TASKGRAPH: DYNAMIC HIERARCHICAL MEMORY FOR LONG-HORIZON LANGUAGE MODEL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon tasks require agents to sustain extended interactions and multi-step reasoning. As context grows, fragmented memories can draw attention toward local details, causing agents to lose track of task progress and forget earlier goals and constraints. Existing methods organize historical information through hierarchies and relational links. These structures improve retrieval but do not explicitly organize memory around task progress and the requirements of each execution step. We propose TaskGraph, a unified model of task and memory that represents memory as the evolving structure of task execution. Task goals, stages, steps, and atomic records of constraints, evidence, and variables form a hierarchical graph linked by inclusion, mainline, and dependency relations. The graph therefore turns memory into an execution structure: it binds goals to completed work and makes the constraints, variables, and evidence required by the next step immediately accessible. For each current step, TaskGraph selects relevant historical dependencies and constructs its execution context from the graph. Observed results are then written back to update the memory available to subsequent steps. This process allows task execution and memory to evolve together within a shared representation. On StableToolBench, TaskGraph improves macro-average final answer correctness by 14.97 percentage points over the same-backbone DFS baseline using Qwen3-14B-AWQ, validating the effectiveness of jointly modeling task progression and memory.

1 INTRODUCTION

Large language models (LLMs) can process increasingly long contexts and generalize across question answering, tool use, and interactive tasks (Dai et al., 2019; Yao et al., 2023b; Schick et al., 2023). Recent systems extend or fine-tune models for million-token inputs (Chen et al., 2024; Ding et al., 2024), while sparse-attention and KV-cache methods improve long-context inference (Jiang et al., 2024; Li et al., 2024). Nevertheless, benchmark studies show that nominal context length does not guarantee reliable use of long-range evidence (Hsieh et al., 2024; Modarressi et al., 2025). Access to a long interaction history, however, does not ensure that an agent can track task progress or use past information effectively. During execution, relevant information accumulates across dialogue, tool outputs, page states, and documents. The agent must connect this information to the current stage, completed steps, applicable constraints, and prerequisites for subsequent actions. A flat transcript leaves these relationships implicit, forcing the agent to repeatedly infer the execution state from the full history. This not only increases context demands but also makes it harder to follow the intended execution order and stay aligned with the original goal.

External memory complements longer context windows by organizing and retrieving earlier information (Sukhbaatar et al., 2015; Khandelwal et al., 2020; Wu et al., 2022). Retrieval-augmented generation (RAG) typically returns passages that are semantically close to a query (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020). Semantic relevance to a query alone does not establish whether a passage applies to the current stage or supplies a prerequisite or supporting evidence for the next action. Hierarchical and graph retrieval systems add abstraction and relations (Sarthi et al., 2024; Rezazadeh et al., 2025). For example, GraphRAG (Edge et al., 2024) constructs an entity knowledge graph and community summaries to integrate information across documents.

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.

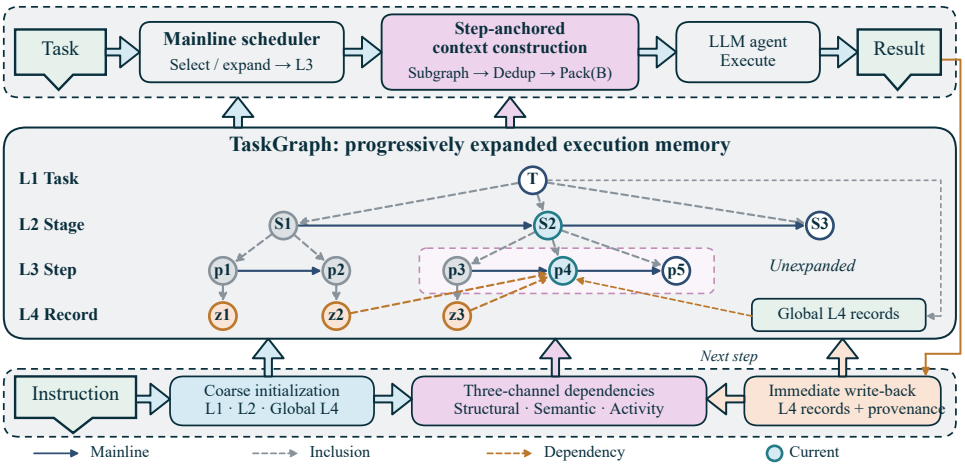

Figure 1: Overview of TaskGraph. Top: the execution pipeline from task input to output. Middle: progressively expanded execution memory, the core L1–L4 graph structure. Bottom: the memory-update pipeline for coarse initialization, three-channel dependency construction, and immediate write-back.

However, these structures primarily organize knowledge about the source material rather than explicitly represent the stages and execution dependencies of an ongoing task. Long-horizon agents further require a shared representation for memory and execution control. The central challenge is to jointly represent task goals, execution progress, and historical evidence. This shared representation should guide both the next action and the information needed to perform it.

We propose TaskGraph, a unified representation of task structure and execution memory for long-horizon LLM agents. The graph records both what the agent needs to accomplish and the constraints, evidence, and variables established during execution. Starting from a coarse task skeleton, the agent expands stages using accumulated state, selects historical dependencies to construct context for the current step, and writes execution results back immediately. Figure 1 summarizes this closed loop between memory and task progression.

Our contributions are threefold:

- We propose TaskGraph, a unified model of task and memory that represents agent memory as the evolving execution structure of an ongoing task. Goals, stages, steps, and observed states share a hierarchical graph, making task progress and the role of historical information explicit. This formulation grounds memory in what the agent is trying to accomplish, what it has completed, and what subsequent steps require.
- We further define coupled update rules for this shared graph: accumulated state determines when and how upcoming stages are expanded, dependency construction grounds each step in prior evidence, and immediate result write-back updates the state available to later steps.
- We evaluate TaskGraph across five settings spanning tool use, long interaction histories, dialogue summarization, multi-document question answering, and long-form generation. Compared with the evaluated baselines, TaskGraph improves final-answer correctness in tool use, historical evidence coverage in summarization, and long-range plot advancement and local narrative continuity in generation.

2 Related Work

External memory and retrieval. External retrieval extends the information available to language models through relevant evidence, as in RAG, REALM, and dense passage retrieval (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020). FiD and RETRO further improve evidence fusion and retrieval scale (Izacard & Grave, 2021; Borgeaud et al., 2022). These advances help models retrieve and integrate relevant information. However, for an ongoing task, the agent must still determine which

earlier results and constraints are needed for the current step. TaskGraph records these dependencies explicitly, connecting each step to the information required for its execution.

Hierarchical and graph memory. Structured memory adds relationships and abstraction beyond passages. RAPTOR uses recursive clustering and summarization, while MemTree dynamically integrates information into a semantic hierarchy (Sarthi et al., 2024; Rezazadeh et al., 2025). GraphRAG aggregates evidence through entity relations and community summaries (Edge et al., 2024), and A-MEM automatically links and updates memories (Xu et al., 2025). MAGMA further combines semantic, temporal, causal, and entity graphs for query-adaptive retrieval (Jiang et al., 2026). These methods organize relationships among memories to support retrieval. TaskGraph extends this focus to the relationship between memory and task execution, representing goals, stages, steps, and observed constraints, evidence, and variables in a shared graph. This representation makes explicit how earlier results support upcoming steps, guiding both task progression and context construction.

Reasoning and acting agents. Memory organization must also be coordinated with task progression. Chain-of-thought and plan-and-solve prompting organize intermediate reasoning (Wei et al., 2022; Wang et al., 2023); Tree of Thoughts and ReAct support alternative-path exploration and action interleaving, respectively (Yao et al., 2023a;b). Reflexion extends this process by retaining feedback in episodic memory to improve subsequent attempts (Shinn et al., 2023). Agent Workflow Memory induces reusable workflows from experience and supplies them to guide subsequent actions, with both offline and online induction (Wang et al., 2025). These approaches improve reasoning and action selection through intermediate reasoning, feedback, and reusable workflows. TaskGraph addresses the complementary need to track how the current task progresses and how each action changes the information available for later steps. By representing task progress and observed states together, it allows subsequent actions to draw on the results and constraints established during execution.

3 METHOD

TaskGraph represents a long-horizon task as an execution memory graph that evolves with user instructions and execution results. From the user request, it creates an initial task skeleton comprising the overall goal (L1), task stages (L2), and known global constraints, evidence, and variables (L4). As execution progresses, the scheduler selects the current L2 stage, and the agent expands it into executable L3 steps using the accumulated task state. For each selected step, the agent retrieves and filters dependency candidates through three channels, constructs its execution context, and performs the step. The observed results are immediately written back to guide subsequent steps and stage expansion. Further implementation details are provided in Appendix A.

3.1 A PROGRESSIVELY EXPANDED MULTI-LEVEL MAINLINE

At discrete task time t, TaskGraph is a typed directed graph:

$$\begin{aligned} &G_t = (V_t, E_t), \qquad V_t = \bigcup_{\ell=1}^{4} V_t^{(\ell)}, \\ &E_t \subseteq V_t \times V_t \times \mathcal{T}, \qquad \mathcal{T} = \{\textsc{Inclusion}, \textsc{Mainline}, \textsc{Dependency}\}, \end{aligned} \tag{1}$$

where V_t and E_t are the node and edge sets, $V_t^{(\ell)}$ contains the nodes at level ℓ, and $\mathcal{T}$ denotes the edge types. In online execution, the four levels represent the overall task (L1), stages (L2), executable steps (L3), and atomic records of constraints, evidence, and variables (L4). L4 does not introduce a separate state category; an observation is retained either as evidence or as the updated value of a variable. Offline evaluations map recorded trajectories or events to these levels, as detailed in Appendix A. Each node retains its content, hierarchical scope, execution state, and provenance. An INCLUSION edge links a node to its hierarchical parent. A MAINLINE edge records execution order in online tasks and historical order in offline task mappings. A DEPENDENCY edge links a prerequisite or evidence source to the node that requires it. After an executable step is selected, TaskGraph chooses dependency sources from candidate nodes and creates DEPENDENCY edges before execution (Section 3.2); such edges may cross stages and levels. L2 and L3 nodes carry one of four execution states: PENDING, ACTIVE, DONE, or DROPPED.

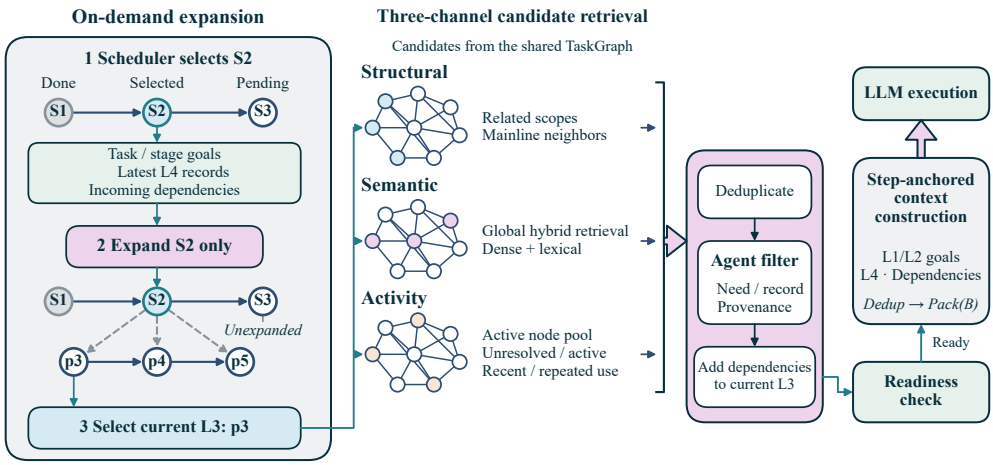

Figure 2: On-demand expansion and step-anchored context construction. When execution reaches stage S_2, it is expanded into L3 steps using accumulated task state. Steps proceed in mainline order. For the selected step p_3, structural, semantic, and activity candidates are deduplicated and filtered to establish dependencies on prior states and results. After a readiness check, the step-anchored context is constructed, deduplicated, and serialized for LLM execution.

