TOWARD GREEN MULTI-AGENTS: UNDERSTANDING AND TACKLING DYNAMIC RESIDUAL DECAY VIA CROSS-LIFECYCLE GRAPH

Anonymous authors
Paper under double-blind review

ABSTRACT

As multi-agent systems adapt their collaboration across tasks, they retain information from earlier rounds that can become outdated while continuing to influence decisions. Reusing such history can propagate invalid evidence; recomputing all affected work increases inference cost. Existing work on topology, persistent memory, and dependency updates does not jointly determine which persistent historical influence is associated with declining current-task utility and which computations the current task needs. We present the first systematic study of Dynamic Residual Decay (DRD), a cross-lifecycle phenomenon where residual influence persists across multi-turn tasks while its positive utility for the current task decays. To tackle DRD while preserving reusable work, we propose the cross-lifecycle graph (CLG), which traces invalidation forward and current demand backward, stops historical tracing at valid replacements, and re-executes the selected dependencies in order. Across four models and three multi-agent task sets, historical influence persists after positive utility declines. In cross-task recovery evaluations, CLG achieves a macro-averaged normalized utility of 0.9781 across task–model cells, compared with 0.9485 under Global Rerun, while reducing model calls by 39.8%, tokens by 29.9%, and elapsed time by 33.9%. Selectively retiring invalid influence and retaining useful history offers a route toward green multi-agent execution that is computationally cleaner and more resource-efficient.

1 INTRODUCTION

Multi-agent systems (MAS) increasingly move beyond fixed roles toward task-dependent agents, communication links, and routing (Wang et al., 2026; Li et al., 2026). This dynamism extends beyond a single task: persistent memory allows messages, observations, and decisions from earlier rounds to re-enter later collaboration (Park et al., 2023; Shinn et al., 2023; Hu et al., 2026; Fang et al., 2026). Such continuity makes green MAS execution a joint problem of computational cleanliness and resource efficiency. Historical computation can reduce redundant work, but outdated information can also compromise the current task. Efficiency therefore depends not only on which agents collaborate now, but also on which earlier computations can be safely retained (Schwartz et al., 2020).

An agent can retire while its artifacts remain available to successors; their later reading, adoption, or derivation constitutes *cross-lifecycle interaction*. Under changing tasks and evidence, a historical artifact may continue to shape decisions after its positive task utility has declined. We call this mismatch *dynamic residual decay* (DRD). Retaining invalid history can redirect subsequent decisions, while recomputing all of its descendants spends model calls, tokens, and time on outputs that may not be needed. Efficient reuse therefore requires separating continued influence from present benefit, then updating only the invalid dependencies demanded by the current task.

Existing work addresses complementary parts of this problem. Topology studies characterize how information propagates (Li et al., 2026; Zhou et al., 2025; Liu et al., 2026), while memory and provenance studies track what persists and where it originated (Yang et al., 2026; Hu et al., 2026; Fang et al., 2026). Incremental computation, in turn, determines which computations are affected by change (Acar et al., 2002; Mokhov et al., 2018). We study their intersection: residual information

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.

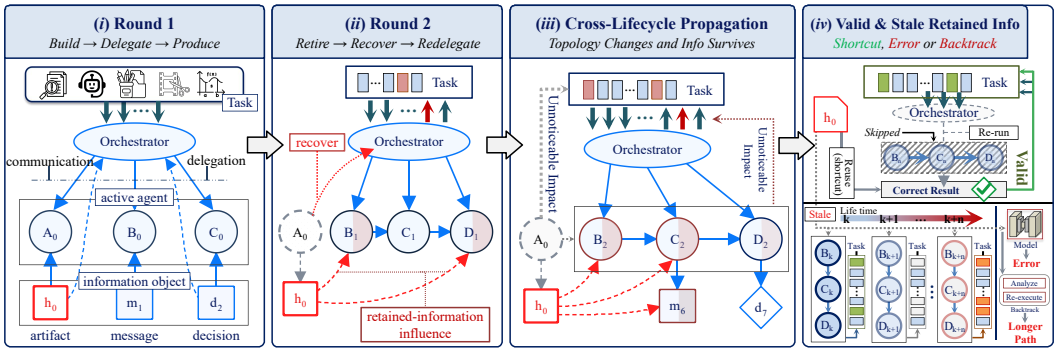


Figure 1: Information from a retired agent can remain influential after its task utility declines. Valid history saves work; outdated history may change a later decision or require correction.

can propagate through multi-agent interactions after its producer retires, even as its utility for the current task changes. Cross-lifecycle recovery then asks which affected outputs the next task actually needs. Figure 1 illustrates this tension between beneficial reuse and the downstream cost of outdated history.

This setting yields two coupled challenges. First, influence and present benefit must be distinguished: evidence access, the timing of counterevidence, and shared origins determine whether outdated information continues to propagate across rounds. Second, once history is invalidated, recovery must avoid both extremes. Recomputing every descendant wastes effort on unrequested outputs, while following current demand alone can reuse invalid ancestors.

Recovery therefore needs to connect invalidation paths with currently demanded outputs: unaffected work should be retained, their shared dependency paths should be refreshed locally, and invalid branches not required by the current task should remain unavailable for reuse until they are repaired. This avoids both redundant computation and the propagation of outdated evidence.

We address these two challenges in sequence. To understand DRD, we measure historical influence, positive utility, and their mismatch under changing tasks, then examine how evidence access, counterevidence timing, and shared provenance shape this mismatch. To tackle DRD, we introduce the cross-lifecycle graph (CLG), which traces influence forward from an invalid source and current demand backward from required outputs, stopping at valid replacements. Their intersection determines the nodes for dependency-ordered recovery.

This formulation connects the green objective to a concrete recovery decision: preserve task utility and exclude invalid historical inputs while reducing model calls, token usage, and elapsed time. We evaluate these outcomes separately, together with the effect of incomplete dependency records on recovery coverage.

The main contributions of this work are as follows.

1. **We first identify and characterize a cross-lifecycle residual-influence phenomenon in dynamic multi-agent systems, which we call Dynamic Residual Decay (DRD).** DRD captures cases in which residual information continues to shape multi-turn decisions after its utility for the current task has declined; we formalize this influence–utility mismatch by separating agent and artifact lifecycles from downstream dependencies, and analyze how evidence access, counterevidence timing, and shared provenance affect correction, verification, and resource-efficient execution.
2. **We propose the Cross-Lifecycle Graph (CLG) to tackle DRD through demand-aware recovery.** CLG traces invalidated influence forward and current-task demand backward, uses valid replacements as boundaries for historical dependence, and recomputes only their intersection in dependency order, thereby retaining valid work and preventing stale-state reuse; controlled experiments evaluate its recovery coverage under varying demand scopes and dependency-record quality.
3. **Across phenomenon studies covering four foundation models and three multi-agent task suites, we observe that historical influence persists even after its positive util-**

ity has decayed. In cross-task recovery evaluations, the task-normalized utility index is 0.9781 for CLG versus 0.9485 for Global Rerun. Under the same matched conditions, CLG reduces model calls by 39.8%, token usage by 29.9%, and elapsed time by 33.9%.

2 Related Work

Changing agent networks. Earlier multi-agent systems organize collaboration through conversational roles (Wu et al., 2023); later work selects or adapts agents, communication links, and routing to fit the task (Wang et al., 2026; Li et al., 2026). These methods change *who* collaborates and how current messages travel. Topology experiments also show that network structure affects the spread of useful and incorrect information (Li et al., 2026; Zhou et al., 2025; Liu et al., 2026). Across multiple tasks, however, retiring an agent does not erase the artifacts it left behind. Their influence can persist even as the active network and task change.

Persistent information and evidential origin. Memory systems retain and retrieve observations or reflections across tasks (Park et al., 2023; Shinn et al., 2023; Hu et al., 2026; Fang et al., 2026; Yang et al., 2026). Work on contamination shows how older or externally supplied content can redirect later decisions (Greshake et al., 2023; Zou et al., 2025; Cui et al., 2026), while provenance and debate studies distinguish independent evidence from judgments inherited from a shared source (Yang et al., 2026; Du et al., 2024; Choi et al., 2025). Retention enables reuse and avoids repeating useful work, but availability does not establish present utility. DRD measures when influence survives a decline in utility and relates that mismatch to correction and verification.

Quality-aware efficiency. Green AI motivates examining computational cost alongside task performance (Schwartz et al., 2020). Model routing adapts which agents contribute to a request (Li et al., 2026; Wang et al., 2026), and prompt compression reduces the length of supplied context (Chen et al., 2026b). In a multi-round system, another source of cost is repeatedly computing descendants of historical information that no longer serves the current demand. Saving that work is useful only when invalid history cannot silently re-enter the answer. We therefore consider quality, invalid-history exposure, calls, tokens, and elapsed time together rather than treating shorter execution as the sole objective.

Selective updates after invalidation. Incremental computation and build systems update results affected by changed inputs rather than recomputing a complete program (Acar et al., 2002; Mokhov et al., 2018). Database provenance and view maintenance similarly connect source records to derived outputs (Cheney et al., 2009; Gupta & Mumick, 1995). By contrast, CLG combines dependency tracking with current task demand, since tracing every descendant can refresh unrequested work while tracing demand alone can miss invalid ancestors. In addition, Appendix A positions CLG relative to related approaches.

