replaceability*: most of its predictive value is captured by first-order Agent Performance and second-order Agent Relations learned from historical workflows. In a controlled sequential LLM-based multi-agent system, Agent Performance provides a transferable prediction baseline, while most finer-grained agent and contextual information adds limited value. Agent Relations provide the strongest additional signal, increasing the Pearson correlation between predicted and observed replacement effects from 0.552 to 0.643. Moreover, only 9 of 90 candidate relations are identified as stable, yet they retain nearly the full predictive performance of the complete relation model. This low-order structure also translates into better replacement decisions. Replacements selected using Agent Performance increase the average success rate from 46.86% to 51.59%, while incorporating Agent Relations further increases it to 53.17%. These results suggest that although replacement effects depend on workflow context, the predictive structure needed to guide replacement decisions can be substantially simpler than the workflow itself.

1 INTRODUCTION

LLM workflows composed of multiple role-specialized agents are increasingly used for complex reasoning tasks (Li et al., 2024; Zhang et al., 2025c; Tran et al., 2025; Li et al., 2023), where different agents are responsible for functions such as planning, solving, verification, and information organization (Wu et al., 2024; Tran et al., 2025; Dang et al., 2025; Hong et al., 2024). Once a workflow has been established, changes in task requirements, new agents, or model updates may require modifying its agents (Hu et al., 2025; Zhang et al., 2025a; Zhuge et al., 2024). When the desired change is local, re-optimizing the entire workflow can be unnecessarily costly. We therefore focus on *Agent Replacement*, where the agent at a single position is replaced while the rest of the workflow remains unchanged.

The challenge is that replacement effects are context-dependent: the same agent may improve one workflow but hurt another. A candidate replacement can be evaluated either by executing the modified workflow or by predicting its performance from a detailed workflow representation, as recent work has done for complete agentic workflows (Trirat et al., 2026). Both are costly: exhaustive execution scales with the replacement space—a workflow of length 8 with 10 agents already admits 72 single-point replacements—while detailed prediction requires modeling multiple levels of workflow context and becomes increasingly difficult as workflows grow more complex (Trirat et al., 2026). This leads to a more fundamental question: *how much information is actually needed to predict the effect of an unseen agent replacement before execution?*

Recent work has increasingly shifted from global workflow optimization toward finer-grained diagnosis and modification. Automated workflow optimization methods search over prompts, modules, and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

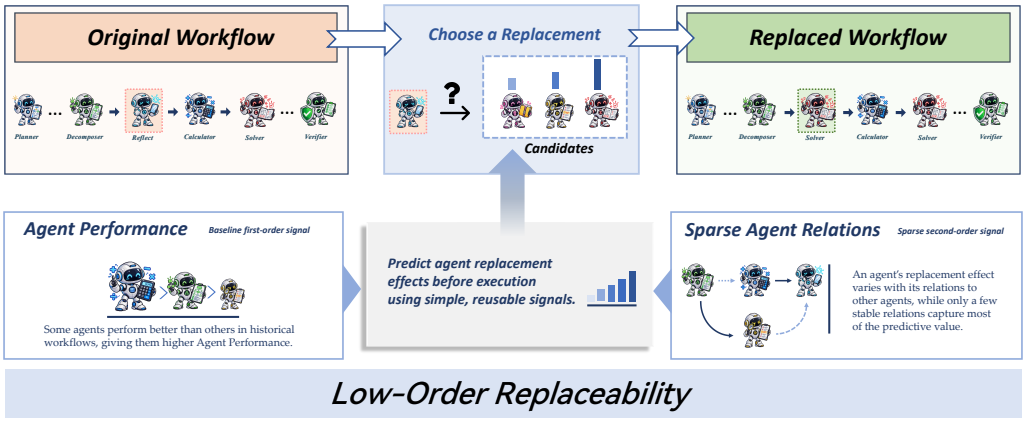


Figure 1: Overview of Low-Order Replaceability in Agent Replacement. Agent Performance provides a reusable cross-workflow baseline, while a small number of Agent Relations refine predictions for the current workflow.

interaction structures (Hu et al., 2025; Zhang et al., 2025a; Zhuge et al., 2024), while component-level methods identify influential or failing agents and use such signals for local optimization and intervention (Zhang et al., 2025b; Chen et al., 2025; Lu et al., 2026; Ma et al., 2026a). Recent studies have also begun to examine agent substitution and replacement (Wang & Cui, 2026; Gao et al., 2026; Yang et al., 2026). However, for heterogeneous replacements across agents with different functional roles, it remains unclear whether unseen replacement effects can be predicted before execution and what information is sufficient for doing so.

To study this problem, we construct a controlled sequential multi-agent system and conduct over 200,000 end-to-end workflow executions across multiple experimental settings. We progressively examine the predictive value of Agent Performance, contextual information, pairwise Agent Relations, and then evaluate whether this information can guide replacement decisions in practice. Figure 1 illustrates the replacement problem and the low-order predictive structure revealed by our analysis.

Our results reveal *Low-Order Replaceability*: most of the predictive value for agent replacement is captured by first-order Agent Performance and second-order Agent Relations. Agent Performance provides a stable baseline for predicting unseen replacement effects, while most finer-grained agent and contextual information adds limited value. Agent Relations provide the strongest additional signal, increasing the Pearson correlation from 0.552 to 0.643. This relational value is highly concentrated: only 9 of 90 candidate relations are identified as stable, yet they retain nearly the full predictive performance of the complete relation model. These low-order signals also improve actual replacement decisions, raising the average success rate from 46.86% to 53.17%. Together, these results show that although replacement effects depend on workflow context, most of their predictable structure is surprisingly low-order. These findings also hold across different models, datasets, workflow lengths, and role-pool sizes (Appendix E.1).

Our contributions are threefold.

- We discover and characterize *Low-Order Replaceability*: most of the predictive value for local agent replacement is captured by transferable first-order agent performance and a small number of stable second-order relations.
- We develop a controlled local-perturbation framework that connects statistical effect estimation, out-of-sample prediction, and executed replacement decisions, providing an end-to-end evaluation of whether historical workflow information generalizes into actual replacement gains.
- We formulate heterogeneous single-point agent replacement as a pre-execution prediction problem, enabling unseen replacement effects to be estimated from historical workflows rather than through workflow re-optimization or repeated end-to-end evaluation.

2 Related Work

Multi-Agent Workflow Optimization and Local Editing. Early representative systems such as ChatDev (Qian et al., 2024) and MetaGPT (Hong et al., 2024) widely adopted role specialization, staged collaboration, and structured workflows to organize multiple agents for complex tasks. Building on these frameworks, works such as GPTSwarm (Zhuge et al., 2024) and AFlow (Zhang et al., 2025a) further treat the agent workflow itself as an optimization target, improving system structures through graph search, program search, or automated design. Subsequent methods extend this direction to joint prompt–topology optimization, transferable orchestration experience, and performance prediction for complete workflow configurations (Hu et al., 2025; Zhou et al., 2026; Lin et al., 2026b; Trirat et al., 2026). Recent studies further refine the optimization granularity toward component-level diagnosis and local editing (Ma et al., 2026a; Lin et al., 2026a; Zhang et al., 2026b). For example, JudgeFlow (Ma et al., 2026b) identifies high-responsibility workflow blocks from execution traces, while FlowSteer (Zhang et al., 2026a) continuously performs local editing actions through interaction with an executable workflow environment. These studies show that agentic workflow optimization has gradually expanded from full-structure search to finer-grained component-level modification.

Agent Attribution and Component Substitution. Agent attribution studies identify agents or components that substantially affect system performance through failure and contribution attribution (Zhang et al., 2025b; Chen et al., 2025; Ma et al., 2026a; Zhang et al., 2026b), Shapley-based methods (Ebrahimi et al., 2025; Li et al., 2026), leave-one-out analysis (Cui et al., 2025), and related importance estimation (Lu et al., 2026). These signals can then be used to support component removal, online optimization, and model allocation or substitution (Wang et al., 2025; Chen et al., 2026; Xia et al., 2026; Yang et al., 2026).

Recent studies have begun to show that agent replacement is context-dependent (Wang & Cui, 2026). However, existing substitution studies mainly consider replacing the underlying model under a fixed role (Wang & Cui, 2026; Yang et al., 2026) or testing interchangeability among agents assigned the same role (Gao et al., 2026). Related studies on multi-agent collaboration further suggest why replacement effects may depend on the surrounding workflow. Davidson et al. (2026) and Furuya & Kitagawa (2026) show that collaboration outcomes can vary with execution order, collaboration partners, and pairwise compatibility, while ShapleyFlow (Yang et al., 2026) shows that component value can depend on the configuration of other components.

Together, these findings suggest that agent replacement may depend not only on the agent being replaced, but also on its workflow context. Yet for replacements across agents with different functional roles, it remains unclear whether unseen replacement effects can be predicted before execution and what information is sufficient for doing so. This work addresses this gap by studying heterogeneous agent replacement and examining how Agent Performance and Agent Relations can be used to predict replacement effects and guide replacement decisions.

3 Problem Definition

Agent Replacement. Let $\mathcal{A}$ denote the agent set. A sequential workflow of length L is represented as

$$C = (r_1, \ldots, r_L), \qquad r_k \in \mathcal{A}.$$

For a task distribution $\mathcal{Q}$, we define the workflow utility as

$$U(C) = \mathbb{E}_{q \sim \mathcal{Q}} \Pr(Y = 1 \mid \mathrm{do}(C), q),$$

where $Y = 1$ indicates successful task completion. We study a *single-point agent replacement*, which replaces agent a at position k with another agent b while keeping the task, execution mechanism, and all other agents unchanged. We denote this replacement as

$$\tau = (k, a \to b),$$

and the resulting workflow as C^τ. The replacement effect is

$$\Delta U_\tau = U(C^\tau) - U(C).$$

Our goal is to predict ΔU_τ before executing C^τ and use this prediction to guide replacement decisions.

Low-Order Replaceability. We use *Low-Order Replaceability* to describe the empirical property that most of the predictive value for Agent Replacement can be captured by first-order Agent Performance and second-order Agent Relations, without modeling the full workflow in detail.

Agent Performance. The most direct first-order signal is Agent Performance across historical workflows. Let $n_a(C)$ denote the number of occurrences of agent a in workflow C, and define its binary presence as $x_a(C) = \mathbb{I}(n_a(C) \geq 1)$. For each agent a, we estimate its historical performance using

$$\operatorname{logit} P(Y = 1) = \alpha_a + \beta_a x_a(C),$$

where β_a captures the overall association between the presence of agent a and workflow success across historical workflows. We refer to β_a as the *Agent Performance* of a. It characterizes the agent's cross-workflow statistical performance rather than its standalone capability.

Given these agent-level estimates, we define the Agent Performance score of a workflow as

$$S_{\text{agent}}(C) = \sum_{a \in \mathcal{A}} \beta_a x_a(C).$$

For a replacement $\tau = (k, a \to b)$, the corresponding first-order change is

$$\Delta S_{\text{agent}}(\tau) = S_{\text{agent}}(C^\tau) - S_{\text{agent}}(C),$$

which provides the baseline signal for predicting the true replacement effect ΔU_τ.

The same agent substitution can nevertheless have different effects across workflow contexts. We therefore examine what task or workflow information can improve replacement prediction beyond $\Delta S_{\text{agent}}(\tau)$.

Agent Relations. Among possible incremental information, we focus particularly on ordered pairwise agent relations. Since sequential workflows induce directional information flow, we represent pairwise context using ordered relations rather than unordered co-occurrence. For two distinct agents u and v, we define

$$z_{uv}(C) = \mathbf{1}[\exists\, i < j :\ r_i = u,\ r_j = v],$$

which indicates whether the ordered relation $u \to v$ is present in workflow C. This relation captures ordered co-occurrence without requiring adjacency.

A replacement can change not only the Agent Performance signal but also the set of Agent Relations present in the workflow. Let $\mathbf{z}(C)$ denote the vector of ordered pairwise relation indicators. The relational change induced by replacement τ is

$$\Delta \mathbf{z}_\tau = \mathbf{z}(C^\tau) - \mathbf{z}(C).$$

We study whether these relational changes provide predictive information beyond Agent Performance, and whether such information is concentrated in a small number of stable relations.