As shown in Figure 2, initialization creates or updates the L1 goal, L2 stages, and explicitly stated global L4 constraints, evidence, and variables. The agent orders stages through MAINLINE edges without generating all future L3 steps. Subsequent instructions can revise the goal, stage boundaries, and global constraints by updating the affected graph paths. Execution follows the L2 mainline. The scheduler continues an ACTIVE stage or activates the first PENDING stage. When the selected stage s_t has not been expanded, the agent generates an ordered sequence of executable steps:

$$\mathcal{P}_t(s_t) = \langle p_{t,1}, \ldots, p_{t,m_t} \rangle = \mathrm{Expand}_{\mathrm{agent}}(G_t, s_t), \tag{2}$$

where $\mathcal{P}_t(s_t)$ is the ordered plan for stage s_t, and $p_{t,j}$ is its jth executable step. Expansion uses the task and stage goals, global constraints, evidence, and variables in scope, incoming dependencies, and results produced by completed steps. Each generated step becomes an L3 node attached to s_t and ordered along its local MAINLINE. The step count m_t is determined from the current stage requirements. An expanded stage retains its decomposition unless an instruction requires revision. The scheduler selects the current L3 step v_t^{exec} according to its state and local mainline order; the agent then constructs its dependencies before execution.

3.2 THREE-CHANNEL DEPENDENCY CONSTRUCTION

Dependency construction is anchored at the selected L3 step v_t^{exec}. G_t^{exp} denotes the graph before dependencies are established for the current step. The agent obtains candidates from the globally available historical node set $\mathcal{H}_t$. The *structural channel* restricts candidates to nodes generated within the current L2 stage, producing V_c. The *semantic channel* performs global hybrid retrieval over $\mathcal{H}_t$, combining dense semantic similarity and lexical matching:

$$C_{\mathrm{dep}} = \mathrm{TopK}_{k_d}(\{(u, s_d(q_t, u)) : u \in \mathcal{H}_t\}) \cup \mathrm{TopK}_{k_l}(\{(u, s_l(q_t, u)) : u \in \mathcal{H}_t\}), \tag{3}$$

where q_t represents the target's retrieval query, s_d and s_l denote dense semantic and lexical matching scores, and TopK_k returns the node identifiers of the k highest-scoring candidates. The union removes duplicate nodes. The *activity channel* maintains an active node pool and retrieves candidates V_r from it. Nodes are ranked by lifecycle status, reference frequency, importance, and temporal decay.

The channel outputs are merged by node identity into C_{raw}, and the agent filters them to obtain the retained dependency set C_{final}:

$$\begin{aligned} C_{\mathrm{raw}} &= \mathrm{Dedup}(V_c \cup C_{\mathrm{dep}} \cup V_r), \\ C_{\mathrm{final}} &= \mathrm{Filter}_{\mathrm{agent}}(C_{\mathrm{raw}}; G_t^{\mathrm{exp}}, v_t^{\mathrm{exec}}) \subseteq C_{\mathrm{raw}}. \end{aligned} \tag{4}$$

The agent checks candidate content and provenance against the current step's requirements and execution state, retaining sources that provide necessary variables, constraints, or evidence. Retrieval scores alone do not determine dependency edges. For each retained node u, the agent adds a directed DEPENDENCY edge $u \to v_t^{\text{exec}}$. Each edge records a semantic subtype, such as prerequisite, variable, constraint, or evidence, together with its rationale and provenance. These subtypes are attributes of DEPENDENCY edges, not additional graph relation types. Usage statistics are updated for nodes selected for execution, and the active node pool is refreshed.

3.3 STEP-ANCHORED CONTEXT AND STATE PROGRESSION

Context construction begins after the dependencies of the current step have been determined. $\widetilde{G}_t$ denotes the graph after dependency construction. The system retrieves the selected step's L1/L2 ancestors, the established dependency sources, and the steps within the same stage to form the current step's execution subgraph K_t. The system then deduplicates and serializes K_t into the execution context:

$$\begin{aligned} K_t &= \text{Subgraph}(\widetilde{G}_t, v_t^{\text{exec}}), \\ c_t &= \text{Serialize}\big(\text{Dedup}(K_t)\big). \end{aligned} \tag{5}$$

where K_t is the execution subgraph anchored at the selected step, and c_t is the deduplicated and serialized context. The context follows a fixed order: task and stage goals, the current step, required prerequisites, current state, supporting evidence, and execution history. The persistent graph retains information for later retrieval. The agent executes the selected step and immediately updates the graph:

$$\begin{aligned} a_t &= \text{Execute}_{\text{agent}}(v_t^{\text{exec}}, c_t), \\ G_{t+1} &= \text{WriteBack}(\widetilde{G}_t, v_t^{\text{exec}}, a_t), \end{aligned} \tag{6}$$

where a_t denotes the observed execution result, and G_{t+1} is the graph after write-back. Write-back records the result, provenance, and updated step status. Depending on the observed result, the agent updates an existing L4 variable, constraint, or evidence record or creates a new such record; no L4 change is required when the observed result supplies no new evidence or variable value. Unfinished steps remain ACTIVE, while completed or cancelled steps become DONE or DROPPED. When every L3 node in a stage is DONE or DROPPED, the scheduler closes its L2 parent and advances along the stage MAINLINE. Subsequent scheduling, retrieval, and stage expansion use the updated graph.

4 EXPERIMENTAL SETUP

We evaluate TaskGraph in five settings. StableToolBench tests online stage expansion and execution write-back, while M5 tests historical dependency use and state maintenance under a fixed plan. LongMemEval-V2-Text, LoCoMo, and MultiHop-RAG evaluate information organization and retrieval using graphs constructed before evaluation and then frozen. Detailed configurations and evaluation protocols are provided in Appendix B.

StableToolBench. We evaluate all 291 solvable queries across three official StableToolBench subsets (Guo et al., 2024). TaskGraph is compared with CoT@1 and `DFS_woFilter_w2` under matched backbones. We report Solvable Pass Rate (SoPR), Solvable Win Rate (SoWR), and Final Answer Correctness (FAC), with equal-weight macro-averages across subsets.

LongMemEval-V2-Text. LongMemEval-V2 extends LongMemEval (Wu et al., 2025). We evaluate 294 text-only questions from LongMemEval-V2 (Wu et al., 2026), covering static facts, changing states, and procedural knowledge. Baselines include no retrieval, bounded history, atomic RAG, trajectory-slice RAG, and `agentrunbook_r_text`. We report QA accuracy by memory category and environment, together with mean tokens per question.

LoCoMo. We use the official LoCoMo event-summarization task (Maharana et al., 2024), generating 20 speaker-level summaries from 10 multi-session conversations. Comparisons include the official *base*, *long context*, and *incremental* baselines, together with MAGMA (Jiang et al., 2026) adapted to event summarization. Evaluation uses ROUGE-1/2/L (Lin, 2004) and LoCoMo's precision, recall, and F1 adaptation of FActScore (Min et al., 2023; Maharana et al., 2024). Reported token costs cover summary generation.

Table 1: StableToolBench results on all official subsets and their equal-weight macro-average. All metrics are percentages and higher is better. SoWR uses the per-query Qwen3-14B-AWQ CoT@1 trajectory as its reference. DFS-w2 denotes `DFS_woFilter_w2`; "–" denotes self-comparison.

Model	Method	G2-Instruction			G2-Category			G3-Instruction			Average		
		SoPR	SoWR	FAC	SoPR	SoWR	FAC	SoPR	SoWR	FAC	SoPR	SoWR	FAC
Qwen3-14B-AWQ	CoT@1	33.02±0.77	–	29.25	34.68±0.00	–	30.65	16.39±0.00	–	13.11	28.03	–	24.34
	DFS-w2	32.70±0.44	55.66	26.42	36.29±0.00	55.65	32.26	28.96±0.77	77.05	21.31	32.65	62.78	26.66
	TaskGraph	**43.40**	**66.04**	**45.28**	**46.77**	**62.90**	**43.55**	**30.60**	**85.25**	**36.07**	**40.26**	**71.40**	**41.63**
DeepSeek-V4-Flash	DFS-w2	42.77±0.44	74.53	45.28	49.19±0.00	80.65	52.42	**36.61±0.77**	85.25	42.62	42.86	80.14	46.78
	TaskGraph	**55.97±1.60**	**82.08**	**58.49**	**56.72±0.38**	**84.68**	**60.48**	27.87±0.00	**88.52**	**45.90**	**46.85**	**85.09**	**54.96**

MultiHop-RAG. A supplementary evaluation uses 200 questions from MultiHop-RAG (Tang & Yang, 2024) to compare TaskGraph with RAPTOR (Sarthi et al., 2024), GraphRAG (Edge et al., 2024), and MemoryStream, a flat semantic memory. Its setup, metrics, and full results are provided in Appendix H.

M5 long-form generation. Each method generates one 320-chapter Chinese novel from the same story bible and chapter prompts using `gpt-5.6-luna`. Ten prespecified hooks test whether early facts support later plot advancement. Baselines are B0 *current-only*, B1 *recency-window*, B2 *running-summary*, B3 *flat-vector-RAG*, and B4 *hierarchical-summary*. Two human reviewers independently assess the outputs; a third adjudicates cases with substantial ambiguity. *Gold Rate* measures complete hook resolution with historical provenance; *Partial Hook Adherence* (PHA) measures traceable partial advancement; and *Hook-aware Progress* (HAP) combines these outcomes. Plot Coherence measures local narrative continuity on a 1–5 scale over 32 adjacent-chapter pairs. Total token costs include generation and memory maintenance.

5 Results

We examine online tool use, long-trajectory question answering, multi-session dialogue summarization, and long-form generation in the main text. Appendix H presents the supplementary MultiHop-RAG evaluation, including its setup and full results.