3 Problem Definition and CLG Method

This section gives the complete CLG recovery rule. We treat historical artifacts as versioned states in the Cross-Lifecycle Graph (CLG), whose edges record dependencies, and compute the exact intersection between states reachable from newly invalidated evidence and states that support outputs demanded by the current task. The definitions, matrix recurrences, execution rule, and guarantee are stated here; implementation details and the proof are in Appendix B.

3.1 What persists after an agent retires

An agent can finish its task while its messages, observations, plans, and decisions remain available to later agents. Reading, adopting, or deriving such an artifact constitutes cross-lifecycle interaction. DRD concerns the loss of positive task utility as circumstances change, even when the artifact still shapes subsequent execution. Continued influence alone does not establish benefit, and declining utility alone does not show that an artifact is still used.

For each matched later task, define the influence indicator $A \in \{0, 1\}$ to be 1 when an artifact or its derivative influences a decision and 0 otherwise. Let ΔU be the difference in official task

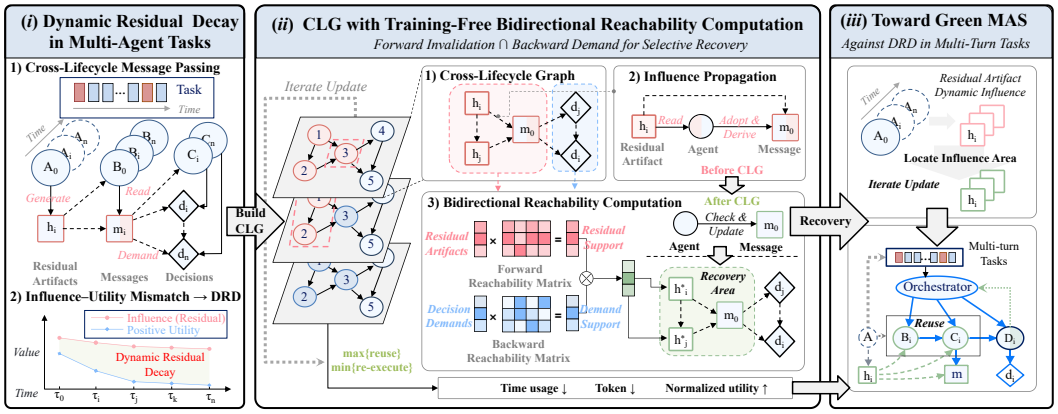


Figure 2: CLG traces invalidation forward and current demand backward through cross-lifecycle dependencies. Their shared support selects computations for local re-execution before final synthesis.

utility with versus without that history. We set the utility threshold separately for each benchmark according to its native scorer resolution; the precise definition and benchmark-specific values are given in Appendix C.1, Eq. 21. We measure influence survival $I = \Pr(A = 1)$, positive-utility survival $U = \Pr(\Delta U > \varepsilon)$, and mismatch occupancy $M = \Pr(A = 1 \wedge \Delta U \leq \varepsilon)$ directly from joint observations. RQ1 measures these quantities under task drift; RQ2 asks when influential history changes decisions or incurs correction.

3.2 Cross-Lifecycle Dependencies and Current Demand

Definition 1 (Cross-Lifecycle Graph) *At recovery time t, $\mathcal{G}_t = (V_t, E_t, \lambda_t, \sigma_t)$ contains artifact and decision states V_t, consumption edges E_t, lifecycle labels λ_t, and validity states σ_t. An edge $(u, v) \in E_t$ records that the version of v actually consumed u; edges are appended in event order, so the recorded graph is acyclic. Retirement changes λ_t but does not erase a recorded state or edge.*

CLG addresses the recovery problem created when current evidence invalidates a historical source $r \in V_t$ and the current task demands a nonempty set $D_t \subseteq V_t$. Let K_t be the set of nodes with valid replacements and let $\chi_t(v) = \mathbf{1}_{\{v \in K_t\}}$. With $n = |V_t|$, let $A_t \in \{0,1\}^{n \times n}$ be indexed by V_t with $(A_t)_{uv} = 1$ iff $(u, v) \in E_t$; define the version-aware adjacency, source indicator, and demand indicator by

$$(A_t^{\text{valid}})_{uv} = (A_t)_{uv}[1 - \chi_t(v)], \qquad \mathbf{e}_r(u) = \mathbf{1}_{\{u=r\}}, \qquad \mathbf{d}_t(u) = \mathbf{1}_{\{u \in D_t\}}. \tag{1}$$

The mask removes historical incoming edges of a node that already has a valid replacement; the replacement itself remains available as an input. Figure 2 relates the recorded graph, the two propagation directions, and local execution.

3.3 Bidirectional Reachability and Selection

The two traversals are written as Boolean matrix recurrences. With $\vee$ and $\wedge$ denoting elementwise Boolean sum and product, let $\mathbf{f}_t^{(k)}$ and $\mathbf{b}_t^{(k)}$ record nodes reached within k steps:

$$\begin{aligned} &\mathbf{f}_t^{(0)} = \mathbf{e}_r, \quad \mathbf{f}_t^{(k+1)} = \mathbf{f}_t^{(k)} \vee (\mathbf{f}_t^{(k)} A_t), && \mathbf{h}_t = \bigvee_{k=0}^{n-1} \mathbf{f}_t^{(k)}, \\ &\mathbf{b}_t^{(0)} = \mathbf{d}_t, \quad \mathbf{b}_t^{(k+1)} = \mathbf{b}_t^{(k)} \vee (\mathbf{b}_t^{(k)} (A_t^{\text{valid}})^\top), && \mathbf{b}_t^{*} = \bigvee_{k=0}^{n-1} \mathbf{b}_t^{(k)}. \end{aligned} \tag{2}$$

The first line is forward invalidation reachability; the second is reverse demand support. Paths longer than $n - 1$ repeat a node in the finite DAG, so the finite Boolean closure is exact. Let $H_t = \{v \in V_t : (\mathbf{h}_t)_v = 1\}$ and $B_t = \{v \in V_t : (\mathbf{b}_t^*)_v = 1\}$. The immediate recovery range is

$$\begin{aligned} Z_t &= \{v \in V_t : (\mathbf{h}_t)_v = 1 \wedge (\mathbf{b}_t^*)_v = 1\} = H_t \cap B_t, \\ &= \{v : \exists r \rightsquigarrow v \rightsquigarrow d,\ d \in D_t,\ \text{with replacement boundaries respected}\}. \end{aligned} \tag{3}$$

Forward reachability alone would refresh unrequested descendants; backward reachability alone could retain an invalid ancestor. Their intersection is the demand-selected recovery range, not a utility estimate. Appendix H gives the path proof, and Appendix B gives the matrix implementation.

Definition 2 (Demand-selected recovery range) *Z_t is the set of recorded states on at least one path from an invalidated source to a currently demanded output in the version-aware graph. The range is conservative when a recorded edge is not semantically necessary, and incomplete when a necessary edge is missing.*

3.4 Local recovery and demand scope

Before recovery, every invalid state in H_t is made unreadable. Let $U_t = Z_t \setminus K_t$ be the states requiring a new version, and let $(v_1, \dots, v_m)$ be any topological order of the induced dependency graph:

$$U_t = Z_t \setminus K_t, \qquad (v_1, \dots, v_m) = \mathrm{TopoSort}(\mathcal{G}_t[U_t]). \tag{4}$$

For each v_i, inputs are resolved in the order "new versions produced in this run, valid replacements, then unaffected valid history." If $P_t(v_i)$ denotes its recorded parents and $\eta_i(u)$ the resolved version of parent u, local execution is

$$v_i^+ = F_i(\{\eta_i(u) : u \in P_t(v_i)\}, x_t, T_i), \qquad a_{t+1}(v_i) = v_i^+, \tag{5}$$

where x_t is current evidence and T_i is the tool context. The new node and its actual input edges are appended before processing v_{i+1}, so later nodes cannot read a stale version. Final synthesis consumes the latest usable version for each demanded output:

$$y_t = S(\{a_{t+1}(d) : d \in D_t\}). \tag{6}$$

States in $H_t \setminus Z_t$ stay unreadable and are deferred; deferral never promotes an invalid state to valid status. If $H_t \neq \varnothing$, the structural scope reduction is $\rho_t = 1 - |Z_t|/|H_t|$. This graph-size quantity is separate from calls, tokens, latency, and answer quality. Exact selection and topological ordering take $O(|V_t| + |E_t|)$ graph time; Appendix B specifies input resolution and resource accounting.

3.5 Validity, provenance, and recovery invariants

Recording actual consumption edges and versions distinguishes historical use from mere artifact availability. CLG revokes the full forward influence set, executes only U_t, and keeps valid replacements available. The following theorem is the main correctness statement used by the experiments.