Research Questions. We study three questions:

1. Can Agent Performance learned from historical workflows reliably predict unseen replacement effects?
2. Beyond Agent Performance, what additional information improves replacement prediction?
3. Is the predictive value of agent relations concentrated in a small number of stable relations, and can these relations improve actual replacement decisions?

4 METHOD

We study agent replacement through three stages: Statistical Effect Estimation, Replacement Prediction, and Replacement Decision Evaluation. We first extract reusable information from historical workflow executions, then evaluate whether this information predicts new replacement effects, and finally test whether it can guide actual replacement decisions.

4.1 STATISTICAL EFFECT ESTIMATION

We estimate Agent Performance and other candidate information from historical random workflow executions using logistic regression, with workflow success as the outcome.

Agent Performance is estimated from the association between each Agent's presence and workflow success. We further examine richer information including Agent occurrence counts, position, task properties, workflow composition, and ordered pairwise Agent Relations.

For Agent Relations, we jointly model Agent presence and ordered pairwise relations:

$$\text{logit}\, P(Y=1) = \beta_0 + \sum_a \beta_a x_a(C) + \sum_{a \neq b} \gamma_{ab} z_{ab}(C),$$

where γ_{ab} captures the additional effect associated with relation $a \to b$. We further identify statistically stable relations and examine whether relational information is concentrated in a small subset. Detailed model specifications are provided in the appendix.

4.2 REPLACEMENT PREDICTION

We evaluate the estimated information on an independent set of replacements. For each replacement, the models predict its Replacement Effect from the change in Agent-level and workflow-level information between C and C^τ.

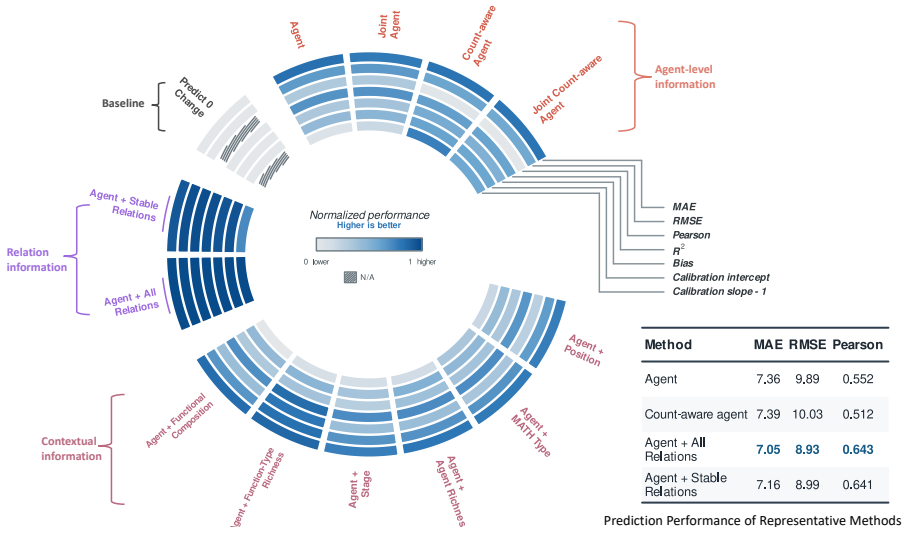
Prediction quality is evaluated using MAE, RMSE, and Pearson correlation. Each original and replaced workflow is independently executed three times, and the main analysis focuses on replacement effects above the estimated execution-noise level. Detailed prediction procedures and noise estimation are provided in the appendix.

4.3 REPLACEMENT DECISION EVALUATION

Finally, we test whether the learned information can improve actual replacement decisions on a separate evaluation set. The Agent Relation Model used in this evaluation uses only the stable relations identified by a cluster-aware FDR correction (Benjamini & Hochberg, 1995), referred to as Cluster-FDR[1]. For each workflow, the Agent Performance Model and the Agent Relation Model evaluate the same Candidate Replacement space and select the candidate with the highest predicted gain.

The selected replacements are then executed and compared with the unreplaced workflows. This allows us to test whether Agent Performance can guide replacement and whether relational information provides further improvement in actual workflow performance.

[1] Cluster-FDR denotes BH-FDR correction applied to p-values obtained from task-cluster-robust Wald tests.



Method	MAE	RMSE	Pearson
Agent	7.36	9.89	0.552
Count-aware agent	7.39	10.03	0.512
Agent + All Relations	**7.05**	**8.93**	**0.643**
Agent + Stable Relations	7.16	8.99	0.641

Prediction Performance of Representative Methods

Figure 2: Replacement-prediction performance across different information types. Each sector represents a model, with darker shading indicating better normalized performance across seven evaluation metrics. The accompanying table reports exact results for representative methods.

5 The Low-Order Structure of Agent Replacement

This section examines whether agent replacement exhibits Low-Order Replaceability. We first evaluate first-order Agent Performance as a transferable prediction baseline, then examine the incremental value of richer contextual and second-order relational information, and finally test whether the relational gain is concentrated in a small number of stable relations.

We evaluate the proposed framework on sequential workflows of length $L = 8$ constructed from 10 role-specialized agents sharing Qwen2.5-14B-Instruct (Yang et al., 2025) as the backbone. The main experiments use MATH-486 (Zhang et al., 2025a), with complete system, data-split, and implementation details provided in Appendix A. All results are evaluated on an independent unseen replacement set. The main analysis uses the above-noise replacement subset identified through repeated runs, containing 100 execution chains. Noise estimation and results on the full sample are provided in Appendix D.

5.1 Agent Performance Provides a Stable Replacement Baseline

The most direct predictive signal is agent performance across historical workflows. The zero-change baseline achieves an MAE of 8.93 pp and an RMSE of 11.66 pp. Using only agent performance reduces the MAE to 7.36 pp and the RMSE to 9.89 pp, while reaching a Pearson correlation of 0.552. This shows that historical agent performance can transfer to the prediction of new local replacement outcomes.

To test whether a finer description of agent presence provides additional information, we use a Count-Aware Agent Model that additionally distinguishes how many times each agent appears in the workflow. It achieves an MAE of 7.39 pp, an RMSE of 10.03 pp, and a Pearson correlation of 0.512, providing no improvement over the basic Agent Performance model. These results indicate that agent performance already captures the main agent-level signal for replacement prediction.

5.2 Agent Relations Provide the Largest Predictive Gain

On top of the agent performance baseline, we further incorporate position, task properties, workflow composition, and agent relations, and compare their incremental predictive value for replacement prediction. As shown in Figure 2, the gains from different factors vary substantially. Most finer-grained agent representations, including jointly modeled Agent presence and occurrence-count

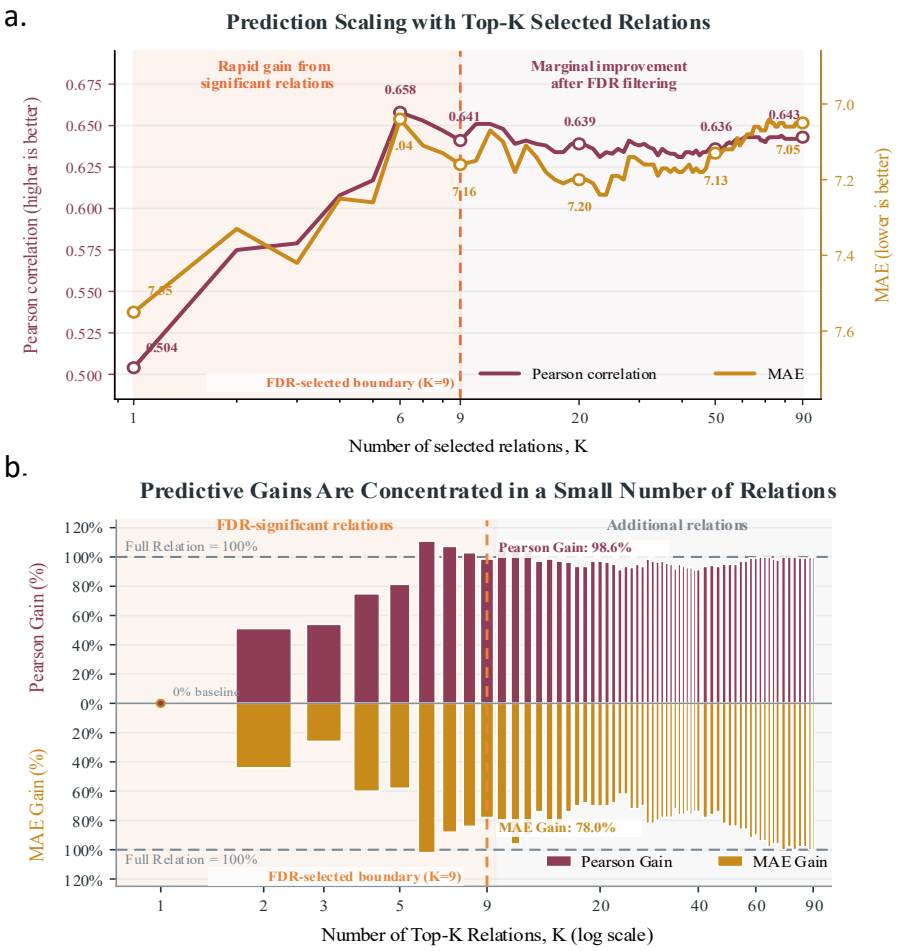


Figure 3: Prediction scaling with the number of selected relations. (a) Pearson correlation and MAE across Top-K relation models. (b) Predictive gains as K increases, showing that most of the improvement is captured near the FDR-selected boundary at $K = 9$.

representations, as well as contextual factors, provide limited additional value over the Agent Performance baseline. *Joint Agent*, *Count-aware Agent*, *Position*, *MATH Type*, *Agent Richness*, *Stage*, and *Functional Composition* achieve similar performance to the baseline. Among these factors, *Function-Type Richness* provides a stronger incremental gain, reaching a Pearson correlation of 0.617. The complete numerical results for all model variants are reported in Figure 7 in Appendix D.3.

In contrast, ordered pairwise agent relations provide the clearest improvement. For two different agents A and B, the relation $A \rightarrow B$ indicates that both appear in the workflow and that A occurs before B. With 10 agents, this yields 90 candidate ordered relations. Adding all relations to the agent performance baseline reduces the MAE to 7.05 pp and the RMSE to 8.93 pp, while increasing the Pearson correlation to 0.643, outperforming all other candidate factors. These results show that the effect of a local replacement depends not only on the replaced agent itself, but also on its relations with other agents in the current workflow. Task properties and workflow composition provide some additional information, but agent relations contribute stronger incremental predictive value. Agent performance captures a shared cross-workflow replacement signal, while relational information further characterizes variation across specific collaborative contexts.

A closer examination of relation effects shows that this incremental value is not evenly distributed across all 90 candidate relations. After Cluster-FDR correction, only 9 relations remain significant, while most relations do not exhibit stable effects. This suggests that the main predictive information in the relation model is concentrated in a small number of stable relations.

5.3 Relational Gains Are Concentrated in a Small Number of Stable Relations

Based on the above result, we further examine whether retaining all relations is necessary. We refit the model using only the 9 stable relations selected by Cluster-FDR. The resulting MAE is 7.16 pp, the RMSE is 8.99 pp, and the Pearson correlation is 0.641, which is nearly identical to the 7.05 pp, 8.93 pp, and 0.643 obtained using all 90 relations. This indicates that most of the predictive gain of relational information can be retained by a small number of stable relations.

The full Top-K analysis in Figure 3 shows the same pattern. Prediction performance improves rapidly after only a few high-ranked relations are retained and then largely saturates. The Top-5 model already reaches a Pearson correlation of 0.617, while the best observed point occurs at $K = 6$, with an RMSE of 8.87 pp and a Pearson correlation of 0.658. At $K = 9$, performance remains similar, with an RMSE of 8.99 pp and a Pearson correlation of 0.641, close to both the observed optimum and the full 90-relation model. The Top-K results indicate that relational predictive gains saturate quickly after a small number of high-ranked relations are included.