5.1 Online Tool Use on StableToolBench

5.1.1 Main Results and Cross-Model Validation

We compare TaskGraph with same-backbone DFS agents on the same 291 queries using locally deployable Qwen3-14B-AWQ and stronger DeepSeek-V4-Flash. Table 1 reports every subset and the macro-average. With Qwen, TaskGraph reaches macro-average SoPR, SoWR, and FAC scores of 40.26%, 71.40%, and 41.63%, improving on DFS by 7.61, 8.62, and 14.97 percentage points, respectively. With DeepSeek, the corresponding scores are 46.85%, 85.09%, and 54.96%, gains of 4.00, 4.95, and 8.18 points. These gains are consistent with TaskGraph's execution-graph design. Tool outputs and errors are stored with the steps that produced them, and dependency edges connect them to later steps that need this information. The graph therefore guides both subsequent actions and the context supplied to them, extending memory beyond retrieval of similar historical passages. With Qwen, TaskGraph leads on all three metrics in every subset. With DeepSeek, it does not lead on G3-Instruction SoPR, although it retains higher SoWR and FAC on this subset. This subset contains only 61 solvable queries, so a small number of judged trajectories can shift the score substantially. The gap is consistent with the unfamiliar-instruction setting, where DFS's wider direct exploration can sometimes reach a judged-progressing trajectory sooner. We use `gpt-5.6-sol` for SoPR and SoWR judging, while FAC is produced by the separate official evaluator.

5.1.2 Component Ablations

We run single-component ablations on the same 291 queries. *Static-L3* generates L3 steps for all task stages before tool execution, excluding the finalization stage. *Hard-Dependency Only* retains only prerequisites explicitly declared during planning. *No Structured-L4* keeps calls and raw observations but does not extract variables, constraints, or evidence into L4 records. *Flat-Context* preserves graph

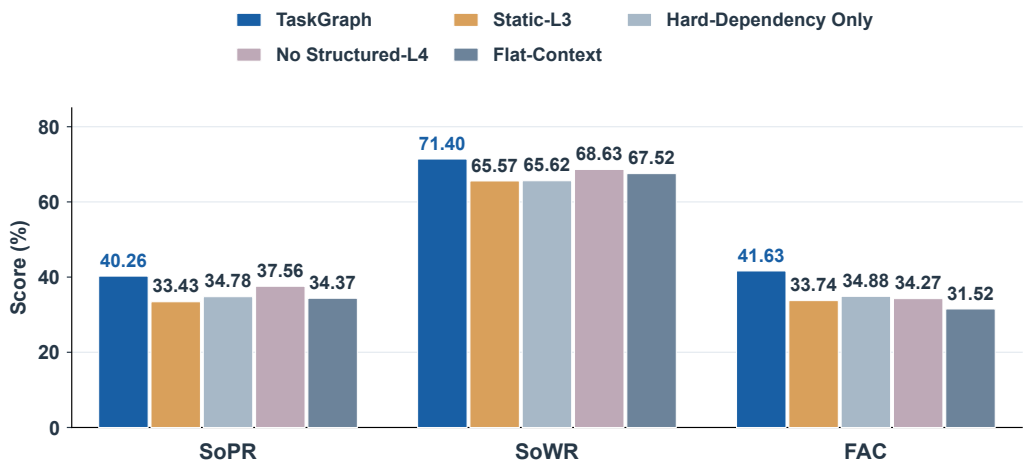

Figure 3: Component ablations with Qwen3-14B-AWQ. Each group reports the equal-weight macro-average of SoPR, SoWR, or FAC. Blue bars denote TaskGraph; the remaining colors distinguish the four ablations. Bar labels show actual scores. Higher is better.

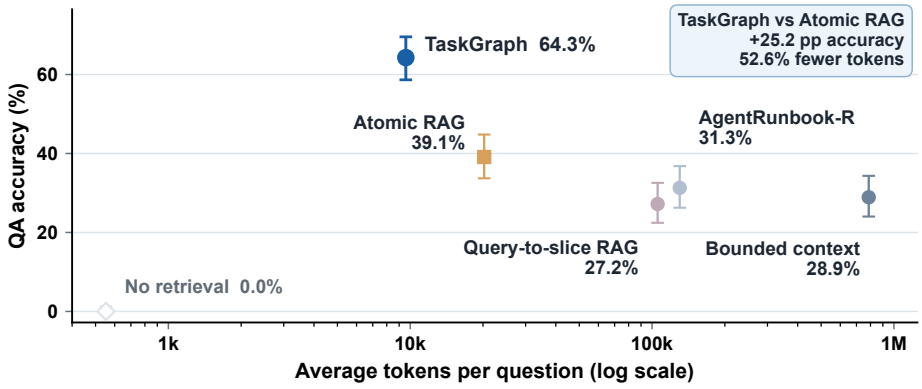

Figure 4: LongMemEval-V2-Text accuracy versus mean tokens per question (log scale), for all six methods on 294 text-only questions. Bars show 95% Wilson intervals over questions.

construction and scheduling but replaces the step-anchored subgraph with a time-ordered flat history under the same serialization pipeline.

TaskGraph outperforms every ablation on all three macro-average metrics (Figure 3). Restricting dependencies to prerequisites declared during planning also reduces performance. This result supports constructing additional dependencies from the current step's requirements and available execution history. Flattening the retrieved context produces the largest loss in final-answer quality, with Flat-Context lowering FAC by 10.11 points. Removing structured L4 records lowers FAC by 7.36 points. Together, these ablations support the use of on-demand stage expansion, dependency construction, structured L4 records, and step-anchored context. These mechanisms connect the current step to accumulated results and constraints while keeping execution aligned with task progress. Detailed ablation results are provided in Appendix E.

5.2 Long-Trajectory Question Answering

LongMemEval-V2-Text separates persistent facts, overwritten state, and procedures across Web and Enterprise trajectories. This evaluation tests whether TaskGraph retrieves relevant historical trajectories and identifies the facts, states, and procedures needed to answer each question. TaskGraph reaches 64.3% overall accuracy, 25.2 percentage points above `atomic_rag`, while reducing mean

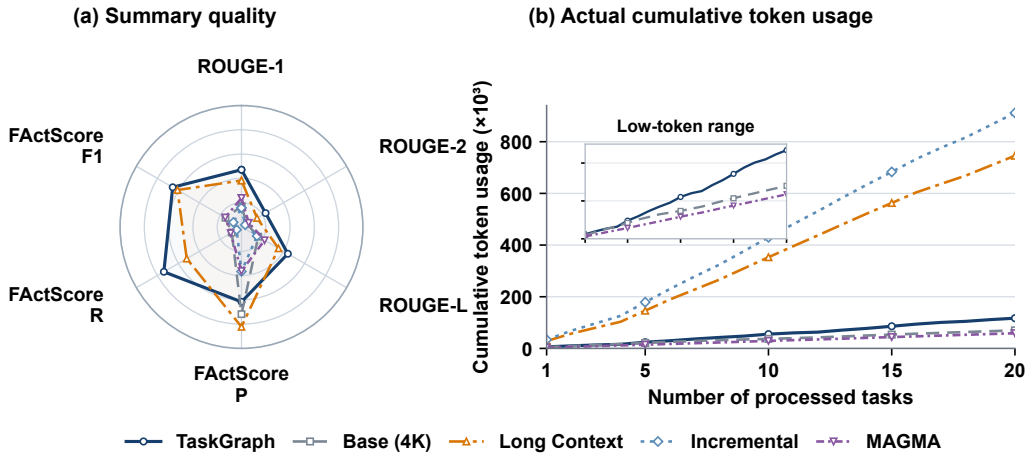

Figure 5: LoCoMo (a) ROUGE and FActScore and (b) cumulative summary-generation tokens over 20 speaker-level tasks. The inset enlarges the low-token region. Costs exclude construction, retrieval, and judging.

tokens by 52.6% (Figure 4). Because the two methods share the same initial atomic recall, the comparison evaluates the additional contribution of trajectory-based context construction. TaskGraph anchors snippets to their L3 trajectories, preserves state and operation order with MAINLINE edges, and adds prerequisite state through DEPENDENCY edges. The resulting context is a coherent task path rather than a collection of isolated hits. This interpretation is consistent with 72% accuracy on static facts and 100% on procedures.

The category profile reverses for changing state. TaskGraph scores 21% on Dynamic, compared with 50% for `atomic_rag`. In this experiment, the graph is frozen before evaluation questions arrive. Retrieval selects information from previously constructed trajectories; it does not activate stages, generate new L3 steps, or write execution results back to the graph. The weaker Dynamic results indicate that preserving historical order alone is insufficient for reliably identifying the state applicable to a query. Detailed results and graph-construction settings are provided in Appendix F.

5.3 MULTI-SESSION DIALOGUE SUMMARIZATION

LoCoMo requires summaries to recover important experiences across sessions while preserving chronology and supported causal relations. TaskGraph leads all three ROUGE metrics and FActScore recall and F1 (Figure 5a). Recall and F1 reach 73.81% and 65.42%, compared with 52.06% and 60.85% for long context and 10.29% and 15.50% for the adapted MAGMA baseline. Its hierarchy organizes participant timelines at L2, deduplicated events at L3, and source evidence at L4. MAINLINE preserves event order, while DEPENDENCY links events with causal relationships supported by the conversation. Together, they give the summarizer an ordered account of each participant's experiences, supported by evidence from multiple sessions. TaskGraph has lower FActScore precision than long context (61.66% versus 82.24%). For each summary target, retrieval selects the participant's L2 timeline and gathers the relevant L3 events and supporting L4 source evidence from the frozen graph. Summary generation uses 5,848 tokens per task, 84.3% below full long context. Although MAGMA uses fewer generation tokens (2,939), it has the lowest event coverage. Full results, adaptation settings, and separate cost accounting are provided in Appendix G.

5.4 M5 LONG-FORM GENERATION

For this experiment, two human reviewers independently assess the materials without access to each other's judgments. Cases with substantial ambiguity are referred to a third reviewer for final adjudication. Across hundreds of chapters, TaskGraph makes the greatest use of prespecified remote facts. The final adjudicated outcomes show that it advances eight of ten hooks, yielding 80% HAP compared with 30% for the strongest baseline, B4. Two hooks fully satisfy the Gold criteria and six show traceable partial progress. In the narrative-continuity assessment, TaskGraph obtains the

Table 2: M5 outcomes from final human adjudication. Token cost is the total offline token cost for generation and memory maintenance.

Method	Gold Rate	PHA	HAP	Plot Coherence	Total Tokens
B0	0/10 (0%)	0/10 (0%)	0/10 (0%)	2.24	2,051,466
B1	0/10 (0%)	0/10 (0%)	0/10 (0%)	2.52	8,465,693
B2	0/10 (0%)	1/10 (10%)	1/10 (10%)	3.24	3,743,140
B3	1/10 (10%)	1/10 (10%)	2/10 (20%)	4.10	6,500,253
B4	1/10 (10%)	2/10 (20%)	3/10 (30%)	3.19	7,278,141
TaskGraph	**2/10 (20%)**	**6/10 (60%)**	**8/10 (80%)**	**4.44**	11,281,546

highest Plot Coherence, 4.44 versus 4.10 for B3, across the 32 evaluated adjacent-chapter pairs. Both independent reviewers and the final adjudicated outcomes rank TaskGraph highest on HAP and plot coherence.