Theorem 1 (Correctness of the CLG recovery range) *Assume that the recorded CLG is finite and acyclic, all task-relevant parents of nodes in Z_t are recorded, valid replacements are readable, the runtime rejects unavailable states, and each input resolver returns $\eta_i(u) \neq \perp$. Then*

$$\begin{aligned} Z_t = H_t \cap B_t = \{&v : \exists r \rightsquigarrow v \rightsquigarrow d,\ d \in D_t, \\ &\textit{with replacement boundaries respected}\}, \\ U_t = Z_t \setminus K_t.& \end{aligned} \tag{7}$$

Let $D_t^+ = \{a_{t+1}(d) : d \in D_t\}$ be the demanded versions after recovery. The topologically executed output then satisfies

$$\left\{v \in V_{t+1} : \exists d^+ \in D_t^+,\ v \rightsquigarrow_{\mathcal{G}_{t+1}} d^+\right\} \cap (H_t \setminus Z_t) = \varnothing. \tag{8}$$

If, in addition, local execution and Global Rerun use the same evidence and tool snapshot, each $u \in U_t$ is sensitive along at least one path to a demanded output, and no unrecorded bypass or replacement exists, then any reliable recorded-state update set S_t (the historical states regenerated in the run) satisfies

$$U_t \subseteq S_t. \tag{9}$$

The range identity follows from the matrix closure, the safety equation follows from the runtime guard, and the inclusion equation is conditional on path sensitivity. None of these equations infers semantic necessity from reachability alone. The proof and the failure modes for missing or extra edges are given in Appendix B and Appendix H. Thus CLG provides a conservative, auditable recovery range even when graph quality is imperfect.

3.6 QUALITY-AWARE GREEN RECOVERY

The green objective is selective reuse under validity constraints. Let $c_v \geq 0$ be the measured cost of updating node v (calls, tokens, or a separately reported time quantity). The selected structural cost and its corresponding saving are

$$\mathrm{NRC}_c(W) = \sum_{v \in W \setminus K_t} c_v, \qquad \mathrm{Saving}_c = 1 - \frac{\mathrm{NRC}_c(Z_t)}{\mathrm{NRC}_c(H_t)}. \tag{10}$$

This is a bookkeeping identity over selected nodes and does not assume equal node costs. Official task utility and invalid-history exposure characterize correctness, while calls, tokens, and elapsed time characterize resources. We report them separately because a smaller Z_t need not imply proportional end-to-end savings.

4 EXPERIMENTS

We investigate four questions. RQ1 characterizes DRD under task drift; RQ2 examines its consequences for correction and verification; RQ3 evaluates CLG in task utility and resource use; and RQ4 tests how demand and dependency quality affect recovery coverage.

Task suites and cross-lifecycle setting. Our evaluation spans three benchmarks with distinct interaction structures: TravelPlanner (Xie et al., 2024) for tool-assisted constrained planning, ScienceWorld (Wang et al., 2022) for sequential interaction in a changing environment, and MultiAgentBench/MARBLE (Zhu et al., 2025) for explicit multi-agent collaboration. We preserve their native objectives, tools, and official scorers, but adapt the temporal organization needed to expose cross-lifecycle behavior. Under the standard benchmark setup, tasks are evaluated independently, with no history carried across task boundaries. We therefore chain semantically related *complete tasks* into successive rounds, with each round instantiating new agents under updated task or environment conditions and keeping earlier messages, artifacts, and decisions addressable. This organization allows agent lifecycles to end before information lifecycles, so historical information can continue to affect successor executions. Sequence construction and task drift are detailed in Appendix C.1.

Model backbones and evaluation protocol. We use DeepSeek-v4.1-Flash, GLM-5.3-Flash, GPT-Terra-Medium, and Qwen-3.8-Flash. Our goal is to study information propagation in dynamic multi-agent systems rather than the capability frontier of individual base models; we therefore use inference-efficient variants for large-scale, repeated multi-round and multi-agent executions under matched conditions. Spanning four model families further tests whether the observed cross-lifecycle phenomena are model-specific. All compared strategies share task conditions, model settings, scoring rules, and post-invalidation resource-accounting windows. Exact configurations, sample sizes, and statistical procedures are provided in Appendix C.

Experimental setup. We implement orchestration with LangGraph StateGraph, with A2A as the inter-agent interface. Workers run in isolated Docker containers and call assigned models through an OpenAI-compatible gateway. We match runs by task sequence and random seed, apply the same scoring and resource accounting, and replay the recorded topology where required.

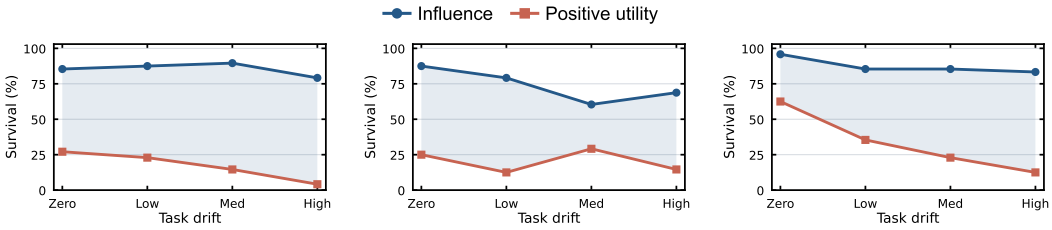
4.1 RQ1: DOES TASK DRIFT EXPOSE DRD IN HISTORICAL INFLUENCE?

To examine whether retired history continues to shape later decisions after it loses value, we compare matched executions with and without that history across four levels of task drift. Table 1 summarizes influence and positive utility by model, while Figure 3 shows their trajectories by task suite (Appendix D).

At high drift, influence remains at 55.6–86.1% across the four models, although positive-utility survival falls to 5.6–13.9%. TravelPlanner's positive-utility survival declines as drift increases; ScienceWorld shows the same influence–utility mismatch even at zero drift. Continued influence therefore does not establish that historical information still benefits the current task.

Table 1: Model-level influence (I) and positive-utility (U) survival at four drift levels (%). Each model row is macro-averaged across the three task suites.

Model	I_0	U_0	I_{low}	U_{low}	I_{medium}	U_{medium}	I_{high}	U_{high}
DeepSeek	94.4	33.3	94.4	19.4	83.3	27.8	86.1	13.9
GLM	100.0	33.3	88.9	22.2	94.4	16.7	80.6	11.1
GPT	100.0	55.6	97.2	36.1	77.8	27.8	86.1	11.1
Qwen	63.9	30.6	55.6	16.7	58.3	16.7	55.6	5.6

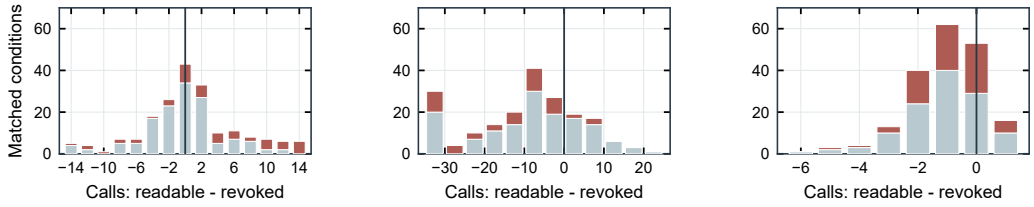


(a) MARBLE (b) ScienceWorld (c) TravelPlanner

Figure 3: Influence and positive utility across four drift levels, averaged equally across the four models within each task suite. Results are aggregated by suite here and by model in Table 1.

4.2 RQ2: How does DRD shape correction and verification?

We next examine whether influential history changes outcomes and creates work for correction and verification. Figure 4 compares calls with history readable or revoked; the outcome trajectories and evidence interventions appear in Appendix E.



(a) ScienceWorld (b) MARBLE (c) TravelPlanner

Figure 4: Matched readable-minus-revoked call differences. Red marks lower utility with readable history; gray marks equal or higher utility.

In TravelPlanner, lower-utility loss rises from 18.8% at zero drift to 61.5% at high drift, and ScienceWorld reaches 42.4% loss at high drift. In the tested current-evidence conditions, additional correction steps reduce history-attributable loss to zero, whereas inherited evidence alone leaves that loss largely unchanged. Late counterevidence increases correction work, and shared evidential roots raise consensus without a corresponding increase in independent rechecking. These results connect residual influence to task outcomes and show why evidence access and provenance matter for recovery.

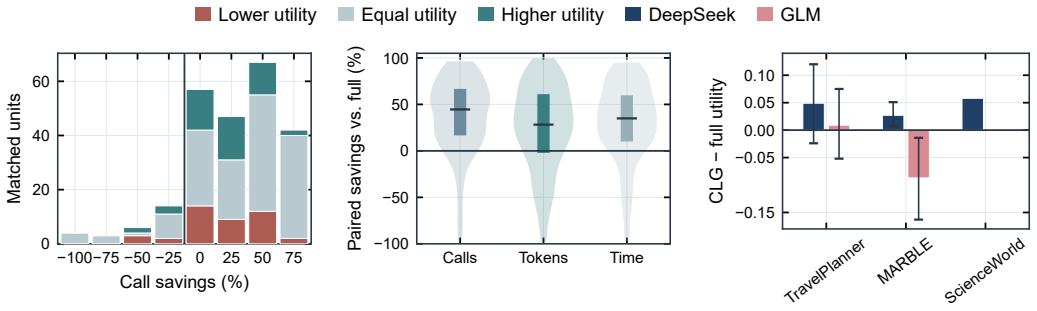
4.3 RQ3: How does CLG balance task utility and recomputation?

We evaluate task utility and resource use under demand-selected recovery. Each task uses its official scorer. Because the three scorers have different native scales and score resolutions, we compare strategies within each benchmark/model combination by dividing a strategy's mean utility by the largest mean among recovery-complete strategies, then average the six resulting values with equal weight. This gives a task-normalized utility index for summarizing results across the three task suites; Appendix F gives the definition and the corresponding reference utilities. We com-

pare CLG with HCache-Safe, an HCache-inspired dependency-aware cache-invalidation strategy with TTL (Gao et al., 2025), and PlanFence-SM, a plan-scoped dependency validator with bounded replanning (Chen et al., 2026a). Global Rerun and CLG (forward-only) provide two reference conditions. Table 2 summarizes the strategies, and Figure 5 shows matched outcomes and task-suite variation.