Taken together, these results establish Low-Order Replaceability. First-order Agent Performance provides the main cross-workflow baseline signal, while most finer-grained agent and contextual information contributes limited additional value. Second-order pairwise agent relations provide the strongest incremental predictive information, and most of this relational gain is captured by only a small number of stable relations. Explicitly adding all 720 ordered third-order relations does not further improve prediction under the available data budget (Pearson 0.559 with pairwise and triple relations, versus 0.643 with pairwise relations alone; Appendix D.6).

6 From Replacement Prediction to Replacement Decisions

Section 5 identifies Low-Order Replaceability in agent replacement: agent performance provides a transferable baseline, while a small number of stable agent relations capture most of the additional predictive value. We next examine whether this predictive structure translates into better replacement decisions.

6.1 Historical Signals Improve Actual Replacement Decisions

We conduct replacement decision evaluation on an independent evaluation set. For each workflow, the Agent Performance Model and the Agent Relation Model are given the same candidate replacement space and each selects one replacement for re-execution. The Agent Relation Model uses only the stable relations identified in Section 5. Table 1 summarizes the resulting replacement performance.

Table 1: Average success rates, gains over the unreplaced baseline, and workflow-level replacement outcomes.

Strategy	Success Rate	Gain over Baseline	Workflow Improved / Tied / Declined
Unreplaced Baseline	46.86%	—	—
Agent Performance Model	51.59%	+4.73 pp	69%/5%/26%
Agent Relation Model	53.17%	+6.31 pp	78%/2%/20%

The unreplaced workflows achieve an average success rate of 46.86%. The Agent Performance Model increases this to 51.59%, a gain of +4.73 pp. The Agent Relation Model further increases the success rate to 53.17%, corresponding to +6.31 pp over the unreplaced baseline and +1.58 pp over the Agent Performance Model. The latter difference has a task-cluster 95% CI of $[+0.68, +2.50]$ pp, indicating that the predictive gain from relational information also translates into actual replacement gains.

This improvement is also broadly distributed across workflows. Among 100 workflows, the Agent Performance Model produces 69 improvements, 26 decreases, and 5 ties, whereas the Agent Relation Model produces 78 improvements, 20 decreases, and 2 ties. Thus, incorporating relational information increases the number of workflows that benefit from replacement while reducing negative replacements. More detailed workflow- and task-level analyses are provided in Appendix E.2.

6.2 Relation-Guided Decisions Perform Better When Models Disagree

To isolate the role of relational information, we divide the 100 workflows according to whether the two models select the same replacement. They agree on 27 workflows and select different replacements on the remaining 73.

In the decision-agreement group, the average success rates are 51.70% for the Agent Performance Model and 52.30% for the Agent Relation Model, a difference of $+0.59$ pp with a task-cluster 95% CI of $[-0.81, +2.00]$ pp. In the decision-disagreement group, the Agent Relation Model achieves 53.49%, compared with 51.54% for the Agent Performance Model, yielding a larger gain of $+1.95$ pp with a 95% CI of $[+0.94, +2.98]$ pp.

These results show that relational information is most useful when it changes the ranking of candidate replacements. Agent performance provides the cross-workflow baseline ranking, while relational information adjusts this ranking for the current workflow.

The Low-Order Replaceability identified in Section 5 therefore extends beyond offline prediction. Agent performance provides a reusable baseline for replacement decisions, while relational information refines these decisions for the current workflow.

7 Discussion

Our results reveal a property of Agent Replacement that we call *Low-Order Replaceability*. Agent Performance captures a transferable cross-workflow tendency of candidate agents, while a small number of stable Agent Relations characterize how replacement effects vary across workflow contexts. This suggests that the information required to predict a replacement effect can be substantially simpler than the workflow in which it occurs. Even across different task conditions, role compositions, and workflow configurations, most of the observed predictive value can be captured without modeling the full workflow in detail.

Low-Order Replaceability also provides a practical basis for workflow optimization. Historical executions can be reused not only to evaluate past workflows, but also to inform future replacement decisions. Agent Performance provides a reusable baseline for ranking candidate agents, while relational information adjusts this ranking according to the current workflow. The resulting replacement gains are also observed across different backbone models, datasets, workflow lengths, and role-pool sizes (Appendix E.1). Together, these signals can help prioritize or filter candidate replacements before expensive end-to-end evaluation, and may support more efficient workflow modification and search.

Low-Order Replaceability should be interpreted as an empirical predictive property under the settings studied in this work, rather than as a claim that higher-order interactions are absent. Explicitly modeling higher-order relations is substantially more data-intensive, and our exploratory analysis of ordered three-agent relations is provided in Appendix D.6. More broadly, the scope of our findings is limited to sequential multi-agent systems with shared context, fixed role pools, and single-point replacement, where Agent Relations are represented through ordered pairwise co-occurrence. More complex settings such as dynamic routing, richer communication topologies, higher-order relational structures, and multi-point replacement remain open for further study.

8 Conclusion

This work studies whether historical workflow executions can support unseen agent replacement in LLM-based multi-agent workflows. We identify Low-Order Replaceability: most of the predictive value for local agent replacement is captured by transferable Agent Performance and a small number of stable Agent Relations. Agent performance provides a cross-workflow baseline, while agent relations contribute the strongest additional predictive value, with this gain also translating into better replacement decisions. Overall, our results show that Agent Replacement can be predicted and guided using compact and reusable information, without modeling the full workflow anew.

AI USE STATEMENT

In this work, we used generative AI tools to polish writing, including improving the grammar, clarity, and fluency of author-written text. We did not use generative AI tools to generate research ideas, design the methodology, conduct experiments, analyze results, or determine the scientific conclusions of this work. All AI-assisted edits were reviewed by the authors to ensure that they preserved the intended technical meaning. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the implementation and reproducibility materials in the accompanying supplementary code archive. The archive includes the experimental configurations, agent prompts and role definitions, context-access and state-update mechanisms, tool interfaces, dependency information, and a small-scale set of actual replacement runs with representative workflow outputs, execution results, replacement outcomes, and method comparisons. The large-scale raw execution logs amount to tens of gigabytes and are therefore not included. They can be regenerated using the provided implementation, configurations, and reproduction scripts. Instead, the provided runs are intended to facilitate inspection of the experimental pipeline, system behavior, and replacement procedures. Detailed descriptions of the experimental setup, data splits, model specifications, and evaluation procedures are provided in the appendix.

REFERENCES

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 01 1995. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1995.tb02031.x. URL https://doi.org/10.1111/j.2517-6161.1995.tb02031.x.

Jianming Chen, Yawen Wang, Junjie Wang, Xiaofei Xie, Jun Hu, Qing Wang, and Fanjiang Xu. Understanding Individual Agent Importance in Multi-Agent System via Counterfactual Reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 39(15):15785–15794, April 2025. doi: 10.1609/aaai.v39i15.33733. URL https://doi.org/10.1609/aaai.v39i15.33733.

Yulang Chen, Haoxuan Peng, Jinyan Liu, Zichen Wen, Dongrui Liu, and Linfeng Zhang. AgentSlimming: Towards Efficient and Cost-Aware Multi-Agent Systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30064–30086, San Diego, California, United States, July 2026. Association for Computational Linguistics (ACL). doi: 10.18653/v1/2026.acl-long.1387. URL https://aclanthology.org/2026.acl-long.1387/.

Yue Cui, Liuyi Yao, Zitao Li, Yaliang Li, Bolin Ding, and Xiaofang Zhou. Efficient Leave-One-Out Approximation in LLM Multi-Agent Debate Based on Introspection, 2025. URL https://arxiv.org/abs/2505.22192.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-Agent Collaboration via Evolving Orchestration. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 165025–165059. Curran Associates, Inc., 2025. doi: 10.52202/085713-5502. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f1320d2e2842169c6fc89dcbd80e94d0-Abstract-Conference.html.

Tim R. Davidson, Adam Fourney, Saleema Amershi, Robert West, Eric Horvitz, and Ece Kamar. The Collaboration Gap: Exploration and Benchmarking of Open-World Agentic Cooperation, 2026. URL https://arxiv.org/abs/2511.02687. Accepted at the Conference on Language Modeling (COLM 2026); proceedings forthcoming.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *Technical Report*, 2026.

Sana Ebrahimi, Mohsen Dehghankar, and Abolfazl Asudeh. An Adversary-Resistant Multi-Agent LLM System via Credibility Scoring. In Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh (eds.), *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 1676–1693, Mumbai, India, December 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics (ACL). ISBN 979-8-89176-298-5. doi: 10.18653/v1/2025.ijcnlp-long.90. URL https://aclanthology.org/2025.ijcnlp-long.90/.

Kotaro Furuya and Yuichi Kitagawa. The Geometry of Dialogue: Graphing Language Models to Reveal Synergistic Teams for Multi-Agent Collaboration, January 2026. URL https://arxiv.org/abs/2510.26352. Oral presentation at the AAAI 2026 Workshop on LLM-Based Multi-Agent Systems (LaMAS).

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, 2025.

Jianxin Gao, Tianyi Yu, Linna Deng, Runze Li, and Zining Wang. Testing Interchangeability in LLM Agent Teams, September 2026. URL https://arxiv.org/abs/2609.05279.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. In *International Conference on Learning Representations (ICLR)*, pp. 23247–23275, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Shengran Hu, Cong Lu, and Jeff Clune. Automated Design of Agentic Systems. In *International Conference on Learning Representations (ICLR)*, pp. 21344–21377, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 51991–52008. Curran Associates, Inc., 2023. doi: 10.52202/075280-2264. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A Survey on LLM-Based Multi-Agent Systems: Workflow, Infrastructure, and Challenges. *Vicinagearth*, 1(1):9, October 2024. ISSN 3005-060X. doi: 10.1007/s44336-024-00009-2. URL https://doi.org/10.1007/s44336-024-00009-2.

Yanming Li, Xuelin Zhang, WenJie Lu, Ziye Tang, Maodong Wu, Haotian Luo, Tongtong Wu, Zijie Peng, Hongze Mi, Yibo Feng, Naiqiang Tan, Chao Huang, Lian Peng, and Li Shen. Who Deserves the Reward? SHARP: Shapley Credit-Based Optimization for Multi-Agent System, 2026. URL https://arxiv.org/abs/2602.08335.

Fulin Lin, Shaowen Chen, Ruishan Fang, Hongwei Wang, and Tao Lin. Stop wasting your tokens: Towards efficient runtime multi-agent systems. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026a. URL https://openreview.net/forum?id=pzFhtpkabh.

Hehai Lin, Qi Yang, and Chengwei Qin. Skill-MAS: Evolving Meta-Skill for Automatic Multi-Agent Systems, June 2026b. URL https://arxiv.org/abs/2606.18837.

Mingyu Lu, Yushan Huang, Chris Lin, and Su-In Lee. Agents That Matter: Optimizing Multi-Agent LLMs via Removal-Based Attribution, May 2026. URL https://arxiv.org/abs/2605.27621.

Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Dover: Intervention-driven auto debugging for LLM multi-agent systems. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026a. URL https://openreview.net/forum?id=mrEK16Jy6h.

Zihan Ma, Zhikai Zhao, Chuanbo Hua, Federico Berto, and Jinkyoo Park. JudgeFlow: Agentic Workflow Optimization via Block Judge, January 2026b. URL https://arxiv.org/abs/2601.07477.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186. Association for Computational Linguistics (ACL), 2024. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-Agent Collaboration Mechanisms: A Survey of LLMs, January 2025. URL https://arxiv.org/abs/2501.06322.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. Multi-view encoders for performance prediction in LLM-based agentic workflows. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7oeKDZsmWp.