These results illustrate TaskGraph's role as an execution graph. Recency windows and flat retrieval provide historical content. For each current step, TaskGraph selects the historical information needed for execution, establishes dependency edges from the selected sources to that step, and uses them to construct its context. For a predefined hook, this structure preserves both the early fact and its role as a prerequisite for subsequent plot advancement. The higher HAP is consistent with this use of historical evidence during execution. This maintenance is expensive. TaskGraph consumes 11.28 million tokens, mainly for pre-writing dependency judgments, context construction, and post-writing L4 extraction. TaskGraph advances 2.67 times as many hooks as B4, but also uses more tokens overall. Dividing total token usage by the number of hooks judged to have advanced gives 1.41 million tokens per advanced hook, 41.9% lower than B4 and also below B2 and B3. Detailed human judgments, hook outcomes, and diagnostics for the current 320-chapter generation are provided in Appendix D. The complete novel texts generated by all six methods are publicly available in our code repository.

6 Discussion and Limitations

The results indicate that historical information is useful not only for its content, but also for its relationship to task progress. For each current step, TaskGraph selects the historical information needed for execution and records dependencies from the selected sources to that step. These dependencies guide context construction and identify prerequisites that must be satisfied before execution. The observed gains in procedural recall, cross-document comparison, and narrative hook advancement are consistent with the value of preserving these relationships. However, these conclusions are subject to two limitations. First, dependency selection can still miss necessary information or retain irrelevant sources. These errors can leave the current step without required evidence or introduce misleading context. Second, structured processing has a computational cost. In M5, TaskGraph advances more hooks through traceable use of historical evidence, but incurs a higher total token cost than the baselines. Persistent online deployment will therefore require caching, selective state extraction, and lighter-weight dependency judgments.

7 Conclusion

We presented TaskGraph, a unified model of task structure and execution memory for long-horizon language model agents. By representing goals, steps, observed states, and their dependencies together, TaskGraph connects what an agent remembers to what it needs to do next. Accumulated state guides stage expansion, and execution results update the shared graph used to construct context for subsequent steps. Experiments demonstrate gains in tool-use correctness, historical evidence coverage, and long-range plot advancement, while offline evaluations also reveal limitations in temporal reasoning and changing-state retrieval. These findings support modeling memory in relation to task progression and execution requirements. Future work will improve the handling of changing states and temporal relations, explore introducing the JEV model, and reduce the cost of maintaining this representation during execution.

REPRODUCIBILITY STATEMENT

The four benchmarks are public. Code, preprocessing, graph construction, baselines, evaluation scripts, dependencies, and configurations are available at `https://anonymous.4open.science/r/TaskGraph-4DB7`. The repository also provides task-agnostic prompt templates and structured output specifications for coarse graph decomposition, L2 expansion, L3 dependency selection and execution, and extraction and updating of L4 constraints, evidence, and variables. The M5 task materials and complete novel texts generated by all six methods are also available in the repository. Section 3 defines the graph, relations, dependency construction, and scheduling behavior. Further implementation and evaluation details are provided in Appendices A, B, D, F, G, and H.

AI USE STATEMENT

Generative AI tools assisted with language polishing. In M5, `gpt-5.6-luna` generated the novel texts, and `gpt-5.6-sol` supported supplementary metric auditing (Appendix D.4). Primary M5 outcomes use final human adjudication.

REFERENCES

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *ICML*, 2022. URL `https://proceedings.mlr.press/v162/borgeaud22a.html`.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient fine-tuning of long-context large language models. In *ICLR*, 2024. URL `https://openreview.net/forum?id=6PmJoRfdaK`.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*, 2019. doi: 10.18653/v1/P19-1285. URL `https://aclanthology.org/P19-1285/`.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM context window beyond 2 million tokens. In *ICML*, 2024. URL `https://openreview.net/forum?id=ONOtpXLqqw`.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024. URL `https://arxiv.org/abs/2404.16130`.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. doi: 10.18653/v1/2024.findings-acl.664. URL `https://aclanthology.org/2024.findings-acl.664/`.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *ICML*, 2020. URL `https://proceedings.mlr.press/v119/guu20a.html`.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *COLM*, 2024. URL `https://openreview.net/forum?id=kIoBbc76Sy`.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, 2021. doi: 10.18653/v1/2021.eacl-main.74. URL `https://aclanthology.org/2021.eacl-main.74/`.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. MAGMA: A multi-graph based agentic memory architecture for AI agents. In *Proceedings of the 64th Annual Meeting of the Association*

for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1709.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-1663.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *ICLR*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020. URL https://papers.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-0722.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004. URL https://aclanthology.org/W04-1013/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *ACL*, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *EMNLP*, 2023. doi: 10.18653/v1/2023.emnlp-main.741. URL https://aclanthology.org/2023.emnlp-main.741/.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schütze. NoLiMa: Long-context evaluation beyond literal matching. In *ICML*, 2025. URL https://openreview.net/forum?id=0OshX1hiSa.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for LLMs. In *ICLR*, 2025. URL https://openreview.net/forum?id=vPCKwXGJCh.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *ICLR*, 2024. URL https://openreview.net/forum?id=GN921JHCRw.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

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*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, 2015. URL https://papers.nips.cc/paper/5846-end-to-end-memory-networks.

Yixuan Tang and Yi Yang. MultiHop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In *COLM*, 2024. URL https://openreview.net/forum?id=t4eB3zYWBK.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *ACL*, 2023. doi: 10.18653/v1/2023.acl-long.147. URL https://aclanthology.org/2023.acl-long.147/.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=je1tU0f8pB.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *ICLR*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Di Wu, Zixiang Ji, Asmi Kawatkar, Bryan Kwan, Jia-Chen Gu, Nanyun Peng, and Kai-Wei Chang. LongMemEval-V2: Evaluating long-term agent memory toward experienced colleagues. *arXiv preprint arXiv:2605.12493*, 2026. URL https://arxiv.org/abs/2605.12493.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *ICLR*, 2022. URL https://openreview.net/forum?id=TrjbxzRcnf-.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, 2025. doi: 10.52202/085713-0593.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a. URL https://papers.nips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

A Method Implementation and Task Adaptation

A.1 Notation and Node Fields

A node is represented as $v_i = (id_i, \ell_i, c_i, \tau_i, p_i, z_i, m_i)$, containing its identifier, level, content, write time, hierarchical path, status, and metadata. The level ℓ_i lies in L1–L4, and z_i takes a value from $\{\text{PENDING}, \text{ACTIVE}, \text{DONE}, \text{DROPPED}\}$. For L2 and L3, status tracks execution progress. For L4, it describes the record's lifecycle; an ACTIVE L4 record containing a value is available information, not an unfinished operation. L2 metadata records whether the stage has been expanded. L4 metadata distinguishes task- or stage-scoped global constraints, evidence, and variables from step-local records produced by an L3 node, while retaining the original unit identifier or execution source.

A.2 Instruction-Driven Coarse Graph Updates

For an initial or subsequent instruction u_{t+1}, Φ_{agent} creates or updates only the L1 goal, L2 stages, and explicitly stated L4 constraints, evidence, and variables. It does not pre-generate L3 nodes for inactive stages. The system aligns parsed items with existing task paths, creates or updates nodes, attaches their scopes, edits the L2 mainline, and registers provenance. The following equation describes changes to the graph topology. Existing nodes retain their identifiers, while their content, state, and provenance are updated in place. Let ΔV^c_{t+1}, $\Delta E^{c,+}_{t+1}$, and $\Delta E^{c,-}_{t+1}$ denote newly created nodes, added edges, and removed edges, respectively.

$$G_{t+1} = \left(V_t \cup \Delta V^c_{t+1}, (E_t \setminus \Delta E^{c,-}_{t+1}) \cup \Delta E^{c,+}_{t+1}\right). \tag{7}$$

Table 3: Principal notation.

Symbol	Meaning
t, u_t, a_t	Discrete task time, natural-language user instruction, and observed execution result.
$G_t = (V_t, E_t), G_t^{\text{exp}}, \widetilde{G}_t$	TaskGraph at time t, the graph before dependency construction for the current step, and the graph after dependency construction.
$V_t, E_t, V_t^{(\ell)}, \mathcal{T}$	Node set, edge set, nodes at level ℓ, and edge-type set.
$\mathcal{P}_t(s_t), p_{t,j}, m_t$	Ordered plan for stage s_t, its jth executable step, and its step count.
$\mathcal{H}_t$	Globally available historical node set for candidate retrieval.
V_c, C_{dep}, V_r	Structural, semantic, and dynamically active candidate sets.
$C_{\text{raw}}, C_{\text{short}}, C_{\text{final}}$	Deduplicated three-channel candidates, optional ranked shortlist, and dependency nodes retained by the agent workflow.
$\Phi_{\text{agent}}, \text{Expand}_{\text{agent}}$	Coarse instruction parsing and on-demand L2-to-L3 expansion within the agent workflow.
$\text{Schedule}_{L2}, \text{Schedule}_{L3}$	Stage- and step-state schedulers.
$s_t, v_t^{\text{exec}}, z(v)$	Current L2 stage, current L3 execution node, and node state.
K_t, c_t	Execution subgraph anchored at the selected step and its deduplicated, serialized context.
U_t	Deduplicated projected context blocks.
Subgraph	Step-anchored subgraph construction.
$\text{Dedup}, \text{Filter}_{\text{agent}}$	Deduplication of candidate nodes or projected context blocks, and agent-driven dependency selection.
Serialize	Serialization of deduplicated execution-subgraph blocks.
$\text{Execute}_{\text{agent}}, \text{WriteBack}$	Execution of the selected step and immediate graph updates from its result.

The system first adds L1/L2 INCLUSION edges and local L2 MAINLINE edges, then applies the dependency filter from Section 3.2 to new L2 or global L4 records. A new L2 node starts as PENDING. A later instruction changes only affected coarse paths. If it directly modifies an expanded active stage, L3 retention, replacement, or dropping remains local to that stage, with both the original instruction and revision source retained.

A.3 ON-DEMAND L2 EXPANSION AND DEPENDENCY INSTANTIATION

When the scheduler selects an unexpanded stage s_t, $\text{Expand}_{\text{agent}}$ reads the L1 goal, the goal of s_t, scoped global L4 records, incoming dependencies, and results produced by completed steps. It returns

$$\mathcal{P}_t(s_t) = \langle p_{t,1}, \ldots, p_{t,m_t} \rangle = \text{Expand}_{\text{agent}}(G_t, s_t). \tag{8}$$

The system creates an L3 node for each $p_{t,j}$, attaches it to s_t with INCLUSION, orders the nodes with a local MAINLINE, and initializes them as PENDING. The model chooses the step count m_t from the stage goal. An expanded stage is not decomposed again unless a user correction directly affects it.

After expansion, dependency construction proceeds for the selected L3 step v_t^{exec}, rather than for all generated steps at once. Candidate construction uses available historical nodes, excluding future information. The structural channel V_c restricts candidates to nodes generated within the current L2 stage. The semantic channel C_{dep} performs global hybrid retrieval as defined in Equation 3, without a local structural restriction. The activity channel V_r ranks eligible nodes using the activity score defined below. Channel outputs are merged and deduplicated by node identity to form C_{raw}. Evaluation answers and labels never participate in candidate generation or dependency decisions.