Table 2: Task-normalized utility and resource savings for recovery strategies. Results are equally weighted across the evaluated cells; resource savings are relative to Global Rerun.

Recovery strategy	Task-normalized utility	Calls saved (%)	Tokens saved (%)	Time saved (%)
Global Rerun	0.9485	0.00	0.00	0.00
HCache-Safe	0.9143	20.04	5.49	20.62
PlanFence-SM	0.8946	32.75	19.77	29.31
CLG (forward-only)	0.8510	28.35	15.16	24.84
CLG	**0.9781**	**39.79**	**29.95**	**33.86**



(a) Utility and calls (b) Savings distributions (c) Utility by task suite

Figure 5: Recovery evidence at matched-unit and task-suite resolution. The first panel links utility outcome to call savings; the second shows matched savings distributions; the third compares utility by task suite.

Across six task–model settings, CLG has higher mean official utility than Global Rerun in four, equal utility in one, and lower utility for MARBLE–GLM. Its task-normalized utility is 0.9781 versus 0.9485, while using 39.79% fewer calls, 29.95% fewer tokens, and 33.86% less time.

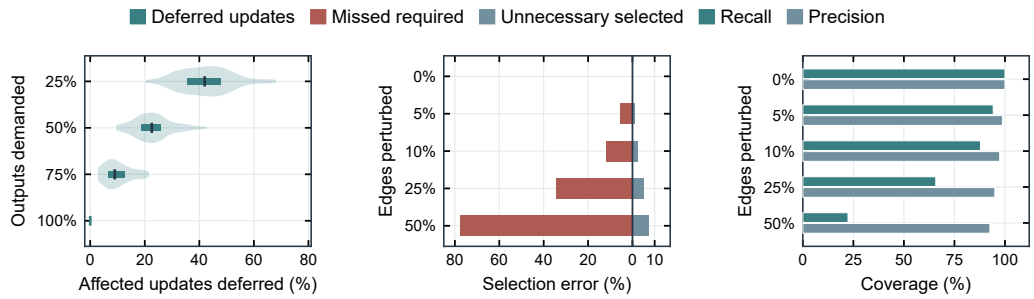
4.4 RQ4: How do demand and dependencies shape CLG recovery coverage?

Finally, we examine how the requested outputs and the recorded dependencies determine the recovery set. Figure 6 shows the effect of changing output demand and perturbing graph edges; Appendix G reports the corresponding values.

When one quarter of outputs are requested, 42.20% of affected updates can be deferred on average; when every output is requested, that fraction falls to zero. Deleting 10% and 50% of dependencies reduces recovery recall to 87.93% and 22.36%, respectively, while adding 50% extra edges preserves recall and lowers precision to 92.53%. Thus, current demand determines how much recovery work can be deferred, and the dependency record determines how accurately that work is selected.

5 Discussion and Limitations

Empirical findings. DRD identifies two lifetimes for historical information. Influence survival records whether a successor still uses an artifact or its derivatives, whereas positive-utility survival records whether that use improves the current task. RQ1 shows that these lifetimes separate under drift, and RQ2 shows why the separation matters: current evidence can correct an inherited judgment, but counterevidence arriving after downstream commitment leaves more state to repair and can increase verification work. This distinction makes residual influence measurable before it becomes a recovery obligation, linking the phenomenon study to the graph-based recovery rule.



(a) Demand-dependent scope (b) Selection error (c) Dependency quality

Figure 6: Exact recovery under changing output demand and recorded-edge quality. The panels show deferred updates, selection error, and coverage under missing or extra edges.

Recovery mechanism. Forward reachability from an invalid source marks the states that can carry its effect; backward reachability from the current demand marks the states still needed now. Their intersection identifies the affected recovery subgraph; states in this subgraph that lack valid replacements are regenerated. RQ4 shows that 42.20% of affected updates can be deferred when one quarter of the outputs is requested. This separates the graph-defined recovery scope from task-level utility and resource outcomes, allowing both to be evaluated within the same execution record. Replacement boundaries preserve repaired history while preventing superseded ancestors from extending the selected subgraph.

Global Rerun comparison. Global Rerun withdraws recorded artifacts from reuse and regenerates the recorded graph from new upstream results; CLG carries forward valid states and regenerates demanded affected nodes. Across six matched task–model settings, CLG has higher official utility in four, equal utility in one, and lower utility in MARBLE–GLM. Overall, CLG attains higher or comparable official utility than Global Rerun in most evaluated settings; however, the higher utility of Global Rerun in MARBLE–GLM shows that single-round multi-agent execution can still retain an advantage. Thus, demand-constrained state updates preserve useful history, but they are not uniformly superior to zero-history task execution.

Limitations. The current evaluation starts from three published task suites with distinct interaction structures: TravelPlanner, ScienceWorld, and MARBLE. We preserve their native objectives, tools, and official scorers, then construct cross-lifecycle sequences by chaining semantically related complete tasks and instantiating new agents across successive rounds. It is not yet a native benchmark designed for long-lived multi-round multi-agent work, where round transitions, agent retirement, evidence replacement, provenance, and resource traces are first-class data. Our current green-efficiency evidence is based on inference calls, tokens, and elapsed time. Future work will establish benchmark suites and a unified evaluation framework for systematic energy and cost analysis across foundation models and MAS frameworks, supporting the development of clean, efficient, and sustainable green multi-agent systems.

6 Conclusion

This work first characterizes Dynamic Residual Decay (DRD) as a cross-lifecycle phenomenon in which historical influence can persist when positive utility for the current task is insufficient. To tackle DRD, we propose the Cross-Lifecycle Graph (CLG), which recovers the dependencies required by the current task. Across four models and three task suites, historical influence outlasts positive utility, increasing correction and verification demands as evidence changes. In recovery, CLG reduces calls by 39.79%, tokens by 29.95%, and elapsed time by 33.86% relative to Global Rerun while achieving higher or comparable official utility in most evaluated settings. By intersecting invalid influence with current demand, CLG retains valid historical computation, blocks stale influence, and concentrates recomputation on the dependencies needed now, providing a concrete path toward green multi-agent systems that preserve task utility while reducing execution cost.

AI USE STATEMENT

We used OpenAI Codex (GPT-5.6-sol) to polish parts of the manuscript, support research execution by implementing, debugging, and running the experimental code, and assist with the mathematical proofs. The research idea, problem formulation, and choice of the open-source datasets were developed entirely by the authors, who also wrote the initial manuscript draft. We reviewed and verified all AI-assisted writing, code, experimental procedures, and mathematical arguments, and take full responsibility for the final manuscript, claims, code, and reported results.

REPRODUCIBILITY STATEMENT

The anonymous repository at `https://anonymous.4open.science/r/QS678ANIUACN/` releases the DRD and CLG implementation, benchmark-neutral adapters, prompt templates, public run configuration, and a containerized smoke path. It documents the official sources for TravelPlanner, ScienceWorld, and MARBLE, and specifies the domains, drift levels, comparison strategies, generation limits, random seeds, and runtime environment variables. The paper and appendix provide the formal definitions, recovery algorithm, proof, experimental design, resource accounting, and statistical procedures. The repository's smoke script and guarantee self-check verify graph construction and demand-selected recovery; full benchmark runs use the official datasets and caller-supplied model credentials. Together, these materials support independent reimplementation and verification of the method and experimental protocol.

REFERENCES

Umut A. Acar, Guy E. Blelloch, and Robert Harper. Adaptive functional programming. In *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pp. 247–259. ACM, 2002.

Evan Chen, Shiqiang Wang, and Christopher G. Brinton. Fresh memory, stale plans: Dependency-scoped validation for distributed LLM-agent memory. arXiv preprint arXiv:2609.03340, 2026a.

Xiaoyu Chen, Fengge Wu, Zhao Junsuo, and Yun Fan. Condenseflow: Scalable latent space collaboration via semantic compression for multi-agent systems. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13694–13712. Association for Computational Linguistics, 2026b.

James Cheney, Laura Chiticariu, and Wang-Chiew Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

Hyeong Kyu Choi, Xiaojin Zhu, and Yixuan Li. Debate or vote: Which yields better decisions in multi-agent large language models? In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 112802–112834, 2025.

Chi Cui, Yixin Wu, Michael Backes, and Yang Zhang. Rethinking assessments of prompt injection attacks. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23773–23799. Association for Computational Linguistics, 2026.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502. Association for Computational Linguistics, 2026.

Shiwei Gao, Youmin Chen, and Jiwu Shu. Fast state restoration in LLM serving with HCache. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, pp. 128–143. ACM, 2025.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*, pp. 79–90. ACM, 2023.

Ashish Gupta and Inderpal Singh Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Engineering Bulletin*, 18(2):3–18, 1995.

Yuyang Hu, Jiongnan Liu, Jiejun Tan, Yutao Zhu, and Zhicheng Dou. Memory matters more: Event-centric memory as a logic map for agent searching and reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22389–22407. Association for Computational Linguistics, 2026.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 23142–23150, 2026.