Renxiang Wang and Jiaming Cui. Beyond Tier Labels: Role- and Deployment-Dependent Model Substitution in Multi-Call LLM Workflows, August 2026. URL https://arxiv.org/abs/2608.09155.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=y10DM6R2r3.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. AgentDropout: Dynamic Agent Elimination for Token-Efficient and High-Performance LLM-Based Multi-Agent Collaboration. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035, Vienna, Austria, July 2025. Association for Computational Linguistics (ACL). ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1170. URL https://aclanthology.org/2025.acl-long.1170/.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversations. In *First Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Yihan Xia, Taotao Wang, Shengli Zhang, Zhangyuhua Weng, Bin Cao, and Soung Chang Liew. HiveMind: Contribution-Guided Online Prompt Optimization of LLM Multi-Agent Systems. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 40(35):29767–29774, March 2026. doi: 10.1609/aaai.v40i35.40222. URL https://doi.org/10.1609/aaai.v40i35.40222.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 Technical Report, 2025. URL https://arxiv.org/abs/2412.15115.

Yingxuan Yang, Bo Huang, Siyuan Qi, Chao Feng, Haoyi Hu, Yuxuan Zhu, Jinbo Hu, Haoran Zhao, Ziyi He, Xiao Liu, ZongYu Wang, Muning Wen, Lin Qiu, Xuezhi Cao, Xunliang Cai, Yong Yu, and Weinan Zhang. Attribution-Based Analysis and Optimization of Modular Agentic Workflows. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 7247–7264, San Diego, California, United States, July 2026. Association for Computational Linguistics (ACL). ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.359. URL https://aclanthology.org/2026.findings-acl.359/.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating Agentic Workflow Generation. In *International Conference on Learning Representations (ICLR)*, pp. 34040–34077, May 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Mingda Zhang, Wenjin Liu, Tiesunlong Shen, Qika Lin, Rui Mao, Erik Cambria, Xiaoying Tang, and Haoran Luo. FlowSteer: Towards Agents Designing Agentic Workflows via Reinforced Progressive Canvas Editing, February 2026a. URL https://arxiv.org/abs/2602.01664.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599. PMLR, 2025b. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Xueqiang Zhang, Xiaofei Dong, Yiru Wang, Dan Zhang, and Feng Cao. A Survey of Multi-AI Agent Collaboration: Theories, Technologies, and Applications. In *Proceedings of the 2nd Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence (DEAI)*, DEAI '25, pp. 1875–1881, New York, NY, USA, July 2025c. Association for Computing Machinery (ACM). ISBN 979-8-4007-1279-1. doi: 10.1145/3745238.3745531. URL https://doi.org/10.1145/3745238.3745531.

Zhiwei Zhang, Xiaomin Li, Yudi Lin, Hui Liu, Ramraj Chandradevan, Linlin Wu, Minhua Lin, Fali Wang, Xianfeng Tang, Qi He, and Suhang Wang. Unlocking the power of multi-agent LLM for reasoning: From lazy agents to deliberation. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026b. URL https://openreview.net/forum?id=5J6u03ObRZ.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Arik. Multi-Agent Design: Optimizing Agents with Better Prompts and Topologies. In *International Conference on Learning Representations (ICLR)*, pp. 15844–15872, April 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1ab4e0e8f35078109bc78d7b465d306f-Abstract-Conference.html.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language Agents as Optimizable Graphs. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62743–62767. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

A Experimental Setup Details

This section provides additional details on the experimental system, agent roles, workflow construction, candidate information, data splits, and reproducibility settings described in Section 4.

A.1 Experimental System and Tasks

All agents in the main experiments use Qwen2.5-14B-Instruct as the backbone model, with the sampling temperature set to 0.2. Different agents share the same model parameters and are instantiated as different functional modules through role instructions, visible context, available tools, and writable states. Therefore, an agent in this work refers to a complete executable module rather than an individual prompt.

The role pool contains 10 agents covering four functional categories: solving, planning, evaluation, and information processing.

Table 2: Agent role pool used in the main experiments.

Category	Agent	Main Function
Solving	Solver	General problem solving
Solving	SolverMulti	Multi-path or multi-solution problem solving
Solving	SolverOnlyQuery	Independent solving based only on the original query
Planning	Planner	Constructing a solution plan
Planning	Decomposer	Decomposing problems and subtasks
Evaluation	Verifier	Verifying existing answers or reasoning
Evaluation	Critic	Identifying problems in existing reasoning
Evaluation	Reflect	Reflecting on and revising previous reasoning
Information Processing	Compact	Compressing existing context
Information Processing	Calculator	Generating Python code and performing numerical computation

Complete role instructions, context access rules, tool interfaces, and state update mechanisms are provided in the accompanying supplementary code files.

The main experiments use sequential workflows of length $L = 8$. Each position is sampled independently and uniformly with replacement from the agent pool $\mathcal{A}$:

$$C = (r_1, \ldots, r_8), \qquad r_k \overset{\text{i.i.d.}}{\sim} \mathrm{Uniform}(\mathcal{A}).$$

Therefore, the same agent may appear multiple times within a workflow. All agents execute in a fixed order, share the message history and answer state, and can access the original task and outputs from previous agents. No dynamic routing is used during workflow execution, and the subsequent agent sequence is not modified according to intermediate states.

For single-point agent replacement, the agent a at a given position can be replaced by any other agent in the role pool:

$$\mathcal{B}(a) = \mathcal{A} \setminus \{a\}.$$

Therefore, each position has 9 candidate replacements, and replacements across different functional categories are allowed. Replacement prediction and replacement decision evaluation use the same candidate space.

The main experiments use MATH-486 (Zhang et al., 2025a), consisting of the 486 Level-5 problems from the MATH benchmark. Workflow success is determined by the mathematical equivalence between the final answer and the reference answer:

$$Y = \begin{cases} 1, & \text{if the final answer is mathematically equivalent to the reference answer,} \\ 0, & \text{otherwise.} \end{cases}$$

Answer evaluation first applies a rule-based mathematical equivalence checker implemented with reference to the OMNI-MATH evaluation code. When the rule-based checker cannot confirm correctness, a `gemini-2.5-flash-nothinking` judge agent further determines whether the reference answer and candidate answer are mathematically equivalent. The judge agent does not receive the original problem and does not re-solve the task. A sample is marked as correct if either stage confirms the answer as correct.

A.2 Candidate Information and Encoding

Replacement prediction starts from agent performance and further examines whether occurrence count, position, task type, workflow composition, and agent relations provide additional information. All parameters are estimated using only the statistical effect estimation data. Detailed estimation and robustness analyses of the agent-level models are provided in Appendix B, and estimation and selection procedures for the Agent Relation Model are provided in Appendix C.

Let $n_a(C)$ denote the number of occurrences of agent a in workflow C, and define binary agent presence as

$$x_a(C) = \mathbb{I}(n_a(C) \geq 1).$$

The candidate information is encoded as follows.

Table 3: Candidate information and encoding used for replacement prediction.

Information	Agent Representation	Additional Encoding
Agent Performance	Binary presence, fitted separately for each agent	–
Agent Count	$n_a = 1, 2, \geq 3$, fitted separately for each agent	–
Position	Binary agent presence	Agent identity at each of the 8 positions
Stage	Binary agent presence	Agent distribution across Early/Middle/Late stages
MATH Type	Binary agent presence	Problem type and Agent $\times$ Problem Type interaction
Agent Richness	Binary agent presence	Number of distinct agents R
Function-Type Richness	Binary agent presence	Number of functional categories R_F
Functional Composition	Binary agent presence	Counts of agents from each functional category
Ordered Pairwise Relations	Binary agent presence, jointly fitted	90 ordered agent relations

For *Stage*, positions are divided into

$$\text{Early} = \{1, 2, 3\}, \qquad \text{Middle} = \{4, 5, 6\}, \qquad \text{Late} = \{7, 8\}.$$

Agent richness is defined as the number of distinct agents in a workflow:

$$R = \sum_{a \in \mathcal{A}} x_a(C).$$

After dividing the role pool into four functional categories—solving, planning, evaluation, and information processing—function-type richness is defined as the number of functional categories that appear in the workflow:

$$R_F = \sum_{f \in \mathcal{F}} \mathbb{I}(n_f > 0).$$

Functional composition further preserves the number of occurrences of each functional category within the workflow.

For two different agents $a \neq b$, an ordered agent relation is defined as

$$z_{ab}(C) = \mathbb{I}[\exists\, i < j : r_i = a,\ r_j = b].$$

The 10-agent role pool therefore contains $10 \times 9 = 90$ candidate ordered relations. This feature records only whether a appears before b at least once, without requiring adjacency or distinguishing the specific distance between them.

A.3 Data Splits and Three-Stage Experimental Protocol

Before the experiments, the 486 problems in MATH-486 are randomly partitioned without replacement into four disjoint task sets: 100 problems for development, 286 for statistical effect estimation, and the remaining 100 equally divided between replacement prediction and replacement decision evaluation. The development set is used only for preliminary method and experimental pipeline design and is excluded from the final evaluation.

Statistical Effect Estimation. For each of the 286 estimation tasks, we generate 100 random workflows, yielding

$$286 \times 100 = 28{,}600$$

task–workflow records. These data are used to estimate agent-level effects, candidate factors, and agent relations, as well as to perform stable relation selection and model fitting within the estimation set. Tasks in the final evaluation sets are not used for parameter estimation or relation selection.

Replacement Prediction. Replacement prediction uses an independent set of 50 tasks and 200 fixed workflow seeds. For each seed, one position is randomly selected and replaced by another agent from the role pool. All prediction methods are evaluated on exactly the same replacements.

This stage contains

$$50 \times 200 = 10{,}000$$

task–workflow combinations. The workflows before and after replacement are each independently executed three times, and the repeated executions are used to estimate the true replacement effect. The full evaluation uses all 200 workflow seeds, while the main analysis focuses on the above-noise replacement subset identified through repeated runs, containing 100 workflow seeds. Details of noise estimation and results on the full sample are provided in Appendix D.2.

Replacement Decision Evaluation. Replacement decision evaluation uses another independent set of 50 tasks and 100 fixed workflow seeds. For each original workflow, the Agent Performance Model and the Agent Relation Model are given the same candidate replacement space and independently select one single-point agent replacement.

A complete evaluation of each strategy contains

$$50 \times 100 = 5{,}000$$

task–workflow combinations and is independently repeated three times. This stage directly compares system performance after executing the selected replacements, testing whether information estimated from historical workflows can improve actual replacement decisions.

Table 4: Data splits and their uses in the main experiments.

Data Subset	Tasks	Workflow Seeds	Main Use
Development Set	100	–	Method design and experimental pipeline validation
Statistical Effect Estimation	286	100 / task	28,600 records; parameter estimation and relation selection
Replacement Prediction	50	200 / task	3 runs before/after replacement; replacement effect prediction
Replacement Decision Evaluation	50	100 / task	3 runs per strategy; replacement decision evaluation

The four task sets are completely disjoint at the task level. Statistical effect estimation, replacement prediction, and replacement decision evaluation therefore correspond to three independent stages for parameter learning, prediction evaluation, and decision evaluation, respectively.

A.4 Implementation and Reproducibility

All experiments use the same workflow execution framework and access the backbone model through API calls. Different methods use identical model configurations, agent definitions, and workflow execution rules. Statistical models are implemented in Python, with logistic regression and related statistical tests performed using Statsmodels.

All workflow seeds are generated in advance and stored as fixed seeds. Different prediction models and replacement strategies reuse the same original workflows, seed lists, and candidate replacement spaces to ensure consistent paired comparisons.

The complete implementation, experimental configurations, agent prompts, and dependency versions are provided in the accompanying supplementary code files.

B Agent Performance and Agent-Level Robustness Analysis

This section provides additional details on the estimation and use of agent performance in Section 5.1, and examines its robustness to occurrence count, model parameterization, and nonlinear representations.

B.1 Agent Performance Model

For agent a, let $n_a(C)$ denote its number of occurrences in workflow C, and define the binary presence feature as

$$x_a(C) = \mathbb{I}(n_a(C) \geq 1).$$

In the main text, we fit the following model separately for each agent:

$$\text{logit}\, P(Y = 1) = \alpha_a + \beta_a x_a(C).$$

Here, $n_a = 0$ serves as the reference state, and β_a represents the overall statistical association between the presence of agent a and success across random workflows. The 10 agents are fitted independently, and the main text therefore uses β_a as the cross-workflow overall performance of agent a, namely its agent performance.