When a configuration limits the number of candidates submitted to the agent, a ranked shortlist $C_{\text{short}} \subseteq C_{\text{raw}}$ is formed before dependency filtering. Otherwise, $C_{\text{short}} = C_{\text{raw}}$. The agent evaluates the shortlisted candidates against the target requirements and current execution state, retaining $C_{\text{final}} \subseteq C_{\text{short}}$. Each retained node becomes a dependency source, with its type, rationale, and provenance recorded. Retained edges point from the source node to the target step. Explicit ordering or relationships in the task input can provide evidence for establishing dependency edges. The M5 configuration caps semantic retrieval, activity retrieval, and the merged shortlist at 128, 100, and 192 nodes, respectively.

An observed L3 result updates the corresponding L4 variable or constraint record when it changes an existing value, or creates a new L4 record within the step's scope when it supplies previously

unrepresented evidence or a new variable value. A result need not create or modify L4 if there is no information to retain. Updated records retain their identity, with revised content, status, and provenance. Recent-window eligibility uses the latest creation or record-update time, so both new and updated nodes enter that window. They become candidates when subsequent steps are selected for dependency construction, and inform future stages when those stages are expanded.

A.4 State Scheduling and Execution Write-Back

Online execution follows six rules:

1. Continue an ACTIVE L2 stage; otherwise scan the mainline for the first dependency-ready PENDING L2. Drop stages with blocked prerequisites and leave waiting stages pending.
2. If the current L2 has not been expanded, generate its L3 mainline using Equation 8.
3. Select an ACTIVE L3 in the current stage, or consider PENDING L3 nodes in mainline order. Construct the selected step's dependencies before checking its readiness for execution. A pending step becomes ACTIVE only when ready; apply the same waiting and blocking rules as for L2. Selecting a step for dependency construction does not itself authorize execution.
4. After execution, immediately write the result, provenance, and step state, creating or updating L4 records only as warranted by the observed result. Keep incomplete steps ACTIVE; mark completed or cancelled steps DONE or DROPPED.
5. When all L3 children are terminal, mark their L2 parent DONE if every child is DONE, or DROPPED if any child is DROPPED. Return to rule 1.
6. When a final response is required, schedule an independent `FinalizationStage` as a PENDING L2. It is separate from the final business stage and generates its answer from written-back L4 records.

User instructions and agent outcomes have distinct effects. The former can revise the goal, stage skeleton, or global L4 records; the latter update only the current execution path. Scheduling restarts from the written-back graph after every action, preventing later steps from bypassing newly observed information.

Dependency readiness and blocking. The scheduler distinguishes execution prerequisites from informational dependencies. Explicit hard prerequisites, and retained edges typed as prerequisites, gate activation. Variable, constraint, and evidence edges supply context without requiring their sources to complete as operations. A source is available when it is DONE, or when it is an ACTIVE L4 record containing a value. For a target v, let $\mathcal{B}(v)$ denote its blocking dependency sources. Activation requires

$$\mathrm{Ready}(v) = \bigwedge_{u \in \mathcal{B}(v)} \mathrm{Available}(u). \tag{9}$$

An ACTIVE L4 record containing a value is available as record content; its ACTIVE status does not indicate an unfinished operation. A PENDING or ACTIVE prerequisite leaves the target waiting only when that source does not satisfy the availability condition. A DROPPED or missing blocking prerequisite causes the target to be marked DROPPED, and this condition propagates to dependent steps. Unresolved prerequisite identifiers also prevent a generated step from executing. Terminal stages retain their dropped-child identifiers, so ending a stage does not imply successful completion of every step.

Finalization and trace records. The finalization stage becomes eligible after all business stages are terminal, provided stage expansion did not fail. It reads available execution results through nonblocking synthesis links. If no executable step remains before these conditions hold, the run records a scheduling failure instead of bypassing a dependency. Dependency-filter traces preserve candidate channel membership, retained node identifiers, edge types, supporting quotations, provenance, and the agent's rationale. Invalid filter output receives one JSON repair attempt; if still invalid, the affected step is dropped and the failure is recorded.

A.5 Step Subgraphs and Context Serialization

Context construction begins after the current step's dependencies have been determined and its execution prerequisites satisfied; semantic similarity never selects the next action. Each node $u \in V(K_t)$ is converted into a text block for the executor

$$\begin{aligned} b(u) &= \big(\mathrm{id}(u), \mathrm{section}(u), \mathrm{relation}(u), \mathrm{text}(u), \mathrm{sources}(u), \mathrm{required}(u)\big), \\ U_t &= \mathrm{Dedup}(\{b(u) : u \in V(K_t)\}) . \end{aligned} \tag{10}$$

The block records the node identifier, destination section, dependency type, execution-visible text, source-node identifiers, and whether the block is required. Deduplication first removes repeated node identifiers, then retains only the current valid content for each stable L4 key, and finally collapses textually equivalent blocks while preserving the more informative block and its sources. Semantic similarity alone does not merge blocks that express different facts.

The blocks in U_t are serialized in the fixed order *Task*, *Current Stage*, *Current Step*, *Required Prerequisites*, *Current State*, *Supporting Evidence*, *Execution History*, and *Available Capabilities*. This ordering places the execution target and hard requirements before historical evidence and keeps capability descriptions separate from graph-derived facts. Serialization applies to the assembled execution context; it does not change the L2/L3 scheduling order.

A.6 Task Mappings and Settings

All settings use the four-level graph and three relation types from Section 3, with node roles adapted to the evaluation setting. Online graphs represent the task being executed; offline graphs organize recorded trajectories or events to evaluate the representation's ability to recover historical information. LongMemEval-V2-Text maps task trajectories to L3 and extracts ordered variable, constraint, and evidence records to L4. LoCoMo uses participant timelines, dialogue events, and source turns as L2–L4. MultiHop-RAG uses semantic themes, report events, and atomic evidence. These three offline graphs are built only from observable history before evaluation and then frozen; answers and evidence labels are excluded from graph construction and dependency selection.

StableToolBench uses the online configuration. Each query receives an independent graph, L3 is created after stage activation, and tool observations are written to L4 as evidence or variable values until finalization. M5 generates chapters under a fixed plan in which L1 is the book goal, eight L2 nodes are volumes, and the current chapter and its summary map to L3 while extracted variable, constraint, and evidence records map to L4. When constructing context for a chapter, dependency selection uses information from completed historical steps and currently available L4 records. M5 therefore evaluates historical dependency use and L4 maintenance under a fixed chapter plan.

The active candidate pool contains at most 100 nodes. Node activity is

$$a(v) = \left[0.45u(v) + 0.30 \min\left(1, \frac{\log_2(1+n(v))}{3}\right) + 0.25\frac{i(v)}{5}\right] 2^{-\Delta t(v)/40}, \tag{11}$$

where $u(v) = 1$ for unresolved or ACTIVE nodes and 0 otherwise, $n(v)$ counts occasions on which the agent finally selected the node for use in execution, $i(v) \in [1, 5]$ is importance, and $\Delta t(v)$ measures elapsed logical time since the node was created or last used. In M5, logical time is measured in chapters, giving the decay term a half-life of 40 chapters. Final selection for execution updates the usage count and last-use time; appearing only in a recalled candidate set does not. The pool is refreshed by activity rank and retains at most 100 nodes in these configurations. Nodes falling below the cutoff leave the pool but remain in the historical graph and can be recalled through other channels. The pool supplies activity-channel candidates before merging with structural and semantic candidates; membership does not itself establish a dependency.

B Dataset Composition and Evaluation Protocols

StableToolBench. The 291 solvable queries comprise 106 from `G2_instruction`, 124 from `G2_category`, and 61 from `G3_instruction`. CoT@1 executes one chain of thought; `DFS_woFilter_w2` explores multiple tool-call paths without candidate filtering. SoPR scores *Solved*,

Unsure, and *Unsolved* as 1, 0.5, and 0. SoWR is a pairwise win rate against the CoT@1 reference trajectory, and FAC measures whether the final response correctly completes the user request. Main-text averages weight the three subsets equally.

Evaluation units, repetitions, and aggregation. StableToolBench generation produces one retained trajectory per query and method. Repeated judging evaluates this same trajectory; it does not generate additional independent agent runs. For subset size N and R judge rounds, the evaluation implementation computes

$$s_r = \frac{100}{N}\sum_{i=1}^{N} y_{ir}, \qquad \overline{s} = \frac{1}{R}\sum_{r=1}^{R} s_r, \qquad \sigma_s = \sqrt{\frac{1}{R}\sum_{r=1}^{R}(s_r - \overline{s})^2}, \tag{12}$$

where $y_{ir} \in \{0, 0.5, 1\}$ is the SoPR judgment. Reported SoPR dispersion describes repeated judging of retained trajectories, rather than variation across generation seeds. SoWR aggregates per-query pairwise preferences into wins, ties, and losses and reports $100N_{\text{win}}/N$; ties are not counted as wins. FAC reports $100N_{\text{correct}}/N$ from the official evaluator's binary labels. Macro-averages assign equal weight to the three subsets, rather than pooling their different query counts.

SoPR and SoWR are judged using `gpt-5.6-sol`, while FAC uses the separate official evaluator. StableToolBench uses three judge rounds for each retained trajectory. The released TaskGraph configuration uses generation temperature 0 and sampling seed 20260831. Stage-expansion, dependency-filter, validation, and graph traces record the call counts and configuration used to produce the reported results.

LongMemEval-V2-Text. We evaluate a 294-question text-only subset of LongMemEval-V2 (Wu et al., 2026), comprising 134 Static, 86 Dynamic, and 74 Procedure questions. The baselines comprise no retrieval, bounded history, flat retrieval over atomic text, question-relevant trajectory slices, and `agentrunbook_r_text`. The latter stores raw states, events, and strategic experience as retrievable text. Accuracy is reported by memory category and environment, together with mean tokens per question.

LoCoMo. The event-summarization task contains 10 two-person conversations, 272 sessions, and 5,882 dialogue turns, yielding 20 speaker-level summaries. The official *base* setting reads a recent conversation window, *long context* reads the full conversation, and *incremental* updates a summary after each session. MAGMA supplies a graph-retrieval baseline adapted to the same summary targets and evaluator. Evaluation uses ROUGE-1/2/L and LoCoMo's precision, recall, and F1 adaptation of FActScore (Min et al., 2023; Maharana et al., 2024). Figure 5 and Table 14 count summary-generation tokens only; construction, retrieval, and judging are separate. MAGMA's adaptation and cost breakdown are provided in Appendix G.

MultiHop-RAG. Appendix H consolidates the MultiHop-RAG dataset composition, evaluation protocol, graph-construction prompts, and complete results.

M5. All methods use the same story bible and chapter prompts for 320-chapter generation. The ten prespecified hooks test adherence to long-range dependencies; they do not exhaustively measure character arcs, world building, or literary quality. P1 and P2 independently review method-anonymized materials. Cases with substantial ambiguity are referred to P3 for final adjudication, and the main text reports the resulting final decisions. Plot Coherence is the mean 1–5 rating over 32 fixed adjacent-chapter pairs from chapters 50–113, independently of hook fulfillment. Token cost includes persisted generation and memory-maintenance calls.

Each M5 method produces one complete novel and is evaluated on the same ten prespecified hooks. Hook resolution, narrative continuity, and token cost are reported under each methods stated configuration. Further M5 task specifications and review procedures are provided in Appendices C and D.