Jinbo Liu, Defu Cao, Yifei Wei, Tianyao Su, Yuan Liang, Yushun Dong, Yan Liu, Yue Zhao, and Xiyang Hu. Topology matters: Measuring memory leakage in multi-agent LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 39728–39746. Association for Computational Linguistics, 2026.

Andrey Mokhov, Neil Mitchell, and Simon Peyton Jones. Build systems à la carte. *Proceedings of the ACM on Programming Languages*, 2(ICFP):79:1–79:29, 2018.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pp. 1–22. ACM, 2023.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

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

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298. Association for Computational Linguistics, 2022.

Zhenhua Wang, Chunlei Wang, Yue Geng, and Bang Wang. Agent newsroom: Efficient chronological report generation via dynamic multi-agent collaboration. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25063–25084. Association for Computational Linguistics, 2026.

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 conversation. arXiv preprint arXiv:2308.08155, 2023.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. arXiv preprint arXiv:2402.01622, 2024.

Ruozhen Yang, Yucheng Jiang, Yueqi Jiang, Priyanka Kargupta, Yunyi Zhang, and Jiawei Han. Grounding agent memory in contextual intent. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12008–12042. Association for Computational Linguistics, 2026.

Jialong Zhou, Lichao Wang, and Xiao Yang. GUARDIAN: Safeguarding LLM multi-agent collaborations with temporal graph modeling. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 8881–8909, 2025.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyin Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Robert Tang, Heng Ji, and Jiaxuan You. MultiAgentBench: Evaluating the collaboration and competition of LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8580–8622. Association for Computational Linguistics, 2025.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. In *Proceedings of the 34th USENIX Security Symposium (USENIX Security)*, pp. 3827–3844. USENIX Association, 2025.

A Related Work Positioning

The main text introduces DRD as the phenomenon of persistent historical influence whose positive utility changes across tasks, and CLG as a recovery formulation for this setting. Table 3 summarizes how CLG relates to the neighboring lines of work discussed in Section 2.

Table 3: Positioning of CLG relative to neighboring approaches.

Line of work	Primary update setting	Cross-lifecycle recovery perspective in CLG
Incremental computation and build systems	Recompute results affected by changed inputs	Combine invalidation reachability with the outputs demanded by the current task
Provenance and view maintenance	Connect source records to derived results	Preserve versioned artifacts and actual consumption edges across agent lifecycles
Agent memory and persistent state	Retrieve information across rounds or tasks	Separate continued influence from present positive utility under task drift
Dynamic agent topology	Adapt active agents, links, or routing	Recover historical dependencies after producers retire while successors and demands change

CLG instantiates demand-aware incremental recovery for cross-lifecycle multi-agent execution. Its formulation combines forward invalidation reachability, backward demand support, replacement boundaries, and dependency-ordered regeneration around the DRD phenomenon.

B Additional Method Details

This appendix follows the subsections of Section 3. It describes the runtime substrate and provenance instrumentation, records the version convention, gives the matrix implementation of both traversals, and proves Theorem 1. CLG uses the execution state and checkpoints exposed by the runtime, together with the recorded version reads and consumption edges described below.

B.1 Runtime substrate and provenance instrumentation

We implement the execution substrate with LangGraph's `StateGraph`. A `StateGraph` specifies the typed execution state, nodes, edges, and conditional transitions. When compiled with a checkpointer, each execution thread can retain successive state snapshots identified by a thread identifier and checkpoint identifier. The runtime can inspect the state history, restore a selected checkpoint, and apply a controlled state update before continuing execution. These settings let us create matched branches from a common checkpoint: one branch keeps selected historical artifacts readable, whereas the other marks them unavailable while keeping the task, model, random seed, and preceding state fixed. This is the counterfactual construction used to compare executions with and without history in the DRD measurement.

Each artifact, message, decision, and versioned state in a checkpointed execution is mapped to a CLG vertex $v \in V_t$; lifecycle labels record the agent and round that produced the vertex. Each state update creates a new version pointer rather than overwriting the historical record. The runtime stores

the producer, lifecycle, validity status, checkpoint, and artifact identifier for that version. A state-version read is logged with the consumer node and the version actually resolved at execution time. We append an edge (u, v) when the execution of v consumes the recorded version of u, yielding the actual consumption relation E_t used by CLG.

Checkpoint restores recover the corresponding version pointers and validity states. Tool calls are recorded as artifact-producing events with their input versions and output artifacts. Artifact provenance links messages, observations, plans, and decisions to their producing node and consumed inputs. Agent retirement changes the lifecycle label while retaining the checkpoint and artifact history for later recovery. LangGraph is broadly used for stateful and multi-agent workflows, so implementing CLG at this StateGraph and checkpoint boundary gives the method a practical integration point for existing agent systems rather than requiring a new orchestration runtime.

The instrumentation adds graph and provenance bookkeeping but does not require an additional model call. We report model calls, tokens, and elapsed time in the same post-invalidation accounting window used for all recovery strategies; graph traversal and version resolution are tracked separately from these model-execution totals.

B.2 Graph records, versions, and validity boundaries

The graph contains three pieces of information for each node: its producer and lifecycle label, the version or validity status used at the current task, and the actual inputs consumed when it was created. The graph is

$$\mathcal{G}_t = (V_t, E_t, \lambda_t, \sigma_t), \qquad (u, v) \in E_t \iff v \text{ consumed the recorded version of } u. \tag{11}$$

Edges point from an input to the state that used it. New states are appended in event order; retirement changes only λ_t. A valid replacement is represented by a separate node and a version pointer, rather than by rewriting the old node. This preserves the historical record while allowing current execution to read the replacement.

Let A_t be the Boolean adjacency matrix of E_t. For a replacement set K_t, the mask used by backward demand tracing is

$$A_t^{\text{valid}} = A_t \odot (\mathbf{1} - \mathbf{1}\mathbf{1}_{K_t}^{\top}), \qquad (\mathbf{1}_{K_t})_v = \mathbf{1}_{\{v \in K_t\}}, \tag{12}$$

where $\odot$ is elementwise multiplication. Thus a replacement node remains readable, but its historical parents do not force another recovery pass. If no replacement is available, the corresponding incoming edges remain active.

B.3 Matrix reachability and empirical implementation

The forward recurrence starts from the one-hot row vector $\mathbf{e}_r$ and uses the ordinary adjacency A_t; the reverse recurrence starts from the demand indicator $\mathbf{d}_t$ and uses $(A_t^{\text{valid}})^{\top}$. All operations are Boolean:

$$\begin{aligned} &\mathbf{f}^{(0)} = \mathbf{e}_r, \qquad \mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} \vee (\mathbf{f}^{(k)} A_t), \qquad \mathbf{h} = \bigvee_{k=0}^{n-1} \mathbf{f}^{(k)}, \\ &\mathbf{b}^{(0)} = \mathbf{d}_t, \qquad \mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} \vee \big(\mathbf{b}^{(k)} (A_t^{\text{valid}})^{\top}\big), \qquad \mathbf{b}^{\star} = \bigvee_{k=0}^{n-1} \mathbf{b}^{(k)}. \end{aligned} \tag{13}$$

The forward support is $H_t = \{v : h_v = 1\}$ and the backward support is $B_t = \{v : b_v^{\star} = 1\}$. Because the graph is finite and acyclic, every simple path has at most $n - 1$ edges, so the finite disjunction equals the transitive closure. An adjacency-list implementation evaluates the same recurrences by breadth-first or depth-first search. Each reachable node and edge is visited a constant number of times, giving $O(|V_t| + |E_t|)$ graph time and space.

The selected set can therefore be computed either by Boolean vectors or by set intersection:

$$Z_t = H_t \cap B_t = \{v : \exists d \in D_t,\ r \rightsquigarrow v \rightsquigarrow d \text{ in the version-aware graph}\}. \tag{14}$$

RQ4 perturbs recorded edges and compares this set with the unperturbed set. Missing edges can remove a true path and reduce recall; extra edges can add a false path and reduce precision.

B.4 Local recovery and output synthesis

Before executing any selected node, the runtime revokes reads from every invalid node in H_t. Let

$$U_t = Z_t \setminus K_t, \qquad (v_1, \ldots, v_m) = \mathrm{TopoSort}(\mathcal{G}_t[U_t]). \tag{15}$$

For a selected node v_i, define the input resolver

$$\eta_i(u) = \begin{cases} u^+, & u \in \{v_1, \ldots, v_{i-1}\} \text{ and a new version was produced,} \\ a_t(u), & u \in K_t \text{ and the replacement is valid,} \\ u, & u \notin H_t \text{ and } u \text{ remains valid,} \\ \bot, & \text{otherwise.} \end{cases} \tag{16}$$

A $\bot$ input blocks publication and triggers an expanded recovery or Global Rerun. When all required inputs are resolved, the node is regenerated as

$$v_i^+ = F_i(\{\eta_i(u) : u \in P_t(v_i)\}, x_t, T_i)\,, \qquad E_t \leftarrow E_t \cup \{(\eta_i(u), v_i^+) : u \in P_t(v_i)\}. \tag{17}$$

The version pointer is updated only after this append. Consequently, a later selected node reads v_i^+ rather than the invalid predecessor. Final synthesis is

$$y_t = S(\{a_{t+1}(d) : d \in D_t\})\,. \tag{18}$$

States in $H_t \setminus Z_t$ remain unreadable and are not silently reused in a later synthesis.

B.5 Guarantee and proof of Theorem 1

We prove the range identity and safety equation first, then state the additional condition needed for the inclusion equation.