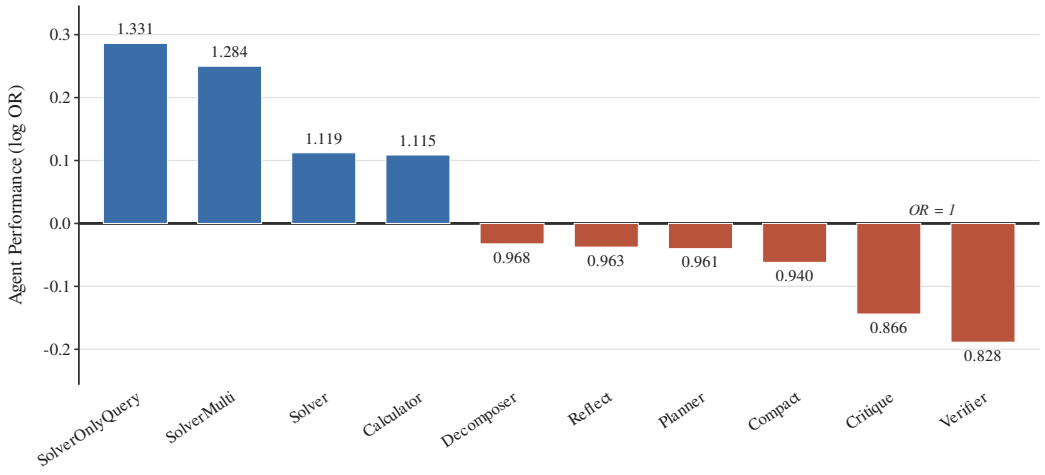


Figure 4: Agent performance estimates for the 10 agents. Bar heights show the binary-presence coefficients β_a on the log-odds scale, while the numeric annotations report the corresponding odds ratios $\exp(\beta_a)$. The horizontal line marks $\beta_a = 0$, or equivalently an odds ratio of 1; blue and red bars indicate positive and negative associations with workflow success, respectively.

Based on these coefficients, we define the agent performance score of a workflow as

$$S_{\text{agent}}(C) = \sum_{a \in \mathcal{A}} \beta_a x_a(C).$$

Here, $S_{\text{agent}}(C)$ is a composite score constructed from the historical performance estimates of individual agents, rather than the linear predictor of a joint logistic regression model. For a candidate replacement τ, the score change is defined as

$$\Delta S_{\text{agent}}(\tau) = S_{\text{agent}}(C^\tau) - S_{\text{agent}}(C).$$

This score uses only changes in agent presence induced by the replacement and does not include occurrence count, position, task properties, workflow composition, or agent relation information.

In replacement decision evaluation, the Agent Performance Model selects

$$\tau^\star_{\text{agent}} = \arg\max_{\tau \in \mathcal{T}(C)} \Delta S_{\text{agent}}(\tau).$$

Since $S_{\text{agent}}(C)$ is identical for all candidate replacements of the same original workflow, this is equivalent to selecting the candidate workflow with the largest $S_{\text{agent}}(C^\tau)$.

B.2 Count-Aware Agent Model

To examine whether explicitly distinguishing agent occurrence counts provides additional information, we further define

$$x_{a,1} = \mathbb{I}(n_a = 1), \qquad x_{a,2} = \mathbb{I}(n_a = 2), \qquad x_{a,3+} = \mathbb{I}(n_a \geq 3).$$

As with agent performance, each agent is fitted independently:

$$\text{logit}\, P(Y = 1) = \alpha_a + \beta_{a,1} x_{a,1} + \beta_{a,2} x_{a,2} + \beta_{a,3+} x_{a,3+},$$

where $n_a = 0$ is the reference state.

The corresponding count-aware workflow score is defined as

$$S_{\text{count}}(C) = \sum_{a \in \mathcal{A}} \left[\beta_{a,1} \mathbb{I}(n_a = 1) + \beta_{a,2} \mathbb{I}(n_a = 2) + \beta_{a,3+} \mathbb{I}(n_a \geq 3) \right],$$

with

$$\Delta S_{\text{count}}(\tau) = S_{\text{count}}(C^\tau) - S_{\text{count}}(C).$$

The Count-Aware Agent model is used only as an agent-level ablation to test whether finer-grained occurrence-count information improves replacement prediction. Replacement decision evaluation continues to use the basic Agent Performance Model as the primary agent-level baseline.

B.3 Separate and Joint Fitting

The main text fits each agent separately to directly characterize its overall performance across random workflows. As a parameterization robustness check, we further include all agent-presence features in a joint model:

$$\text{logit}\, P(Y = 1) = \beta_0 + \sum_{a \in \mathcal{A}} \beta_a x_a.$$

We construct the corresponding joint model for the Count-Aware Agent representation as

$$\operatorname{logit} P(Y=1) = \beta_0 + \sum_{a \in \mathcal{A}} \left(\beta_{a,1} x_{a,1} + \beta_{a,2} x_{a,2} + \beta_{a,3+} x_{a,3+} \right).$$

Table 5 compares the replacement prediction results of separate and joint fitting.

Table 5: Replacement prediction results of separately and jointly fitted agent-level models.

Prediction Set	Model	MAE (pp)	RMSE (pp)	Pearson
Large-change 100	Agent Performance	7.36	9.89	0.552
	Joint Agent Performance	7.44	9.87	0.551
	Count-Aware Agent	7.39	10.03	0.512
	Joint Count-Aware Agent	7.40	10.09	0.507
All 200	Agent Performance	5.24	7.54	0.435
	Joint Agent Performance	5.27	7.55	0.435
	Count-Aware Agent	5.31	7.67	0.413
	Joint Count-Aware Agent	5.27	7.69	0.403

The predictive results of separate and joint fitting are very similar. We therefore use separate fitting in the main text, not because it provides clearly better predictive performance, but because its coefficients directly represent the overall statistical performance of individual agents across random workflows.

B.4 Leave-One-Agent-Out Robustness Check

To examine whether the ranking induced by agent performance depends on the specific representation of logistic coefficients, we further conduct a distribution-level leave-one-agent-out (LOO) analysis.

For agent a, we use the random workflows in the training data satisfying $n_a = 0$ as the leave-out subset and define

$$p_{-a} = \Pr(Y = 1 \mid n_a = 0), \qquad \Delta_a^{\mathrm{LOO}} = p_{\mathrm{all}} - p_{-a},$$

where the empirical success rate over all 28,600 training records is

$$p_{\mathrm{all}} = 0.4528.$$

A positive Δ_a^{LOO} indicates that random workflows without agent a have a lower average success rate. The 95% confidence intervals are obtained using 10,000 cluster-bootstrap resamples at the task level.

Table 6: Agent performance and distribution-level leave-one-agent-out results. β_a denotes the binary-presence log-odds coefficient, while Δ_a^{LOO} and its confidence interval are reported in percentage points (pp).

Agent	β_a	β Rank	Δ_a^{LOO}	95% CI	LOO Rank
SolverOnlyQuery	0.286	1	4.00	$[3.05, 4.95]$	1
SolverMulti	0.250	2	3.50	$[2.57, 4.48]$	2
Solver	0.112	3	1.56	$[0.80, 2.33]$	3
Calculator	0.109	4	1.53	$[0.59, 2.49]$	4
Decomposer	-0.032	5	-0.46	$[-1.11, 0.20]$	5
Reflect	-0.038	6	-0.53	$[-1.23, 0.17]$	6
Planner	-0.040	7	-0.56	$[-1.18, 0.08]$	7
Compact	-0.062	8	-0.87	$[-1.52, -0.22]$	8
Critic	-0.144	9	-2.01	$[-2.72, -1.31]$	9
Verifier	-0.189	10	-2.66	$[-3.41, -1.92]$	10

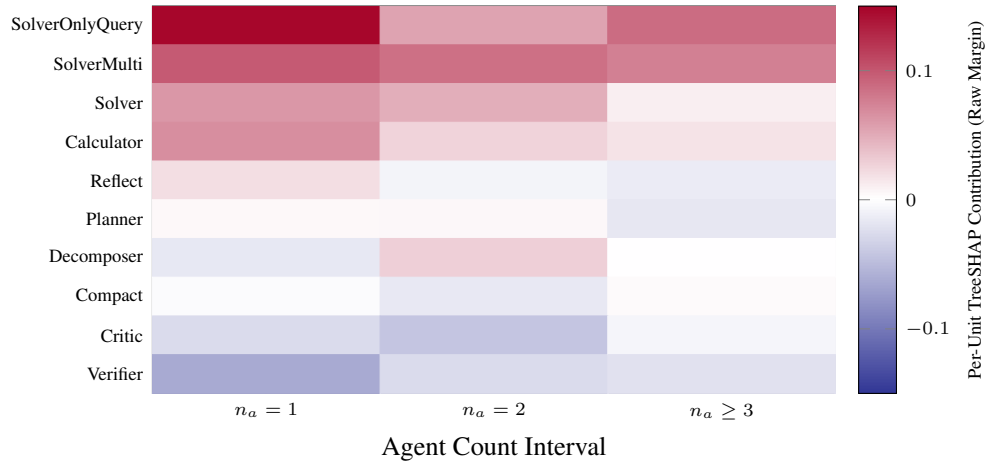


Figure 5: Average per-unit TreeSHAP contributions across different agents and agent count intervals. Red indicates positive contributions, and blue indicates negative contributions. Values are shown on the raw-margin (log-odds) scale. The sample size in each cell ranges from 1,037 to 10,954.

The agent performance coefficients and LOO estimates produce exactly the same ranking over all 10 agents, indicating that the main agent-level ordering does not depend on the specific representation of logistic coefficients.

LOO and agent performance use the same training data, so this result should be interpreted as a parameterization robustness check rather than an independent-sample validation. The LOO confidence intervals for Decomposer, Reflect, and Planner all cross zero, and therefore do not provide clear statistical separation among these three agents. In addition, Δ_a^{LOO} compares distributions of random workflows rather than re-executing the same workflow after removing an agent, and should therefore not be interpreted as a causal deletion effect at the individual-workflow level.

B.5 Nonlinear Agent-Level Robustness Check

To examine whether the three-level count representation in the Count-Aware Agent model misses substantial nonlinear agent-level structure, we train an XGBoost binary classifier using the 10-dimensional agent-count vector of each workflow and perform five-fold cross-fitting with `GroupKFold` at the task level. TreeSHAP values are computed out of fold on the raw-margin (log-odds) scale.

For agent a, within count interval $k \in \{1, 2, 3+\}$, we define the average per-occurrence SHAP contribution as

$$S_{a,k}^{\mathrm{unit}} = \mathbb{E}\left[\frac{\phi_{ia}}{n_{ia}} \,\middle|\, n_{ia} \in B_k \right],$$

where

$$B_1 = \{1\}, \qquad B_2 = \{2\}, \qquad B_{3+} = \{3, 4, \ldots\}.$$

Across the three count intervals, the Spearman correlations between the agent rankings produced by the Count-Aware Logistic Model and TreeSHAP are

$$0.988, \qquad 0.988, \qquad 0.976.$$

The out-of-fold AUC, Brier score, and log loss of XGBoost are

$$0.5581, \qquad 0.2425, \qquad 0.6767.$$

After changing the random seed, the OOF AUC is 0.5587, while the aggregated agent ranking remains exactly the same (Spearman $= 1.000$). As a negative control, randomly permuting labels within each task reduces the OOF AUC to 0.4987.

These results show that a more flexible nonlinear model does not change the main agent-level ranking, indicating that the Count-Aware Agent representation preserves the main count-related structure. At the same time, single-sample prediction based only on agent counts remains limited. We therefore use this analysis only as a representation robustness check rather than as a primary model for replacement prediction.

Because the fixed workflow length satisfies

$$\sum_{a \in \mathcal{A}} n_a = 8,$$

the TreeSHAP results are interpreted only as descriptive attributions within the predictive model, rather than as causal effects of agent count.

C Agent Relation Model, Stable Relation Selection, and Sparsity

This section provides additional details on the estimation of the Agent Relation Model, Stable Relation selection, and Top-K analysis in Sections 5.2 and 5.3. All relation parameters, significance tests, and relation rankings are determined using only the Statistical Effect Estimation data.