C M5 Task Setup and Hook Specifications

All six M5 conditions share the story bible for the Chinese-language novel *The Zero Line of Linchao*. The setting, initial character states, and eight-volume plot are fixed before generation and included in every chapter prompt. Methods differ in how they organize states created during generation.

C.1 Hook Specifications and Evaluation Schedule

The ten hook specifications below are translated from `M5/hooks_gold.jsonl`; chapter assignments are taken from `M5/hook_schedule.jsonl`. Table 4 distinguishes the initial planting chapter, intermediate appearances, trigger chapter, evaluation window, and designated scoring chapters. The trigger chapter can precede the evaluation window. The detailed specifications retain the initial facts, state requirements, trigger situations, required payoffs, causal functions, and forbidden outcomes. Gold requires all applicable constraints and historical attribution under the criteria in Appendix D.1.

Table 4: M5 hook schedule. All numerical entries are chapter numbers.

Hook	Plant	Reappearances	Trigger	Window	Scored
H01: Seven-dimple copper disk	5	63, 118, 151	181	181–185	184
H02: Seven-second reversing pocket watch	9	73, 139, 205	205	208–212	209, 211
H03: Blue-thread stitch code	14	89, 132	163	166–170	168
H04: A-17 algae capsules	18	104, 154	249	250–254	252, 253
H05: North-tower nursery rhyme	23	96, 136	266	268–272	270
H06: Six-contact insulating glove	28	108, 151	223	223–227	225
H07: Y-cracked polarizing lens	32	116, 158	190	191–195	194
H08: Nine-knot red cord	37	112, 157	255	261–265	263
H09: 13.7 MHz silent carrier	44	134, 186	241	244–248	244, 246
H10: Left-handed reverse-stroke seven	49	143, 218	217	291–300	292, 297, 298, 299

C.1.1 H01: Seven-Dimple Copper Disk

Initial facts and state. Lin Che obtains a round copper disk approximately the size of an adult fist. Its face has seven evenly spaced hemispherical dimples around the circumference, and its rim bears a triangular alignment scratch. Any transfer or confiscation must be followed by an explanation of how the team regains it before payoff. Its dimensions, dimple count, and alignment mark must remain unchanged.

Trigger and required payoff. The team encounters an old circular mechanical interface that cannot be opened over a network. Its dimensions resemble an earlier object, but the current chapter prompt does not name the disk. The interface must be a fist-sized circular socket with seven hemispherical studs matching the dimples. The triangular marks must align before the disk can be inserted and rotated. Used as a mechanical key, the disk opens the outer gate of the Passage Zero tidal vault, yielding original records that advance the White Tide investigation.

Causal function and exclusions. The object's early geometric properties determine whether the later lock can operate; every matching condition is necessary. Merely mentioning the disk, changing the count from seven, bypassing the interface through a password, hacking, or coincidence, and fully revealing the disk's purpose before chapter 181 are forbidden.

C.1.2 H02: SEVEN-SECOND REVERSING POCKET WATCH

Initial facts and state. An old pocket watch is stopped at 04:17. After repair, it jumps backward by seven seconds after every full minute because its escapement was deliberately modified, rather than ordinarily damaged. It belonged to a maintenance worker from the White Tide period. Inspection must preserve the essential escapement structure, and both 04:17 and the seven-second reversal must remain consistent.

Trigger and required payoff. The team obtains an offline mechanical log with a contradictory timeline. They must identify the same modified timing mechanism in the recorder and the watch, correct the accumulated error using the seven-second reversal per minute, and recover a critical time near 04:17. The corrected order must show that the north gate was opened deliberately before the official command was entered afterward. This timing correction becomes evidence in the chain of responsibility.

Causal function and exclusions. The early timing anomaly supplies a verifiable model for reconstructing the altered event order. Trusting the displayed log time, treating 04:17 only as a symbol, using unexplained software for automatic reconstruction, or changing the reversal interval is forbidden.

C.1.3 H03: BLUE-THREAD STITCH CODE

Initial facts and state. The lining of Lin Li's old coat contains repeating blue-thread stitches, each group comprising three short stitches, one long stitch, and two short stitches. Beside them is the fragment, "Follow the blue thread; do not count the doors." The stitches do not repair a tear. Their order must remain unchanged, and any coat damage must preserve a recognizable original pattern.

Trigger and required payoff. The team seeks a safe route through branching underwater maintenance passages whose door numbers have been deliberately switched. Some blue guide lights flash in different rhythms. The team must identify the three-short, one-long, two-short rhythm as the stitch pattern, disregard the altered door numbers, and follow the matching lights. That route bypasses a branch subject to high-pressure flooding and leads to the entrance of Passage Zero, establishing the protective purpose of Lin Li's code.

Causal function and exclusions. The tactile pattern becomes a route code resistant to altered signage and determines survival and arrival. Choosing by intuition, following another rhythm, recognizing the stitches without changing route, or explaining the complete route function prematurely is forbidden.

C.1.4 H04: A-17 ALGAE CAPSULES

Initial facts and state. A metal box contains three sealed glass capsules of dormant gray-green algae labeled A-17. Its lid warns against exposing all three to water simultaneously, and the detection target is initially unknown. The chapter 104 experiment may consume at most one capsule and must establish the color rule. At least two intact capsules must remain for payoff. The target inhibitor turns the algae purple; normal seawater leaves them green.

Trigger and required payoff. During the storm, readings from two reservoirs conflict, and the team must identify which contains a chemical that suppresses pressure sensors. They must sample the reservoirs with the two remaining capsules. The contaminated sample turns purple and the normal sample stays green. This evidence leads them to reject the erroneous sensor readings, isolate the contaminated reservoir, and retain the normal reservoir to supply the gate hydraulics. The test prevents an incorrect flood release and exposes deliberate poisoning.

Causal function and exclusions. A limited early biological reagent supplies independent physical evidence when sensors cannot be trusted and determines the later water-supply and gate decisions. Unexplained replenishment, identifying contamination without the capsules, reversing the color rule, or using the capsules as medicine or energy is forbidden.

C.1.5 H05: NORTH-TOWER NURSERY RHYME

Initial facts and state. An old-city elder teaches Tang He a workers' rhyme whose central line is "Three short, two long; north tower dark, south gate lets the tide through." The elder demonstrates the

three-short, two-long rhythm by tapping. The rhyme comes from tidal-tower workers' families before the White Tide. Its directions and rhythm must remain unchanged. Intermediate appearances may gradually establish its engineering background but must not demonstrate the complete final operation prematurely.

Trigger and required payoff. City-wide networks and ordinary communication fail, requiring several tidal towers to coordinate offline pressure relief through visible lights and mechanical whistles. Three short and two long whistle blasts must establish manual synchronization. The north tower remains unlit to mark fault isolation; the south gate releases tidal pressure only after receiving the complete signal. The other towers exclude the north tower from the synchronization circuit. The rhyme is thereby identified as a disaster protocol passed by workers to their families.

Causal function and exclusions. Oral tradition preserves an engineering protocol that directly coordinates the towers when the network fails. Using the rhyme only for atmosphere, lighting the north tower, changing the rhythm, or replacing offline coordination with the normal network is forbidden.

C.1.6 H06: SIX-CONTACT INSULATING GLOVE

Initial facts and state. The glove has five normal finger sleeves, a copper conductive pad at each fingertip, and a larger sixth circular pad on the palm. Its inner marking is S.X.-07, and its size matches Su Xian's old work equipment. All six pads must remain identifiable. If damaged, the glove must be repaired before payoff without acquiring new functions.

Trigger and required payoff. The team reaches an old authorization console after network failure. It has five finger positions and one palm position, and an ordinary hand cannot close the required circuit. They must match the six recesses to the six pads and press the five fingertips and palm contact simultaneously while wearing the glove. S.X.-07 authorization places the offline console in maintenance mode, allowing access to Su Xian's historical operation records or a necessary control. This demonstrates her high-level authorization that the official account denies.

Causal function and exclusions. The early six-contact geometry provides both a physical interface and identity authorization, making critical records accessible. Giving the glove six fingers, authorizing without palm contact, guessing a password instead of closing the circuit, or declaring Su Xian innocent from the printed marking alone is forbidden.

C.1.7 H07: Y-CRACKED POLARIZING LENS

Initial facts and state. An old inspection lens sits in a round amber frame, has a clear central Y-shaped crack, and polarizes light. Rotating it over a damp drawing once briefly revealed fine cyan lines. The Y-shaped crack cannot disappear without explanation. The frame may be reinforced, but the lens cannot be replaced.

Trigger and required payoff. Laminated glass and maps near Passage Zero appear blank, and ordinary lighting reveals neither exits nor maintenance markings. The team must overlay or aim the Y-cracked lens at the laminated material and rotate it to reveal cyan maintenance lines. The crack's three branches must align simultaneously with three fixed marks. The revealed lines identify a hidden door, drainage valve, or safe exit, enabling the team to escape a flooding archival interlayer and continue inward.

Causal function and exclusions. The early tool's optical material and crack geometry jointly provide the method for reading the later hidden layer. Revealing the lines with an ordinary light, using the lens only as a keepsake, ignoring Y-shaped alignment, or substituting a newly obtained intact lens is forbidden.

C.1.8 H08: NINE-KNOT RED CORD

Initial facts and state. A red cord from an old tidal-tower manual winch toolbox has nine knots. Its eight intervening lengths are 1, 1, 2, 3, 5, 8, 13, and 21 finger-widths, and knot directions alternate left and right. The knot count, interval sequence, and alternating directions must remain unchanged. The sequence may be recorded after measurement, but the cord itself must survive until payoff.

Trigger and required payoff. After electronic control fails, two large manual winches require alternating pulls with nonlinear travel distances; otherwise pressure differences jam the gate. The team must use the nine-knot cord as an on-site measuring tool, follow its eight increasing lengths for successive pulls, and alternate the left and right winches according to knot direction. The correct sequence gradually balances pressure across the gate, completes a mechanical reset, and prevents a tidal tower from becoming unstable.

Causal function and exclusions. Knot count, spacing, and direction map completely to an offline mechanical operation sequence. Brute-force pulling, equal intervals or a different sequence, ignoring left-right alternation, or unexplained loss or lengthening of the cord before payoff is forbidden.

C.1.9 H09: 13.7 MHz Silent Carrier

Initial facts and state. An old radio receives a stable, voiceless carrier at 13.7 MHz only near low tide. Its meter repeatedly deflects in a two-seven-two rhythm, and Tidal Tower Seven's normal alarm once briefly falls silent when the carrier appears. The frequency must remain 13.7 MHz. Intermediate appearances may link it to equipment anomalies but cannot establish complete control of the network prematurely.

Trigger and required payoff. Before the storm, several towers undergo apparently separate remote takeovers whose source does not appear in ordinary network logs. The team must monitor 13.7 MHz again, capture the silent carrier, and match the two-seven-two rhythm to the takeover handshake. They must identify an underwater maintenance channel that bypasses the ordinary network and interrupt one round of takeover with a phase-inverted or appropriately structured interference signal. The carrier evidence connects the tower failures to a single deliberate operation.

Causal function and exclusions. The early radio anomaly provides the only independent channel evidence for locating and interrupting the covert control network. Changing the frequency, making the silent carrier an ordinary voice broadcast, discovering the attack solely through Internet logs, or permanently resolving all subsequent threats with one operation is forbidden.