Path characterization. By construction, $h_v = 1$ iff there is a directed path from r to v: induction on k proves the forward recurrence for paths of length at most k, and taking the disjunction over $k \le n-1$ covers every simple path. The same induction applied to the reverse recurrence proves $b_v^\star = 1$ iff v reaches some $d \in D_t$ without crossing a replacement boundary. Hence $v \in Z_t$ iff it lies on a recorded invalidation-to-demand path. This proves the range identity in Eq. 7.

Safety of local recovery. Take a demanded output d. Every selected predecessor of d is either in K_t or appears earlier in the topological order of U_t. The resolver therefore returns a valid replacement, a newly generated version, or an unaffected valid state. Induction over $(v_1, \ldots, v_m)$ shows that no published node in the selected subgraph has an unreadable input. The runtime guard prevents any state in $H_t \setminus Z_t$ from entering synthesis, proving Eq. 8.

Conditional minimality. Suppose each selected node is sensitive along at least one path to a demanded output when its recorded invalid input is changed with the shared evidence and tools fixed, and there is no unrecorded bypass or replacement. If an update set omits $u \in U_t$, follow a path from the invalid source through u to a demanded node. Parent sensitivity propagates the difference to that demand, so the omitted set cannot guarantee the Global Rerun output. Therefore every reliable recorded-state update contains U_t, which proves Eq. 9. Since CLG updates exactly U_t and reuses K_t, it is inclusion-minimal under these additional assumptions. Without parent sensitivity or complete lineage, only the conservative path-sound claim is justified.

B.6 Structural recovery and task utility evaluation

Theorem 1 establishes the structural recovery range, stale-state exclusion, and dependency-ordered update conditions of CLG. RQ3 evaluates the resulting recovery behavior through official task utility under matched execution conditions. These two analyses characterize complementary aspects of CLG: graph-level recovery correctness and task-level execution outcomes.

B.7 Scope accounting and resource measures

The structural fraction excluded from immediate recovery is

$$\rho_t = 1 - \frac{|Z_t|}{|H_t|} \quad (H_t \neq \varnothing). \tag{19}$$

For any resource C measured after invalidation, we report the paired saving

$$\mathrm{Saving}(C) = 1 - \frac{C(\mathrm{CLG})}{C(\text{Global Rerun})}. \tag{20}$$

Calls, tokens, and elapsed time are separate resource coordinates: node count determines the selected graph scope, while node-specific inference and tool costs determine the measured resource. The runtime bookkeeping described in Section B.1 is tracked separately from these model-execution coordinates.

C Experimental Design and Statistical Protocol

Sampling and scale. The main text defines the benchmark tasks, continuous-round construction, and model selection. We report the evaluation at the matched task–drift level: TravelPlanner and MARBLE each provide 48 matched task–drift conditions per model, while the validated ScienceWorld panel provides 24. These conditions are evaluated under the same budget and scoring protocol. End-to-end recovery uses DeepSeek-v4.1-Flash and GLM-5.3-Flash with temperature 0.2 and a 4096-token generation limit; GLM uses low reasoning intensity. Across the five comparative recovery strategies plus the ungoverned condition (using the corresponding experimental configuration from RQ1), the 48-condition TravelPlanner cells account for approximately 7,771 model/API calls (DeepSeek) and 5,282 calls (GLM), and the 48-condition MARBLE cells account for approximately 12,441 calls (DeepSeek) and 5,827 calls (GLM). Thus, the analyzed RQ3 panel contains 1,440 strategy-arm executions and approximately 57,329 post-invalidation model/API calls; the corresponding token total is approximately 170.4 million. All six recovery conditions are evaluated on matched conditions using the same task assignments, run seed 42, and task–drift assignments.

Comparisons. Global Rerun regenerates the full recorded artifact graph. HCache-Safe conservatively refreshes dependency-linked history under a TTL; PlanFence-SM verifies dependencies of actions specified by the current plan and bounds replanning. CLG (forward-only) follows the invalid source's influence range without backward selection. The ungoverned condition retains invalid states. We compare strategies within the same task suite, model, task sequence, and drift condition using common scoring and resource-accounting definitions. End-to-end CLG uses exact forward reachability and version-aware backward demand slicing; RQ4 tests the corresponding graph selection under demand and edge perturbations.

Measures. RQ1 reports influence survival, positive-utility survival, and mismatch occupancy as defined in Section 3. RQ2 measures propagation-specific final-outcome loss, correction cost, and verification behavior. RQ3 checks whether the affected states required by current demand have been handled and whether invalid states remain in use, and reports official task utility. Task-normalized utility scores follow the normalization and aggregation defined in Section 4.3; savings are relative to Global Rerun. RQ4 reports deferred affected updates, recall, and precision relative to unperturbed exact reachability. Paired inference uses the task sequence as its unit (Appendix C.3).

C.1 Task sequences and drift construction

Zero drift retains the original task; low, medium, and high drift use quantiles of structural distance from the original task to the later task. TravelPlanner uses a weighted difference in task fields, ScienceWorld a lexical distance between task descriptions, and MARBLE differences in the cause of the invalidated root. Drift levels are assigned before task execution within each domain, and outcomes under all four conditions are compared for the same history.

C.2 MEASURES AND AGGREGATION

Positive utility is determined from the official utility difference ΔU between using and not using history on the same task. The three task suites retain different official scoring procedures, so their utility differences have different score resolutions. TravelPlanner uses the mean of the official binary hard and commonsense checks, ScienceWorld uses the simulator's native progress score (reported on a 0–100 scale and normalized to $[0, 1]$), and MARBLE uses the official set-based F1 score. For each domain d, let

$$s_d = \min\{|\Delta U_i| : |\Delta U_i| > 0,\ i \in d\} \quad \text{and} \quad \varepsilon_d = \frac{s_d}{2}. \tag{21}$$

The frozen panel gives $s_d = 0.0500$ for TravelPlanner, 0.0100 for ScienceWorld, and 0.1667 for MARBLE, yielding $\varepsilon_d = 0.025$, 0.005, and 0.08335, respectively. Thus ε is a domain-specific score-resolution convention: $\Delta U > \varepsilon_d$ denotes positive utility, $\Delta U < -\varepsilon_d$ denotes stale utility, and the interval $|\Delta U| \leq \varepsilon_d$ is neutral. The full-step value s_d is used as the threshold-sensitivity specification. Each sequence records the influence indicator and ΔU at each drift level to calculate I, U, and M. Table 1 averages the three domains equally within each model; Table 4 averages the four models equally within each domain. For MARBLE's adopt-without-repoll executions, a direct retired-root use is counted when the ledger records the root in both the explicit adoption list and the finalization use set; this captures prompt-injected reports whose use is not emitted as a separate read event. Derived uses continue to require the recorded provenance path.

Every strategy in a recovery setting uses the same reference utility, selected as the largest mean official utility among strategies satisfying recovery completeness. Because the three official scorers produce utilities on different native scales and with different resolutions, RQ3 compares strategies within each task–model cell after dividing the mean official utility by that cell's reference utility. The resulting task-normalized values are averaged equally across cells, placing the heterogeneous task suites on a common comparative scale while preserving the native scorer within each cell. Resource savings are computed separately against Global Rerun within each setting and then macro-averaged. The reference values used as denominators are listed with the extended recovery results below.

C.3 PAIRED COMPARISONS AND UNITS OF ANALYSIS

End-to-end comparisons match strategies under the same task–drift condition and use 20,000 task-sequence-clustered bootstrap resamples with analysis seed 42. TravelPlanner and MARBLE each contain 48 matched task–drift units per model, corresponding to approximately 7,771/5,282 model/API calls for DeepSeek/GLM in TravelPlanner and 12,441/5,827 calls in MARBLE across the six strategies. ScienceWorld contains 24 validated matched units per model, corresponding to approximately 20,304/5,704 calls for DeepSeek/GLM. Formal combination-level interval estimation is applied only where the task-sequence support meets the preregistered threshold; ScienceWorld differences are therefore reported descriptively.

The counterevidence-timing comparison in RQ2 contains 21 parent tasks, four conditions, and three repeats with seeds 42, 43, and 44. Each of the two source-independence conditions includes 36 records from 12 parent tasks and repeated executions. Corresponding paired tests use parent-task differences; repeated records are not counted as independent parent tasks.

RQ4 generates 120 six-layer directed acyclic graphs with fixed graph seeds 0–119. Layer sizes are respectively 1, 7–10, 10–15, 12–18, 10–15, and 8–12 nodes. Each successor connects randomly to one to three parents in the preceding layer; further edges ensure that each node in a layer reaches at least one successor, and a small number of cross-layer edges are added. Nested demand sets are selected in a fixed random order from the final-layer terminals after rounding the demand fraction times the terminal count. Numbers of removed and added edges are computed from the original edge count times the perturbation fraction; added edges retain the layer direction. Each graph and perturbation level is repeated three times with edge-sampling seeds 42, 43, and 44. The demand experiment compares exact intersection with complete forward reachability; dependency perturbations compare with the unperturbed exact intersection. Intervals use 10,000 graph-level bootstrap resamples after averaging perturbation repeats within each graph.