C.1 Full Agent Relation Model

For two different agents $a \neq b$, we define the ordered relation

$$z_{ab}(C) = \mathbb{I}[\exists\, i < j : r_i = a,\ r_j = b].$$

This feature records whether agent a appears before agent b at least once in the workflow, without requiring the two agents to be adjacent or distinguishing their specific distance. With 10 agents, there are $10 \times 9 = 90$ candidate ordered relations.

The Full Relation Model jointly estimates binary agent presence and all relation terms:

$$\text{logit}\, P(Y = 1) = \beta_0 + \sum_{a \in \mathcal{A}} \beta_a x_a(C) + \sum_{a \neq b} \gamma_{ab} z_{ab}(C).$$

Here, β_a represents the conditional agent effect after controlling for other agent-presence features and relations, while γ_{ab} represents the additional statistical information provided by the ordered relation $a \to b$ after controlling for agent presence.

This differs from the Agent Performance Model in the main text. Agent Performance estimates the overall cross-workflow performance of each agent separately, whereas the Relation Model jointly estimates agent presence and pairwise relations to isolate the incremental information contributed by relations. γ_{ab} represents a conditional statistical association and is not interpreted as the causal effect of an individual relation.

Figure 6 provides a global view of the fitted source-to-target relation terms. Rows identify the source agent A and columns identify the target agent B. The 90 off-diagonal cells correspond to the ordered relations with $a \neq b$; the 10 diagonal cells are omitted because self-relations are undefined. An odds ratio above 1 indicates a positive conditional association with workflow success, while a value below 1 indicates a negative conditional association, after accounting for agent presence and the other relation terms in the Full Relation Model. The matrix is visibly asymmetric, showing that the fitted association for $A \to B$ can differ substantially from that for $B \to A$ and motivating the use of ordered relations.

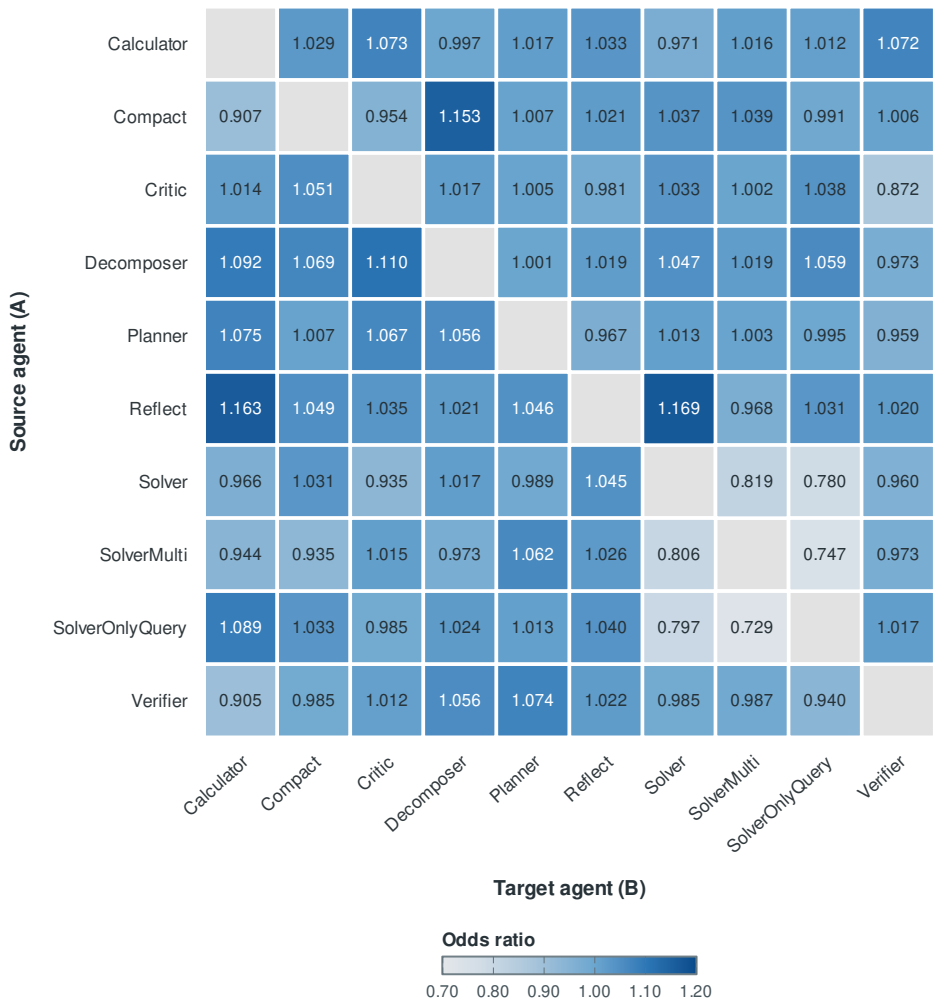


Figure 6: Odds-ratio heatmap of the source-to-target terms fitted by the Full Relation Model. Rows denote source agents and columns denote target agents. The 90 annotated off-diagonal cells represent ordered relations with $a \neq b$; the gray diagonal is left blank because self-relations are undefined. Values above 1 and below 1 indicate positive and negative conditional associations with workflow success, respectively.

C.2 Stable Relation Selection and Refit

Relation selection uses only the 286 training tasks. For the 90 γ_{ab} terms in the Full Relation Model, we perform cluster-robust Wald tests with task as the clustering unit and apply Benjamini–Hochberg FDR correction to all p values. Let q_{ab} denote the corrected value, and define

$$\mathcal{R}_{\text{stable}} = \{(a, b) : q_{ab} \leq 0.05\}.$$

A total of 9 relations satisfy this criterion. After selection, we retain only these relation terms and refit the joint model on the same training data:

$$\text{logit}\, P(Y = 1) = \widetilde{\beta}_0 + \sum_{a \in \mathcal{A}} \widetilde{\beta}_a x_a(C) + \sum_{(a,b) \in \mathcal{R}_{\text{stable}}} \widetilde{\gamma}_{ab} z_{ab}(C).$$

Table 7 reports the Full Model coefficients used during selection, the FDR-adjusted q values, and the coefficients after refitting.

Table 7: The 9 Stable Relations selected by Cluster-FDR and their refitted coefficients.

Relation	Full γ	BH-FDR q	Refit $\widetilde{\gamma}$
SolverOnlyQuery → SolverMulti	-0.3148	1.770×10^{-8}	-0.2954
SolverMulti → SolverOnlyQuery	-0.2913	1.186×10^{-7}	-0.3095
Solver → SolverOnlyQuery	-0.2480	5.231×10^{-6}	-0.2681
SolverOnlyQuery → Solver	-0.2262	8.122×10^{-5}	-0.2072
SolverMulti → Solver	-0.2157	9.941×10^{-5}	-0.2000
Solver → SolverMulti	-0.1979	8.310×10^{-4}	-0.1894
Reflect → Solver	+0.1561	0.0050	+0.1659
Reflect → Calculator	+0.1489	0.0114	+0.1142
Compact → Decomposer	+0.1428	0.0154	+0.0783

All 9 relation coefficients retain their original signs after refitting, with 6 negative and 3 positive coefficients.

As a robustness check for relation selection, using conventional IID standard errors yields 10 FDR-significant relations, while task-cluster standard errors yield 9. All 9 cluster-selected relations are shared by the two sets, giving a Jaccard similarity of 0.900. The only relation removed is CRITIC→VERIFIER. We therefore use the more conservative Cluster-FDR results in the main text.

It is important to distinguish the statistical roles of selection and refitting. The cluster-robust Wald tests and BH-FDR correction in the Full Relation Model are used to determine Stable Relations, while the refitted $\widetilde{\beta}_a$ and $\widetilde{\gamma}_{ab}$ are used for subsequent Replacement Prediction and Replacement Decision Evaluation. Significance after refitting is not treated as a new independent statistical test.

C.3 Relation Ranking and Top-K Analysis

To examine whether relational information is concentrated in a small number of relations, we rank the 90 candidate relations according to the BH-FDR-adjusted q_{ab} values from the Full Relation Model:

$$q_{(1)} \leq q_{(2)} \leq \cdots \leq q_{(90)}.$$

For a given K, we define

$$\mathcal{R}_K = \{r_{(1)}, \ldots, r_{(K)}\}.$$

We then retain only the relations in $\mathcal{R}_K$ and independently refit the model for each $K = 1, \ldots, 90$:

$$\operatorname{logit} P(Y = 1) = \beta_0^{(K)} + \sum_{a \in \mathcal{A}} \beta_a^{(K)} x_a(C) + \sum_{(a,b) \in \mathcal{R}_K} \gamma_{ab}^{(K)} z_{ab}(C).$$

Therefore, the Top-K models are not formed by directly truncating the top K coefficients from the Full Relation Model. Instead, all retained parameters are re-estimated after fixing the relation ranking. The relation ranking is determined entirely from the training data, and the Replacement Prediction data are not used for relation selection.

As a ranking robustness check, the Spearman correlation between $|\gamma_{ab}|$ and $-\log_{10} q_{ab}$ in the Full Relation Model is 0.9822. The two ranking criteria select exactly the same relation set at Top-9. We therefore use the FDR ranking as the formal Top-K ranking criterion in the main text.

Replacement Prediction performance improves rapidly as a small number of highly ranked relations are added and then gradually saturates. The 9 Stable Relations pre-specified by $q \leq 0.05$ retain predictive performance close to that of the Full Relation Model.

C.4 Replacement Score of the Relation Model

For the Stable Relation Model, we define the workflow score as

$$S_{\text{relation}}(C) = \sum_{a \in \mathcal{A}} \widetilde{\beta}_a x_a(C) + \sum_{(a,b) \in \mathcal{R}_{\text{stable}}} \widetilde{\gamma}_{ab} z_{ab}(C).$$

The intercept is identical across all candidate workflows under the same model and is therefore omitted from replacement scoring. For a candidate replacement τ,

$$\Delta S_{\text{relation}}(\tau) = S_{\text{relation}}(C^\tau) - S_{\text{relation}}(C).$$

This change can be decomposed into agent-presence and relation components:

$$\Delta S_{\text{relation}}(\tau) = \Delta S_{\text{presence}}(\tau) + \Delta S_{\text{rel}}(\tau),$$

where

$$\Delta S_{\text{presence}}(\tau) = \sum_{a \in \mathcal{A}} \widetilde{\beta}_a [x_a(C^\tau) - x_a(C)],$$

and

$$\Delta S_{\text{rel}}(\tau) = \sum_{(a,b) \in \mathcal{R}_{\text{stable}}} \widetilde{\gamma}_{ab} [z_{ab}(C^\tau) - z_{ab}(C)].$$

Note that $\Delta S_{\text{presence}}$ uses the jointly estimated $\widetilde{\beta}_a$ from the Stable Relation Model and is therefore not equivalent to ΔS_{agent} from the Agent Performance Model, which is based on separately estimated agent coefficients.

In Replacement Prediction, $\Delta S_{\text{relation}}$ is converted to the prediction scale defined in Appendix D and then used to predict the true Replacement Effect. In Replacement Decision Evaluation, the Agent Relation Model directly selects

$$\tau^\star_{\text{relation}} = \arg\max_{\tau \in \mathcal{T}(C)} \Delta S_{\text{relation}}(\tau).$$

Therefore, the Agent Performance Model ranks candidate replacements using only the overall cross-workflow performance of agents, whereas the Agent Relation Model additionally uses the ordered pairwise relations changed by the replacement in the current workflow.

D Complete Results and Robustness Analysis for Replacement Prediction

This section provides additional details on the Replacement Prediction setup and results in Section 5. We first describe how the scores produced by different models are converted into Replacement Effects on the success-rate scale, then estimate repeated-execution noise, and finally report results on both the full set of 200 workflow seeds and the above-noise replacement subset. The full 200 seeds constitute the complete evaluation set, while the above-noise replacement subset is used to analyze predictive performance when the true effect exceeds the execution-noise scale.

D.1 Prediction Scores and Evaluation Metrics

For a candidate replacement τ, each model first computes the score change between the original and replaced workflows:

$$\Delta S_\tau = S(C^\tau) - S(C).$$

The Agent Performance Model uses binary-presence coefficients estimated separately for each agent. The Full Relation Model and Stable Relation Model instead use jointly estimated binary agent presence and ordered pairwise relations, as defined in Appendices B and C.