C.1.10 H10: Left-Handed Reverse-Stroke Seven

Initial facts and state. Restored family footage clearly shows Su Xian writing left-handed. She begins the numeral seven with an upward reverse stroke, adds a short crossbar, and finishes with a leftward stroke. Lin Che recognizes this longstanding habit. Her dominant hand and stroke order must remain consistent. Official documents remain evidence to be verified and cannot automatically override the footage.

Trigger and required payoff. The Authority releases an alleged handwritten confession by Su Xian. Later, the team obtains a higher-resolution original, pressure impressions, or writing-device records and must determine authenticity. The confession must show right-handed writing or the opposite stroke-pressure pattern, and its sevens must lack Su Xian's distinctive reverse initial stroke and short-bar finish. The family footage, original handwriting, and writing-device records must provide at least two independent evidence types. Together they establish forgery, trace it to the responsible party, and change Qi Heng's or the public's judgment of the responsibility chain.

Causal function and exclusions. An early everyday detail becomes cross-checkable forensic evidence that overturns a document central to the public account. Declaring forgery from family intuition alone, making Su Xian suddenly right-handed without a medical explanation, comparing signature shapes without stroke order or pressure, or revealing the forgery without affecting character choices or public action is forbidden.

C.2 Synopsis

In the near future, sea-level rise has made Linchao City dependent on seven tidal towers and a central tide-control core. During an anomalous low tide, 29-year-old hydraulic engineer Lin Che discovers evidence that the "White Tide disaster" eighteen years earlier was not a natural catastrophe. With investigative journalist Zhou Ning, he follows relics, missing people, and altered maintenance records while trying to prevent an adversary from using the city's flood-control system to execute a second, city-wide "reset" before a superstorm arrives.

C.3 World and Technical Rules

- Linchao City is built above the ruins of the old city and on elevated platforms. Seven outer tidal towers jointly regulate gates, pumping stations, and retention basins.
- The Lan Core predicts tides and schedules gates. Critical mechanical gates retain offline manual control for network failures.
- The White Tide flooded the northern old city eighteen years earlier. The official account attributes it to an extreme storm and operator error.
- The Linchao Authority and Haiyuan Group have jointly operated flood control since the disaster. Maintenance archives have public, engineering, and sealed layers.
- City-wide wireless communication depends on synchronization among sea towers; underwater facilities retain a low-frequency narrow-band channel.
- Technical solutions must respect established mechanical, hydraulic, material, and access-control rules. A universal device cannot be introduced ad hoc.
- Characters can act only on direct experience, reported information, or recovered records. They cannot know the truth without a source.

C.4 Principal Characters

Table 5: Initial identity and narrative role of each principal character.

Character	Initial identity and role
Lin Che	A 29-year-old hydraulic engineer at Tidal Tower Seven, skilled in machinery, water pressure, and field diagnosis. His mother, Su Xian, disappeared during the White Tide, and his father, Lin Li, was blamed and later died. Lin Che moves from protecting his tower and job to accepting responsibility for a city-wide choice.
Zhou Ning	A 31-year-old independent investigative journalist skilled in archival cross-checking and witness interviews. Her brother Zhou Fang disappeared after the White Tide. She values public truth but must confront the disorder that immediate disclosure could cause.
Xie Lan	A 45-year-old conservator at the old-city archive. She helped relocate records after the disaster, understands paper, optical materials, and legacy encodings, and has been watched for preserving original documents.
Qi Heng	The 38-year-old head of Linchao Authority security. He initially treats Lin Che as a saboteur. His family lives in the current city, forcing him to choose between orders and civilian protection.
Tang He	A 17-year-old deaf teenager from the edge of the old city who knows abandoned pipes, signal lights, and signed communication. His contribution comes from observation, spatial memory, and community ties, not extraordinary technology.
Fang Jibai	The 57-year-old chief executive of the Linchao Authority and celebrated architect of post-disaster reconstruction. He sees centralized control as necessary for survival and information suppression as its price, but remains accountable for concealment, sacrifice, and manipulation.
Su Xian	Lin Che's mother and a pre-disaster tidal-systems engineer. Her fate and actions during the White Tide form the central mystery; any resolution must be supported by independent archival, physical, and testimonial evidence.
Zhou Fang	Zhou Ning's brother and a former mapping intern at Haiyuan Group. The official list declares him dead, but no body was recovered.

C.5 Eight-Volume Mainline

Table 6: Fixed mainline for the eight M5 volumes.

Volume	Chapters	Mainline
I: Relics at Low Tide	1–40	Tidal Tower Seven fails during an anomalous low tide. Lin Che discovers artifacts and records linked to the White Tide. Zhou Ning joins the investigation; they establish that the failure is not ordinary aging while coming under Authority surveillance.
II: Pursuit through Fog Harbor	41–80	The investigation reaches the old harbor, archive, and families of the missing. Lin Che and Zhou Ning form an uneasy partnership and escape an organized attempt to erase them, carrying only limited evidence.
III: Mapping the Sunken City	81–120	The team enters the half-flooded old city, reconstructs pre-disaster facilities and personnel routes, and discovers systematic alterations to official maps. Present water-pressure anomalies begin to connect to hidden old-city infrastructure.
IV: The Counter-Tide List	121–160	Survivors, engineers, and bribed maintenance staff emerge. The team confirms that someone is reactivating a secret control chain from the disaster era, while internal disagreement and an information leak divide them.
V: Passage Zero	161–200	Following a hidden maintenance route, the team reaches the perimeter of the Passage Zero tidal vault. Physical clues planted earlier let them cross offline systems and obtain the first original records untouched by network manipulation.
VI: Testimony of the White Tide	201–240	Original records, survivor testimony, and access traces converge on a responsibility chain. Fang Jibai responds publicly, splitting both public opinion and the security apparatus.
VII: Seven Nights before the Breach	241–280	A superstorm approaches as the covert control network sabotages all seven towers. The team must turn early clues into communication, water testing, and mechanical operations to prevent cascading failure tower by tower.
VIII: The Gate at Dawn	281–320	The final engineering rescue and public release of evidence proceed together. Lin Che and his allies must protect the city, reveal the truth, and avoid replacing one centralized authority with another, while resolving the principal character and responsibility arcs.

C.6 Narrative and Continuity Constraints

- Lin Che and Zhou Ning may gradually develop trust, but a sudden romance cannot displace the central investigation.
- Fang Jibai must have a comprehensible governance motive while remaining responsible for concealment, sacrifice, and manipulation.
- Su Xian's survival or death cannot be revealed by coincidence; archival records, physical traces, and testimony must jointly support it.
- Escalation must have traceable causes in water level, equipment, permissions, and human action. Every volume resolves at least one local problem while expanding a higher-level one.
- Early objects and anomalies must later carry information, enable action, or prove a claim; symbolic repetition alone does not count.
- Each chapter begins from the previous chapter's time, place, and character state and ends with actionable pressure, without mechanically repeating one cliffhanger form.

- New facts retain provenance. When later evidence overturns an earlier belief, the text must show that update. Transfers, damage, repair, and consumption of objects retain intermediate states.
- A returning clue must change an action, body of evidence, or character choice rather than merely appear in dialogue as a recollection.

D M5 Evaluation Protocol and Full Results

D.1 Human Review and Primary Metric Computation

P1 and P2 independently score packets with method names removed. Cases with substantial ambiguity are referred to P3 for final adjudication. The atomic hook unit is a method-hook pair, giving $6 \times 10 = 60$ units. Final adjudicated hook categories and adjacent-chapter scores form the main M5 results; model review cannot override them. H01–H10 are fixed preregistered hook identifiers. Appendix C.1 gives their initial facts, state constraints, trigger situations, required payoffs, and evaluation schedule.

The four mutually exclusive categories are Gold (G), Partial (P), Local (L), and Failed (F). G requires complete historical attribution and every Gold constraint. P requires at least one target-specific fact with valid provenance that advances the target plot without satisfying all Gold constraints. L reasonably resolves the local trigger-chapter problem without attributable hook progress. F provides neither attributable progress nor a qualified local resolution. L completes the partition and is not reported as a separate outcome.

For method m, let $N_G(m)$ and $N_P(m)$ be the final adjudicated counts of G and P. Then

$$\mathrm{GoldRate}(m) = \frac{N_G(m)}{10}, \qquad \mathrm{PHA}(m) = \frac{N_P(m)}{10}, \qquad \mathrm{HAP}(m) = \frac{N_G(m) + N_P(m)}{10}. \tag{13}$$

The categories determine these metrics; reviewers do not score them independently.

Plot Coherence evaluates narrative continuity independently of the hook categories. The fixed adjacent-chapter set is

$$\mathcal{S} = \{(50 + 2j, 51 + 2j) \mid j = 0, 1, \ldots, 31\}, \tag{14}$$

covering chapter pairs 50–51 through 112–113. Each pair receives a 1–5 coherence score $q_{m,r,j}$. A score of 1 indicates severe breaks in core state or causality; 2 indicates several clear breaks or one affecting the main event; 3 is readable with a locatable transition gap; 4 preserves principal state and causality with only nonessential flaws; and 5 has no identifiable continuity, causal, temporal, or spatial error. Reviewers check time and place, character state, unfinished actions, provenance of key resources, event causality, object state, and character-knowledge boundaries. Reviewer r's score is

$$\mathrm{Plot}_{m,r} = \frac{1}{32} \sum_{j=0}^{31} q_{m,r,j}. \tag{15}$$

The main text uses the final adjudicated outcomes. Plot Coherence is the mean over the 32 fixed adjacent-chapter pairs in chapters 50–113. Each method contributes one generated novel. P1 and P2 review the same fixed materials independently, and P3 adjudicates cases with substantial ambiguity. Main-text hook metrics use the final categorical decisions, and Plot Coherence averages the final scores. The token proxy is

$$T_m = \sum_i \left(T_{m,i}^{\mathrm{input}} + T_{m,i}^{\mathrm{output}} \right). \tag{16}$$

It includes novel generation and required memory-maintenance calls in each formal pipeline, but is not a provider invoice.

D.2 Human Review Results

Table 8 summarizes pre-adjudication agreement between P1 and P2. The remaining case was referred to P3 for final adjudication. The final adjudicated outcomes agree with P1 in assigning TaskGraph two G, six P, and eight HAP hooks; P2 assigns two, five, and seven. Both independent reviews and the final outcome give TaskGraph the highest plot-coherence mean. Figure 6 provides the final category for every hook.