C.4 Intervention-based validation of recorded influence and dependencies

We replace the content of a historical artifact and rerun from the same state immediately before its direct consumer, using two identical-content controls to assess output changes. Labels follow the consumer's reuse decision and content change relative to the controls rather than dependency declarations. The three models are DeepSeek-v4.1-Flash, GLM-5.3-Flash, and Qwen-3.8-Flash. Across these models, there are 14, 8, and 14 consumers, of which 9, 5, and 6 can be classified; 6, 2, and 5 are actually affected. The RQ1 ledger uses observed reads, adoption, and provenance to define execution influence, while this intervention tests whether changing the consumed artifact changes the direct consumer under a fixed preceding state. Reading relations identify all classifiable affected consumers, whereas semantic declarations identify 0, 0, and 2 across the three models. Consumers with unstable identical-content controls remain indeterminate and are counted neither as affected nor unaffected.

C.5 Reproducibility materials

The anonymous repository accompanying the submission releases the DRD and CLG implementation, benchmark-neutral adapters, prompt templates, a provider-neutral public run configuration, and a containerized smoke path. It identifies the official sources for TravelPlanner, ScienceWorld, and MARBLE but does not redistribute their raw data. The public configuration fixes the domains, drift levels, comparison strategies, generation limits, and seeds 42, 43, and 44, and requests that prompt, configuration, model, and seed identifiers be recorded with each run. The smoke script and guarantee self-check provide a lightweight verification of graph construction, reachability, and demand-selected recovery; complete benchmark runs require the official datasets and caller-supplied model credentials.

Models, providers, and container resources. The four model families in the phenomenon panel were paired with their first-party providers: DeepSeek-v4.1-Flash with the official DeepSeek API, GLM-5.3-Flash with Zhipu AI (BigModel), GPT-Terra-Medium with OpenAI, and Qwen-3.8-Flash with Alibaba Cloud Model Studio/DashScope. The client used an OpenAI-compatible request interface, while each run recorded its model and provider identifiers and used no model fallback. Workers ran in Docker containers built from the pinned `a2a-langgraph:smoke` image (`PYTHONUNBUFFERED=1`, `PYTHONPATH=/workspace/src`). The Docker Desktop Linux VM provided 7.61 GiB of memory; the launcher reserved 1,000 MiB and bounded the experiment budget to 6,790 MiB and at most 10 concurrent containers. Measured working sets were approximately 53 MiB for MARBLE containers, 0.92–1.08 GiB for ScienceWorld containers, and 1.16–1.28 GiB for TravelPlanner containers. CPU was scheduled by the Docker VM and launcher; no per-container CPU quota or pinning was used, so reproducibility depends on the stated VM budget and concurrency limits.

D Extended Phenomenon Results

DRD appears as a separation between historical influence and positive utility. Across sequences with changed tasks, we measure whether history still influences subsequent decisions and whether using it improves the official task score. Four drift levels characterize the distance of the later task from the original. Table 1 gives model-level values at all four drift levels; Table 4 reports registered task-suite summaries and mismatch occupancy.

Influence survival asks whether the history still matters to execution; positive-utility survival asks whether using it still helps. Across the four model rows, the highest drift level averages 77.1% influence and 10.4% positive utility. Parent-task bootstrap resampling retains the matched model panel within each task; trajectories differ by task and model.

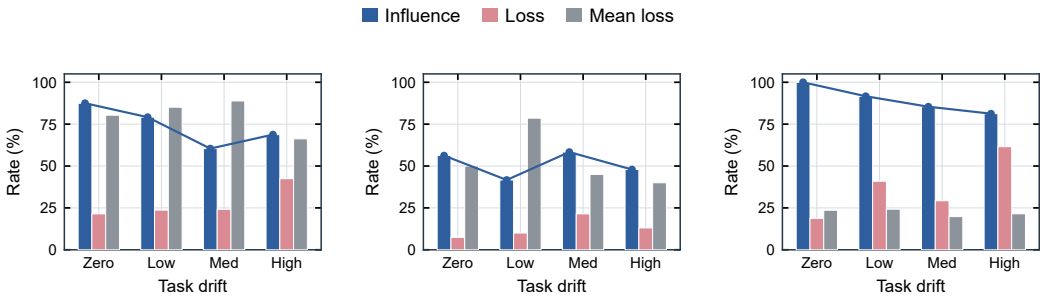
The GPT–MARBLE ledger was rechecked for utility-positive records that the original influence marker missed. Eleven such records carry both an explicit direct-adoption declaration and a matching finalization use, so they are counted as influence in the model row. Records with only a positive utility difference and no adoption, read, or provenance evidence remain excluded.

Table 4: Domain summaries of historical influence, positive utility, and mismatch across four drift levels. The four models are weighted equally within each domain. Mismatch occupancy directly counts cases with influence but insufficient positive utility; it is computed separately from the difference between the two survival rates. All entries are percentages.

Domain	Drift	Influence	Positive utility	Mismatch
MARBLE	Zero	85.4	27.1	58.3
	Low	87.5	22.9	66.7
	Medium	89.6	14.6	75.0
	High	79.2	4.2	75.0
ScienceWorld	Zero	87.5	25.0	64.6
	Low	79.2	12.5	68.8
	Medium	60.4	29.2	43.8
	High	68.8	14.6	58.3
TravelPlanner	Zero	95.8	62.5	37.5
	Low	85.4	35.4	54.2
	Medium	85.4	22.9	64.6
	High	83.3	12.5	72.9

In TravelPlanner, positive-utility survival falls from 62.5% at zero drift to 12.5% at high drift, while influence remains at 83.3%; mismatch occupancy increases from 37.5% to 72.9%. MARBLE reaches 75.0% mismatch at high drift, whereas ScienceWorld has 64.6% mismatch already at zero drift and 58.3% at high drift. These trajectories distinguish declining utility under task change from mismatch already present at the initial task.

E Extended Interaction Results



(a) ScienceWorld (b) MARBLE (c) TravelPlanner

Figure 7: Stage-wise influence, lower-utility loss, and conditional mean loss across task drift.

Once history remains influential, its consequences depend on the evidence available to later agents and when they can revise a decision. We vary access to current evidence, the arrival time of counterevidence, and whether agents' judgments share an evidential source, measuring final outcomes and correction behavior (Figure 7).

Matched cost reanalysis. The RQ1 execution records contain 576 matched readable–revoked conditions across 12 task set–model combinations, each with 12 sequences and four drift levels (1,152 second-round arms). The cost analysis joins the frozen official-utility difference to each arm's logged agent-call, input-token, output-token, and latency counters; MARBLE excludes shared first-round costs from its arm-specific increments. In the 32 ScienceWorld conditions with both lower readable-arm utility and more calls, paired differences total 254 calls, 12,003,546 input tokens, and 213,663 output tokens. Total-token differences nevertheless vary in sign within these conditions; the token sum is a descriptive workload total, not a per-task saving estimate. The correction experiment below uses different parents and remains a separate analysis.

New matched executions. Twelve parent tasks share the same source-history construction in six combinations of current or inherited evidence and correction budget 0, 1, or 3. Each combination contains 12 readable and 12 revoked branches; all 144 branch records passed execution and provenance validation. Source-side correction remains at three steps, while the successor branch receives its assigned budget. In this distinct online population, loss under current evidence is 33.3%, 0.0%, and 0.0% at budgets zero, one, and three, compared with 33.3% at all three budgets using inherited evidence alone. The stale branch's mean correction rounds are 0.000, 0.417, and 0.583 with current evidence and 0.000 at every budget with inherited evidence.

Parent-level reanalysis. A separate repeated comparison covers 54 parent tasks, with seeds 42, 43, and 44 averaged within each parent-condition. An offline reanalysis (20,000 parent bootstrap samples; seed 42) gives an evidence-by-budget interaction of 22.8395 percentage points with a 95% interval [11.7284, 35.1852]; an exact sign-flip over 12 nonzero parent contrasts yields $p = 0.00048828$. The parent contributions are 42 zeros, 11 contrasts of 100 points, and one contrast of 133.3 points. Deleting each parent in turn gives estimates of 20.7547–23.2704 points. The new 12-task execution and this repeated comparison have distinct statistical denominators.

Table 5: Shared versus independent evidential origins. Each condition contains 36 records from 12 parent tasks and repeat executions. The final row is the shared-minus-independent difference in percentage points. The 95% intervals for adoption, consensus, and independent rechecking are [2.8, 36.1], [33.3, 77.8], and [-36.1, -2.8], respectively. Multiple-comparison-adjusted $p = 0.0195$ for consensus and $p = 1.0$ for adoption.

Evidence origin	Adoption	Consensus	Independent recheck
Shared root	80.6%	69.4%	19.4%
Independent roots	61.1%	13.9%	38.9%
Shared − independent (pp)	+19.4	+55.6	−19.4

When external evidence can be revisited, raising the correction budget from zero to three increases the final pass rate from 48.1% to 70.4% and reduces propagation-specific loss relative to revocation from 22.2 points to zero. When agents can only inherit historical information, the corresponding pass rate changes from 48.8% to 48.1% and loss from 21.6 to 22.2 points. Additional reasoning corrects an inherited judgment when new evidence is available; processing the same history again does not produce a comparable gain. The shared-origin comparison is in Table 5. When external evidence can be revisited, raising the correction budget from zero to three increases the final pass rate from 48.1% to 70.4% and reduces propagation-specific loss relative to revocation from 22.2 points to zero. When agents can only inherit historical information, the corresponding pass rate changes from 48.8% to 48.1% and loss from 21.6 to 22.2 points. Additional reasoning corrects an inherited judgment when new evidence is available; processing the same history again does not produce a comparable gain.