All of these scores are constructed from coefficients on the log-odds scale. To convert them to the success-rate scale in a unified manner, we use the observed mean success rate of the original workflow seed, $\widehat{p}_{\tau,0}$, as the baseline and treat ΔS_τ as a log-odds adjustment:

$$\widehat{\Delta U}_\tau = \sigma(\mathrm{logit}(\widehat{p}_{\tau,0}) + \Delta S_\tau) - \widehat{p}_{\tau,0},$$

where $\sigma(\cdot)$ denotes the sigmoid function. $\widehat{p}_{\tau,0}$ is computed from three repeated executions of the workflow seed over the 50 Replacement Prediction tasks.

The predicted $\widehat{\Delta U}_\tau$ is compared with the observed Replacement Effect ΔU_τ. MAE and RMSE are reported in percentage points, while Pearson correlation measures the relative variation across different replacements. This conversion does not use any post-replacement execution results and does not involve additional calibration.

D.2 Execution Noise and the Noise-Aware Subset

Replacement Prediction uses 50 independent test tasks and 200 fixed workflow seeds. For each seed, one position is randomly selected and replaced with another agent through a single-point agent replacement. The workflows before and after replacement are each executed independently three times.

Let ΔU_{sr} denote the Replacement Effect of workflow seed s in repeated run r. We first compute the sample variance across the three repeated runs for each seed and then average it over the 200 seeds:

$$\widehat{\sigma}_1^2 = \frac{1}{200}\sum_{s=1}^{200} \mathrm{Var}_{r=1,2,3}(\Delta U_{sr}) = 38.520 \text{ pp}^2.$$

The estimated noise standard deviation for a single repeated run is therefore

$$\widehat{\sigma}_1 = 6.206 \text{ pp}.$$

The final observed Replacement Effect is computed as the mean over three independent runs, giving an estimated noise standard deviation of

$$\widehat{\sigma}_3 = \sqrt{\frac{38.520}{3}} = 3.583 \text{ pp}.$$

Bootstrap resampling over workflow seeds gives a 95% CI of $[3.313, 3.863]$ pp for $\widehat{\sigma}_3$.

In addition to the complete 200-seed evaluation, we define replacements satisfying

$$|\Delta U_\tau| > 3.583 \text{ pp}$$

as the above-noise replacement subset, containing 100 workflow seeds. This subset is used to analyze the predictive value of different structural information when the true Replacement Effect exceeds the noise scale associated with the mean over three repeated runs.

The above-noise replacement subset changes only the range over which results are summarized. It does not alter the replacement samples, model training, parameter estimation, or relation selection. Because the subset is defined using the observed Replacement Effect, it should be interpreted as a noise-aware diagnostic rather than as an independently constructed test set.

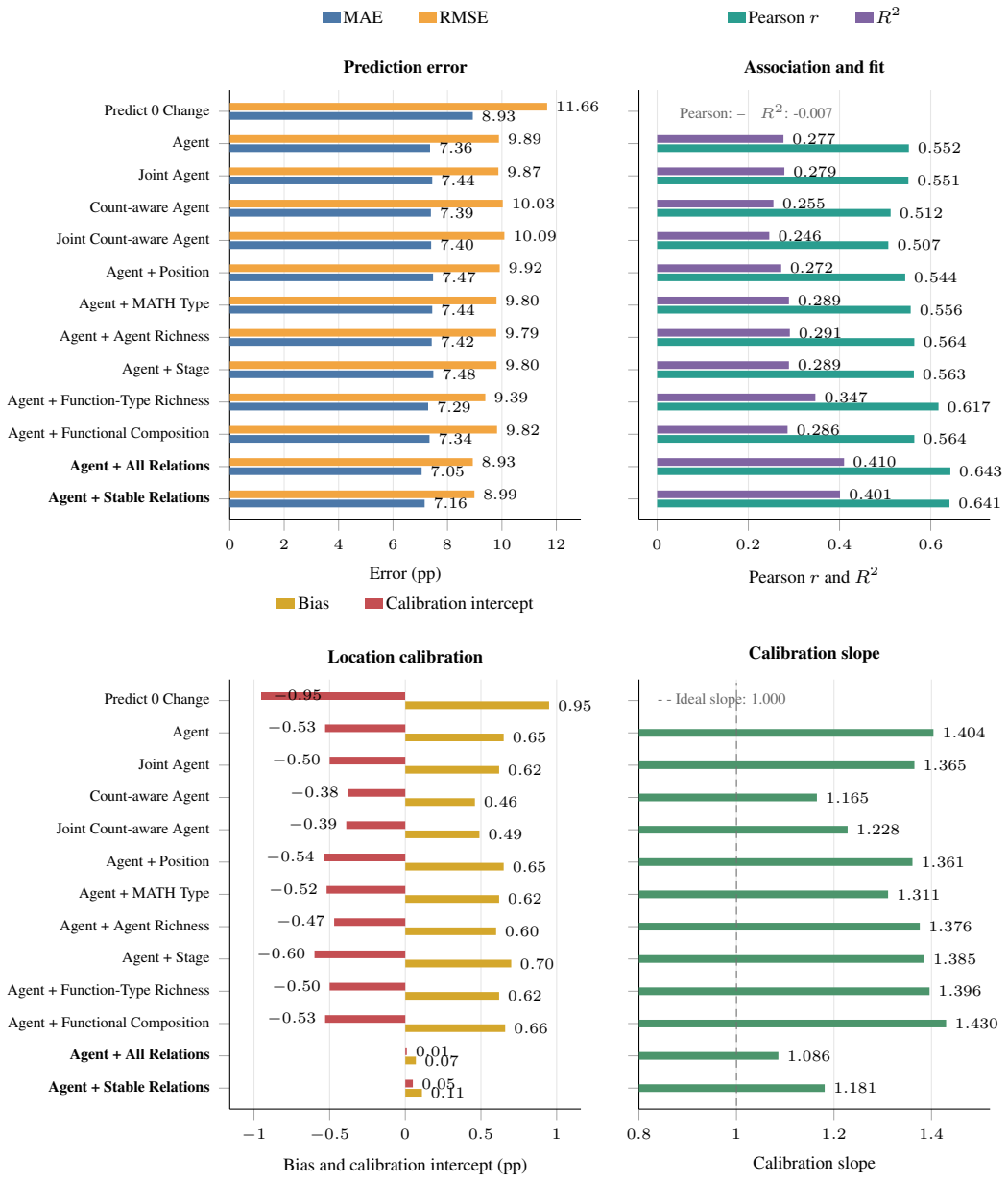


Figure 7: Complete replacement-prediction results on the above-noise subset across all model variants and reported metrics. The panels show MAE and RMSE, Pearson correlation and R^2, signed bias and calibration intercept, and calibration slope. Lower errors, higher Pearson correlation and R^2, and calibration quantities closer to their ideal values (zero for bias and intercept; one for slope) indicate better performance. Dashes mark metrics that are not defined for the zero-change baseline.

D.3 Complete Numerical Results on the Above-Noise Subset

Section 5 compares the incremental predictive value of different candidate information on the above-noise subset. The main-text Figure 2 summarizes the relative performance patterns across different metrics, while omitting exact numerical values to emphasize the overall comparison. Here, Figure 7 provide a detailed visualization with exact metric values for all models, including MAE, RMSE, Pearson correlation, R^2, bias, calibration intercept, and calibration slope.

These results use exactly the same model parameters and evaluation samples as the corresponding analysis in the main text. Overall, finer-grained first-order information produces only limited changes, while jointly modeling binary agent presence and ordered pairwise relations further improves overall Replacement Prediction performance. The definition of the above-noise subset is provided in Appendix D.2.

D.4 Complete Prediction Results on All 200 Seeds

Table 8 reports Replacement Prediction results on all 200 workflow seeds. All models use the same training data, test replacements, and evaluation procedure.

Table 8: Replacement Prediction results on all 200 workflow seeds. The Relation Model uses jointly estimated binary agent presence and ordered pairwise relations.

Model	MAE (pp)	RMSE (pp)	Pearson
Predict 0 Change	5.25	8.35	–
Agent Performance	5.24	7.54	0.435
Count-Aware Agent	5.31	7.67	0.413
Agent + Position	5.40	7.77	0.389
Agent + MATH Type	4.97	7.52	0.454
Agent + Agent Richness	5.30	7.50	0.446
Agent + Stage	5.34	7.54	0.439
Agent + Function-Type Richness	5.20	7.22	0.504
Agent + Functional Composition	5.15	7.47	0.450
Agent + Full Relation	5.49	7.20	0.531
Agent + Stable Relation	5.24	**7.07**	**0.540**

Across all 200 seeds, the large number of near-zero Replacement Effects compresses the differences in MAE across models. Predict 0 Change achieves an MAE of 5.25 pp, while Agent Performance achieves 5.24 pp. Some first-order candidate factors produce limited changes, but none yields consistent improvement across MAE, RMSE, and Pearson correlation.

Relational information provides a clearer gain in modeling relative effect variation and larger prediction errors. Agent Performance achieves an RMSE of 7.54 pp and a Pearson correlation of 0.435; the Full Relation Model reaches 7.20 pp and 0.531; using only the 9 Stable Relations further reaches 7.07 pp and 0.540.

Therefore, even when near-zero replacements are fully included in the evaluation, the predictive gain from ordered pairwise relations over agent performance remains.

D.5 Complete Evaluation and Above-Noise Analysis

The complete evaluation and the above-noise analysis use the same fitted models and the same underlying 200-seed replacement set. The above-noise analysis reports only the 100 seeds satisfying $|\Delta U_\tau| > 3.583$ pp, where 3.583 pp is the estimated execution-noise scale for the mean over three repeated runs.

On the complete 200-seed evaluation, near-zero Replacement Effects make absolute prediction errors less discriminative across models. Predicting no change already achieves an MAE of 5.25 pp. Agent Performance achieves an MAE of 5.24 pp, an RMSE of 7.54 pp, and a Pearson correlation of 0.435, while the Stable Relation Model achieves 5.24 pp, 7.07 pp, and 0.540, respectively.

On the above-noise subset, Agent Performance reaches an MAE of 7.36 pp, an RMSE of 9.89 pp, and a Pearson correlation of 0.552, while the Stable Relation Model reaches 7.16 pp, 8.99 pp, and 0.641. Thus, filtering out changes within the estimated execution-noise scale makes the differences in absolute prediction error more visible, while the correlation advantage of relational information remains present under both evaluation ranges.

The relational gain is therefore not induced by the above-noise selection. The complete 200-seed evaluation measures overall predictive performance over the random replacement distribution, while the above-noise subset provides a complementary view of prediction when the Replacement Effect exceeds the estimated execution-noise scale.

D.6 Exploratory Replacement Prediction Models

Beyond the prediction models studied in the main analysis, we additionally examine several exploratory variants to evaluate alternative representations of replacement information. These models use the same training data, replacement prediction set, score conversion, and evaluation procedure as the main analysis. Table 9 compares these variants with Agent Performance and the pairwise-relation models used in the main text.

Table 9: Replacement prediction performance of the main and exploratory models on the above-noise subset. The exploratory variants examine relation-only prediction and explicit higher-order relational representations.

Model	MAE (pp)	RMSE (pp)	Pearson	R^2	Bias (pp)	Calibration Intercept (pp)	Calibration Slope
Agent Performance	7.36	9.89	0.552	0.277	0.65	-0.53	1.404
Agent + Full-90 Ordered Pairs	7.05	8.93	0.643	0.410	0.07	0.01	1.086
Joint Binary Agent + Stable Relations	**7.16**	**8.99**	**0.641**	**0.401**	**0.11**	**0.05**	**1.181**
Full-90 Ordered Pairs Only	7.27	9.34	0.607	0.354	0.71	-0.66	1.207
Agent + All-720 Ordered Triples	8.52	10.71	0.545	0.151	-1.05	0.24	0.595
Agent + Full-90 Ordered Pairs + All-720 Ordered Triples	8.94	10.74	0.559	0.146	-1.19	0.30	0.586

Relation-Only Prediction. We first examine whether Agent Relations alone can predict replacement effects without explicitly modeling agent-level information. The *Full-90 Ordered Pairs Only* model uses all 90 ordered pairwise relations but excludes binary agent-presence features. It achieves an MAE of 7.27 pp, an RMSE of 9.34 pp, and a Pearson correlation of 0.607. This substantially exceeds the Agent Performance baseline in correlation, but remains below the combined Agent + Full-90 Ordered Pairs model, which reaches 0.643. This result suggests that relational information carries predictive value on its own, while agent-level and relational information provide complementary signals when modeled together.