The arrival time of counterevidence affects correction cost. Before a decision is committed, both propagation-span loss and correction load are zero. After commitment, they rise to 0.9680 [0.6670, 1.2700] and 1.7300 [0.9680, 2.5710], respectively ($p = 3.05 \times 10^{-5}$). Sharing an evidential source raises consensus from 13.9% with independent sources to 69.4%, so agreement among agents need not reflect independent confirmation. Across the tested propagation spans of one to three generations, propagation-specific loss remains 22.2 percentage points.

F Extended Recovery Results

The end-to-end recovery comparison uses DeepSeek-v4.1-Flash and GLM-5.3-Flash, two inference-efficient model families selected to support matched repeated execution across all recovery strategies while retaining cross-model coverage. The four-model RQ1 study provides broader evidence for the DRD phenomenon; this section focuses on controlled strategy-level recovery accounting.

The native task utilities are retained for within-cell comparisons. TravelPlanner and MARBLE report utilities on a $[0, 1]$ scale, whereas ScienceWorld reports the simulator score on a $[0, 100]$ scale; their score resolutions also differ because the underlying official scorers aggregate different

checks. Consequently, cross-suite aggregation uses the reference-normalized utility

$$\widetilde{U}_{b,m}(r) = \frac{\overline{U}_{b,m}(r)}{U^{\star}_{b,m}}, \qquad U^{\star}_{b,m} = \max_{r \in \mathcal{R}_{\text{complete}}} \overline{U}_{b,m}(r), \tag{22}$$

within each benchmark–model cell (b, m). Here $\overline{U}_{b,m}(r)$ is the mean official utility for recovery strategy r, and the maximum is taken over strategies satisfying the recorded recovery-completeness checks. We then macro-average $\widetilde{U}$ across cells. This normalization makes the RQ3 quality summary comparable across the three official scoring systems; cell-level utilities remain reported in Table 6, with ScienceWorld shown after the stated division by 100. Accordingly, the RQ3 utility summary measures relative utility retention under each native scorer and its cross-cell aggregation, rather than absolute task-quality differences across suites. Absolute quality evaluation for multi-round multi-agent collaboration requires benchmark suites and metrics designed for that setting, which remains a direction for future work.

Table 6: Official task utility, task-normalized utility, and CLG resource savings relative to Global Rerun in each recovery cell; ScienceWorld utility is divided by 100.

Task set	Model	CLG	Global Rerun	Norm. utility	Calls (%)	Tokens (%)	Time (%)
TravelPlanner	DeepSeek	0.6266	0.5772	0.9600	30.69	19.22	22.04
	GLM	0.6434	0.6340	1.0000	35.23	20.73	25.66
MARBLE	DeepSeek	0.9931	0.9661	1.0000	58.47	42.73	48.61
	GLM	0.8646	0.9514	0.9088	40.72	22.23	39.46
ScienceWorld	DeepSeek	0.3800	0.3225	1.0000	47.95	55.90	42.11
	GLM	0.5000	0.5000	1.0000	25.69	18.89	25.26

We compare CLG with two extreme controls, two neighboring strategies, and one demand-selection variant across three domains and two models. Figure 5 summarizes task utility and resource savings; Table 6 reports utility and resources by domain and model.

Table 7: Reference utilities used as normalization denominators for the recovery evaluation.

Task set	Model	Reference utility
TravelPlanner	DeepSeek	0.6527
TravelPlanner	GLM	0.6434
MARBLE	DeepSeek	0.9931
MARBLE	GLM	0.9514
ScienceWorld	DeepSeek	0.3800
ScienceWorld	GLM	0.5000

Each strategy in a recovery cell uses the same reference utility in Table 7. The reference is the largest mean official utility among strategies satisfying the recorded recovery-completeness checks in that cell; the resulting normalized utilities are macro-averaged with equal cell weight.

All matched CLG recovery runs satisfy the recorded completion checks. The task-normalized utility index is 0.9781 for CLG and 0.9485 for Global Rerun. Relative to Global Rerun, model calls decline by 39.79%, tokens by 29.95%, and elapsed time by 33.86%. Call savings range from 25.69% to 58.47%, token savings from 18.89% to 55.90%, and time savings from 22.04% to 48.61%. All resource quantities use the same post-invalidation accounting window.

F.1 Variation across domains and models

Compared with Global Rerun, the point estimate of official task utility is higher for TravelPlanner–DeepSeek, TravelPlanner–GLM, MARBLE–DeepSeek, and ScienceWorld–DeepSeek; equal for ScienceWorld–GLM; and lower for MARBLE–GLM (Table 6). Each TravelPlanner and MARBLE combination contains 48 matched task–drift units and the corresponding six-strategy call totals are reported above; each ScienceWorld combination contains 24 validated matched units and its call

total is reported above. Raw official utilities retain the scoring scale of the corresponding task set. Table 8 reports elapsed seconds and paired utility differences.

Table 8: CLG and Global Rerun mean recovery-branch seconds and paired official-utility differences. Intervals resample task sequences; ScienceWorld entries give descriptive differences.

Task set	Model	CLG (s)	Global Rerun (s)	ΔU [95% CI]
TravelPlanner	DeepSeek	37.52	48.13	$+0.049$ [-0.024, 0.120]
	GLM	61.35	82.53	$+0.009$ [-0.052, 0.075]
MARBLE	DeepSeek	37.18	72.35	$+0.027$ [0.007, 0.051]
	GLM	47.88	79.09	-0.087 [-0.163, -0.014]
ScienceWorld	DeepSeek	124.90	215.76	$+0.058$
	GLM	82.55	110.45	0.000

F.2 Recovery completeness and computational scope

All recovery-completeness conditions hold for the matched task conditions. Exact forward and version-aware backward traversal select nodes linking the invalid source to current demand; savings depend on branching structure, the position of demand, and node-specific execution cost. Controlled graph experiments next isolate changes in demand and dependency records.

G Demand and Dependency Perturbations

RQ4 reuses 120 seeded irregular directed acyclic graphs. The fraction of demanded outputs takes values 0.25, 0.50, 0.75, and 1.00 (Table 9). At demand fraction 0.50, the fraction of dependencies deleted or added takes values 0.00, 0.05, 0.10, 0.25, and 0.50, with three seeded repeats per graph and condition. Exact forward–reverse reachability determines the selected set (Table 10). An unperturbed graph with the same demand supplies the reference; empty selections have precision 0.0. Mean coverage is first computed within graph across repeats; 10,000 graph-resampling draws give 95% percentile intervals.

Table 9: Affected updates deferred under current demand (120 graphs; graph-bootstrap 95% intervals).

Demand fraction	Mean deferred (%)	95% interval (%)
0.25	42.20	[40.62, 43.83]
0.50	22.72	[21.63, 23.86]
0.75	9.94	[9.12, 10.77]
1.00	0.00	[0.00, 0.00]

Table 10: Recovery recall and precision under perturbed dependency records. Each perturbation and fraction includes 120 graphs and three repeats (360 observations). Positive support and demand fraction 0.50 are fixed; precision for an empty selection is defined as zero.

Perturbation	Missing recall	Missing precision	Extra recall	Extra precision
0.00	100.0%	100.0%	100.0%	100.0%
0.05	94.18%	100.00%	100.00%	98.74%
0.10	87.93%	100.00%	100.00%	97.38%
0.25	65.77%	100.00%	100.00%	94.99%
0.50	22.36%	85.00%	100.00%	92.53%

As current demand contracts, selection excludes affected branches that no longer lead to a requested output. When the demand fraction increases from 0.25 to 1.00, deferred affected updates fall from 42.20% to zero; the reference selection has full recall by construction.

Missing and extra dependencies have different effects. At deletion fractions of 0.05, 0.10, 0.25, and 0.50, recall is 94.18%, 87.93%, 65.77%, and 22.36%, respectively. Added edges retain 100.00% re-

call; precision at an addition fraction of 0.50 is 92.53%. Missing edges mainly sever required paths, while added edges mainly expand the recovery set. Demand determines potential scope reduction, whereas the completeness of recorded dependencies determines coverage.

H SUPPORT OF BIDIRECTIONAL REACHABILITY

Proposition. On a finite directed dependency graph, let H_t contain the invalidated source r and all its descendants. Let B_t contain every node with a path to current demand D_t, stopping backward traversal at valid replacements. A node belongs to $Z_t = H_t \cap B_t$ if and only if it lies on a recorded dependency path from r to a demanded output that does not trace a replacement's historical inputs.

Path characterization. If $v \in H_t \cap B_t$, a path from r to v and a path from v to a demanded output concatenate to give an invalidation-to-demand walk through v. Conversely, each node on such a path is reachable from r and can reach a demanded output, so it belongs to both sets. Stopping reverse traversal at a replacement removes paths that require its superseded historical inputs; the replacement remains available for execution.

Computation. A forward traversal from r and a multi-source traversal backward from D_t visit each graph node and edge at most a constant number of times. Intersecting their visited sets and topologically ordering selected nodes on a dependency DAG take $O(|V_t| + |E_t|)$ total graph time.

Dependency errors. These properties hold relative to the recorded graph. Missing edges may delete true paths and reduce recall; extra edges may introduce paths and reduce precision. RQ4 perturbs these two kinds of error independently to estimate sensitivity.