Higher-Order Relation Prediction. We further explore whether explicitly introducing higher-order agent relations improves replacement prediction. For three distinct agents a, b, and c, we define an ordered three-agent relation as

$$t_{abc}(C) = \mathbb{I}[\exists\, i < j < k :\ r_i = a,\ r_j = b,\ r_k = c].$$

With 10 agents, this yields $10 \times 9 \times 8 = 720$ candidate ordered three-agent relations. We evaluate two variants: one that combines binary agent information with all 720 ordered triples, and another that additionally retains all 90 ordered pairwise relations.

Neither higher-order variant improves out-of-sample replacement prediction under the available data budget. Agent + All-720 Ordered Triples reaches a Pearson correlation of 0.545, while adding both all pairwise and all triple relations reaches 0.559, compared with 0.643 for the pairwise relation model. The higher-order models also exhibit larger prediction errors and poorer calibration.

Explicit higher-order modeling substantially enlarges the feature space and introduces many more interaction terms that must be estimated from historical workflow executions. The resulting performance should therefore not be interpreted as evidence that higher-order interactions are absent.

Rather, under the data regime considered in this work, these higher-order representations do not provide a practical predictive advantage over the substantially more compact low-order models. Since our goal is to identify reusable information that can support agent replacement from available historical executions, the main analysis focuses on Agent Performance and pairwise Agent Relations.

D.7 Cross-Setting Replacement Prediction on MMLU-Pro

To examine whether the predictive pattern observed in the main experiment extends to a different task type, we conduct an additional Replacement Prediction experiment on MMLU-Pro (Wang et al., 2024). Because the task requirements differ from mathematical reasoning, we select 7 agents from the 10-agent role pool used in the main experiment and set the workflow length to $L = 5$.

For statistical effect estimation, we use 200 tasks and generate 100 random workflow seeds for each task, yielding 20,000 task–workflow records for estimating Agent Performance and Agent Relations. Replacement Prediction is evaluated on another 100 tasks with 50 randomly generated workflows per task. For each workflow, one agent is randomly replaced, and both the original and replaced workflows are independently executed three times to estimate the observed Replacement Effect.

Table 10 reports representative prediction results. Agent Performance substantially improves over the zero-change baseline, reducing MAE from 5.252 pp to 3.794 pp and reaching a Pearson correlation of 0.609. Incorporating a small subset of ordered pairwise relations provides a further improvement: the best observed Top-k variant uses only 9 relations and reaches a Pearson correlation of 0.637, with an MAE of 3.575 pp and an RMSE of 4.353 pp.

Table 10: Replacement Prediction results on MMLU-Pro. The Top-9 relation model combines Agent Performance with the 9 highest-ranked ordered pairwise relations.

Model	MAE (pp)	RMSE (pp)	Pearson
Predict 0 Change	5.252	5.629	–
Agent Performance	3.794	4.505	0.609
Agent + Top-9 Relations	**3.575**	**4.353**	**0.637**

These results reproduce the main predictive pattern under a different task distribution, role pool, and workflow length. Agent Performance remains an effective replacement-prediction baseline, while incorporating only a small number of Agent Relations provides additional predictive value. This provides further evidence that the usefulness of relational information is concentrated in a sparse subset rather than requiring the full relation space.

We also conduct Replacement Decision Evaluation under the same MMLU-Pro setting. The corresponding intervention results are reported in Appendix E.1.

D.8 Scope of Cross-Setting Prediction Experiments

Large-scale Replacement Prediction is conducted primarily under the main experimental setting, with an additional cross-setting prediction experiment on MMLU-Pro reported above. Constructing such prediction sets requires repeated independent executions of both the original and replaced workflows for the same replacement, making them substantially more expensive than Replacement Decision Evaluation. We therefore do not reconstruct equally large prediction sets for every extended setting. Instead, for the remaining variations in agent-pool size, backbone model, dataset, and workflow length, we re-estimate Agent Performance and Agent Relations under the corresponding setting and evaluate their practical effectiveness through Replacement Decision Evaluation. The corresponding results are reported in Appendix E.1.

E Replacement Decision Evaluation and Cross-Setting Validation

This section provides additional details on the Replacement Decision Evaluation in the main text and further examines whether the effects of agent performance and agent relations on actual replacement decisions persist across different role-pool sizes, backbone models, task distributions, and workflow lengths.

Unlike Replacement Prediction, this evaluation directly executes the replacement selected by each model and uses the success rate of the resulting system as the evaluation metric. The Agent Performance Model ranks candidate replacements using Binary Agent Performance estimated separately for each agent, while the Agent Relation Model uses jointly estimated Binary Agent Presence and Ordered Pairwise Relations. For the same original workflow, both models are given exactly the same candidate replacement space and independently select

$$\tau^{\star}_{\text{agent}} = \arg\max_{\tau \in \mathcal{T}(C)} \Delta S_{\text{agent}}(\tau), \qquad \tau^{\star}_{\text{relation}} = \arg\max_{\tau \in \mathcal{T}(C)} \Delta S_{\text{relation}}(\tau).$$

All replacement decisions are made before executing the selected candidate replacement.

E.1 Cross-Setting Experiments and Results

The main experiment uses 10 agents, Qwen2.5-14B-Instruct, MATH-486, and sequential workflows of length $L = 8$. The complete data split and execution scale are described in Appendix A.3. Based on this setting, we construct the following extensions.

Different Workflow Lengths and Role-Pool Sizes. $L = 4$ and $L = 12$ modify only the workflow length of the main setting to $L = 4$ and $L = 12$, respectively. 5 Agents reduces only the role pool from 10 agents to 5. The backbone model, task set, and workflow execution mechanism remain unchanged from the main experiment.

Different Backbone Model and Mathematical Task Set. We further replace the backbone model with `DeepSeek-V4-Flash` (DeepSeek-AI, 2026) and the task set with OMNI-MATH (Gao et al., 2025). The agent role definitions, role-pool size, workflow length, and single-point replacement protocol remain unchanged from the main experiment.

Different Task Type Setting. To further extend the evaluation to a different task type, we construct a new workflow setting on MMLU-Pro (Wang et al., 2024). Because the task objective is no longer restricted to mathematical reasoning, the role pool is adjusted to 7 agents with corresponding modifications to the role definitions, and the workflow length is set to $L = 5$. Results for this setting are reported in Table 11.

For all extended settings, we regenerate historical random workflows under the corresponding environment and re-estimate agent-level and relation-level information. The final Replacement Decision Evaluation does not reuse agent or relation coefficients from the main experiment.

Table 11 summarizes the Replacement Decision Evaluation results across all settings.

Table 11: Replacement Decision Evaluation under different experimental settings. Agent Gain and Relation Gain are computed relative to No Replacement; Relation vs. Agent denotes the additional change of the Agent Relation Model over the Agent Performance Model.

Setting	No Replacement	Agent Performance	Agent Relation	Agent Gain	Relation Gain	Relation vs. Agent
Main: 10 Agents, $L = 8$	46.86%	51.59%	53.17%	+4.73 pp	+6.31 pp	+1.58 pp
5 Agents	42.48%	46.40%	46.96%	+3.92 pp	+4.48 pp	+0.56 pp
$L = 4$	35.92%	50.40%	51.44%	+14.48 pp	+15.52 pp	+1.04 pp
$L = 12$	48.15%	52.16%	55.76%	+4.01 pp	+7.61 pp	+3.60 pp
DeepSeek-V4-Flash + OMNI-MATH	52.50%	56.00%	57.00%	+3.50 pp	+4.50 pp	+1.00 pp
MMLU-Pro: 7 Agents, $L = 5$	63.72%	65.48%	65.64%	+1.76 pp	+1.92 pp	+0.16 pp

Absolute success rates and replacement gains should not be compared directly across settings, because changes in the role pool, workflow length, backbone model, and task distribution also change the underlying workflow distribution and candidate replacement space. The focus here is whether the same relative pattern reappears under different settings.

Across all evaluated settings, the Agent Performance Model improves the average success rate over No Replacement, and the Agent Relation Model further improves the point estimate on top of Agent Performance. The additional gains of the Relation Model over the Agent Performance Model are $+1.58$, $+0.56$, $+1.04$, $+3.60$, $+1.00$, and $+0.16$ pp, respectively.

These results indicate that agent performance estimated from historical workflows provides a reusable replacement signal, while the additional information from ordered pairwise agent relations is not limited to a single role pool, backbone model, task set, or workflow length in the main experiment.

E.2 Additional Analysis of the Main Experiment

In the main experiment, the Agent Performance Model increases the average success rate from 46.86% to 51.59%, while the Agent Relation Model further improves it to 53.17%. The paired difference between the Relation Model and the Agent Performance Model is $+1.58$ pp, with a task-cluster 95% CI of

$$[+0.68, +2.50] \text{ pp}.$$

Beyond the average success rate, we further examine how these gains are distributed across workflow seeds.

Table 12: Distribution of replacement outcomes across workflow seeds in the main experiment.

Strategy	Improved	Declined	Tied	Positive (%)	I/D Ratio
Agent Performance Model	69	26	5	72.63	2.65:1
Agent Relation Model	78	20	2	79.59	3.90:1

The Agent Relation Model improves more workflow seeds while reducing the number of workflows with performance degradation, indicating that its average gain is not driven primarily by a small number of extreme improvements.

Higher-Performance Workflows. We next examine whether the additional gain from relational information is limited to workflows with relatively low baseline performance. We divide the unreplaced workflows into two equal-sized groups according to their success rates under the unreplaced baseline and focus on the higher-performance group.

Table 13: Replacement performance on the higher-success half of workflows under the unreplaced baseline.

Strategy	Success Rate	Gain over Baseline	Task-Cluster 95% CI
Unreplaced Baseline	52.93%	–	–
Agent Performance Model	52.57%	-0.36 pp	[-1.53,+0.80]
Agent Relation Model	54.25%	+1.32 pp	[+0.09,+2.61]

For these workflows, the Agent Performance Model provides no improvement over the unreplaced baseline, whereas the Agent Relation Model achieves a positive gain of $+1.32$ pp, with a task-cluster 95% CI of $[+0.09, +2.61]$ pp. This indicates that the additional value of relational information is not limited to identifying low-performing workflows that can be improved by replacing weaker agents with globally stronger ones. Relational information can also provide useful workflow-specific adjustments when the original workflow already performs relatively well.

Decision Disagreement. We further divide the 100 workflow seeds into decision-agreement and decision-disagreement groups according to whether the Agent Performance Model and the Agent Relation Model select the same replacement.

Table 14: Actual performance of the Agent Performance Model and Agent Relation Model on workflows with agreement and disagreement in replacement decisions.

Subset	N	Agent Performance	Agent Relation	Difference	Task-Cluster 95% CI
Same decision	27	51.70%	52.30%	+0.59 pp	$[-0.81, +2.00]$
Different decision	73	51.54%	53.49%	+1.95 pp	$[+0.94, +2.98]$

Across the 27 workflow seeds where both models select the same replacement, the average difference between the two strategies is $+0.59$ pp, with a confidence interval that crosses zero. Across the 73 seeds where the models select different replacements, the Agent Relation Model reaches 53.49%, compared with 51.54% for the Agent Performance Model, corresponding to a difference of $+1.95$ pp with a task-cluster 95% CI of $[+0.94, +2.98]$ pp.

Therefore, the additional benefit of the Relation Model is concentrated mainly in workflows where relational information changes the ranking of candidate replacements. On average, these workflow-specific adjustments correspond to higher realized execution performance.