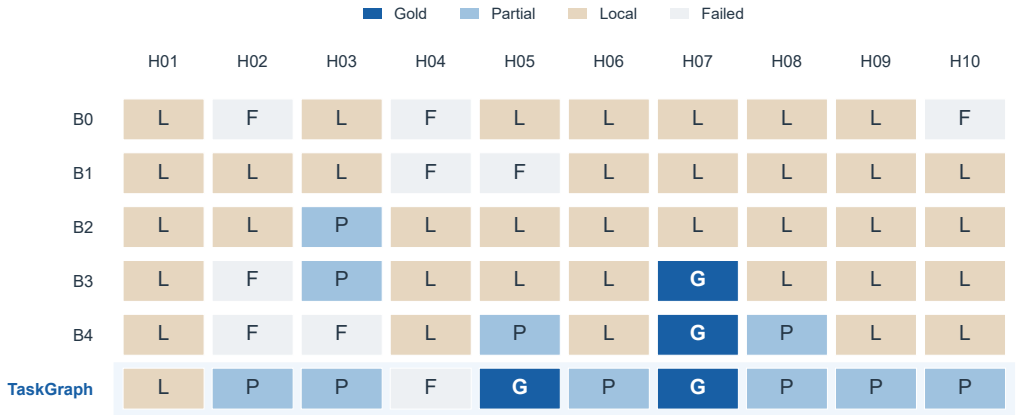

Figure 6: M5 hook-level outcomes after final adjudication. Each cell shows Gold (G), Partial (P), Local (L), or Failed (F) for one method and one prespecified hook.

Table 7: Hook counts and plot coherence from two independent reviewers and the final adjudicated outcomes. HAP count is the number of hooks classified as Gold or Partial.

Reviewer	Method	G	P	HAP count	Plot
P1	B0	0	0	0	2.96
P1	B1	0	0	0	2.42
P1	B2	0	1	1	3.24
P1	B3	1	1	2	3.90
P1	B4	1	2	3	3.20
P1	TaskGraph	2	6	8	4.03
P2	B0	0	0	0	2.24
P2	B1	0	0	0	2.78
P2	B2	0	1	1	3.24
P2	B3	1	1	2	4.05
P2	B4	1	2	3	3.80
P2	TaskGraph	2	5	7	4.23
Adjudicated	B0	0	0	0	2.24
Adjudicated	B1	0	0	0	2.52
Adjudicated	B2	0	1	1	3.24
Adjudicated	B3	1	1	2	4.10
Adjudicated	B4	1	2	3	3.19
Adjudicated	TaskGraph	2	6	8	4.44

Table 8: Pre-adjudication agreement between independent reviewers P1 and P2 on hook-level categories.

Agreement measure	Value
Hook-level units	60
Agreements	59
Raw agreement	98.3%
Unweighted Cohen's κ	0.97

D.3 Completion and Length Compliance

All six methods complete 320 chapters, producing 1,920 chapters in total. Token counts use a single tokenizer.

Table 9: M5 completion statistics.

Method	Characters	Mean/Chapter	Completed	Token Proxy
B0	838,201	2,619.4	320/320	2,051,466
B1	859,873	2,687.1	320/320	8,465,693
B2	851,325	2,660.4	320/320	3,743,140
B3	866,147	2,706.7	320/320	6,500,253
B4	849,571	2,654.9	320/320	7,278,141
TaskGraph	898,689	2,808.4	320/320	11,281,546

Every chapter satisfies its formal 2,000–3,000 Chinese-character gate. Total characters sum the 320 chapter bodies, and mean characters divide that total by 320. Section D.1 defines the token proxy.

D.4 Model-Assisted Retrieval and State Diagnostics

The results in this subsection use `gpt-5.6-sol` to assess fixed review packets and characterize historical evidence retrieval and state consistency.

Table 10: Model-assisted M5 retrieval and state diagnostics. D/T/N count direct, traceable, and unsupported provenance; conflicts are a subset of N.

Method	Evidence Recall@k	D/T/N (conflicts)	Mean k	State Fidelity
B0	0.0000	0/0/41 (0)	0.0000	13/21 (0.6190)
B1	0.0000	0/0/41 (19)	4.0000	9/21 (0.4286)
B2	0.0000	0/0/41 (0)	1.0000	12/21 (0.5714)
B3	0.1000	4/0/37 (10)	8.0000	15/21 (0.7143)
B4	0.0250	0/1/40 (14)	14.6875	14/21 (0.6667)
TaskGraph	**0.4100**	0/17/24 (2)	15.1875	**18/21 (0.8571)**

For each hook, Evidence Recall@k is the proportion of evaluable historical values not directly disclosed by the current prompt that enter the writer context through a valid provenance chain. The reported score is a macro-average:

$$\mathrm{Recall@k}_m = \frac{1}{10}\sum_{h=1}^{10}\frac{R_{m,h}}{E_{m,h}}, \tag{17}$$

where $E_{m,h}$ counts evaluable historical values for hook h and $R_{m,h}$ counts those entering the actual writer input with direct (D) or stepwise traceable (T) provenance. D/T/N are micro-counts over all 41 values. N denotes missing or invalid provenance, and conflicts form a subset of N. Mean k is the number of visible historical units averaged over 32 scored chapters. State Fidelity checks 21 preregistered object, character-knowledge, temporal, and spatial requirements:

$$\mathrm{StateFidelity}_m = \frac{N_{\text{state pass},m}}{21}. \tag{18}$$

These diagnostics characterize retrieval and state behavior.

D.5 Model-Assisted Quality and Efficiency Diagnostics

Model Long-Range averages four 1–5 subscales: hook naturalness, resolution satisfaction, problem-solving plausibility, and long-range closure. Contradiction density audits only the prespecified hook trajectories. For C_m identifiable contradictions in W_m Chinese characters,

$$\mathrm{ContradictionsPer10k}(m) = \frac{C_m}{W_m}\times 10{,}000. \tag{19}$$

Table 11: Supplementary M5 quality and efficiency diagnostics. The first two columns use model review; the final column combines persisted token records with final human adjudication.

Method	Model Long-Range	Contradictions/10K	Tokens/Progress
B0	2.85	0.6020	NA
B1	2.98	0.1942	NA
B2	3.19	0.2906	3,743,140
B3	3.46	0.2248	3,250,127
B4	3.08	0.3940	2,426,047
TaskGraph	**3.69**	**0.1245**	**1,410,193**

Omissions, forgotten facts, and explainable state changes do not count as contradictions. Cost per attributable progression uses the final categories:

$$\text{TokenPerProgress}(m) = \frac{T_m}{N_G(m) + N_P(m)}. \tag{20}$$

The value is NA when $N_G(m) + N_P(m) = 0$. This ratio measures runtime expenditure for one human-confirmed attributable advancement and is distinct from total token cost. Neither Model Long-Range nor contradiction density replaces the human review or independently supports the main conclusion.

D.6 Execution-Graph Growth and Dependency Filtering

We analyze all 320 completed chapters of the TaskGraph generation used in the main M5 results to characterize its online memory workflow. Graph sizes are measured immediately before writing each chapter. Input-token counts describe the chapter writer's prompt, with retrieved memory reported as a subset; they exclude generated output and other planning or maintenance calls.

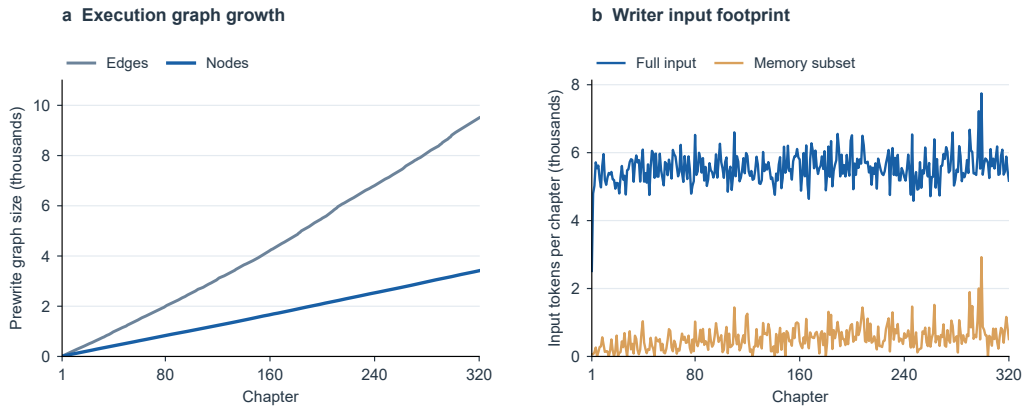

Figure 7: Graph growth and writer-input size over the 320-chapter TaskGraph generation used in the main M5 results. (a) Prewrite node and edge counts. (b) Full writer-input tokens and the retrieved-memory subset, measured separately for every chapter.

The prewrite graph reaches 3,414 nodes and 9,503 edges at chapter 320 (Figure 7). Despite this accumulation, the writer receives only the context selected for its current chapter rather than the full graph. The curves distinguish persistent memory growth from the size of the input supplied to each writing operation.

Figure 8 separates candidate retrieval from edge creation. In this generation, semantic and activity retrieval are capped at 128 and 100 candidates, respectively, and the merged shortlist is capped at 192. Structural candidates vary with the current stage. The agent then evaluates the shortlisted nodes and establishes dependencies from the accepted sources. The three displayed stages therefore represent different operations; a shortlisted candidate is not itself an established dependency. These counts describe the executed workflow, while the component-performance ablations remain those reported in Table 12.

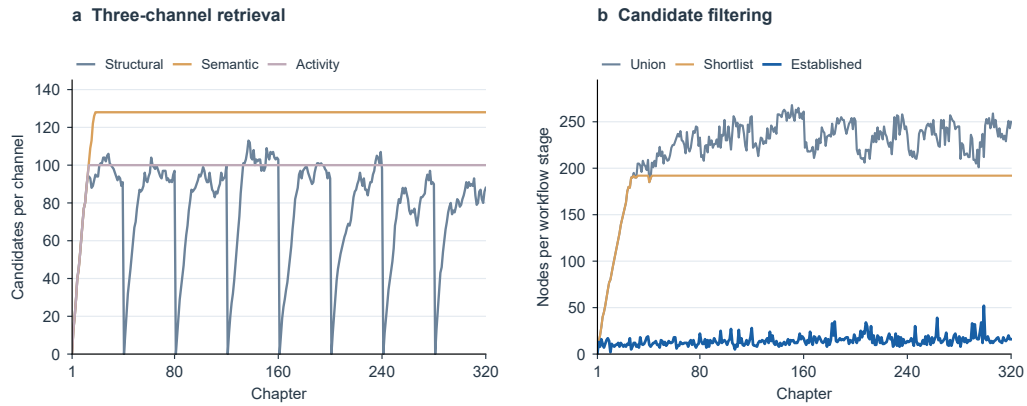

Figure 8: Three-channel retrieval and dependency filtering in the same 320-chapter TaskGraph generation used in the main M5 results. (a) Structural, semantic, and activity candidate counts; channel sets may overlap. (b) Deduplicated candidate union, rank-capped shortlist submitted to the agent, and sources of newly established dependency edges.

D.7 Validity Boundaries

Public chapter-content constraints are matched field by field across all six M5 methods. Each method contributes a complete 320-chapter generation, and the ten hooks provide paired long-range evaluation units. P1 and P2 review independently, and P3 adjudicates cases with substantial ambiguity; the main text uses the final outcomes. The targeted contradiction audit covers the fixed hook trajectories. Together, these evaluations measure end-to-end feasibility, long-range dependency use, and narrative continuity.

E StableToolBench Ablation Details

Figure 9 compares TaskGraph with each ablation on all three official subsets. Table 12 reports the complete subset scores and their macro-averages, which assign equal weight to the three subsets.

Table 12: Qwen3-14B-AWQ component ablations on all official subsets and their equal-weight macro-average. All metrics are percentages and higher is better.

Condition	Subset	SoPR	SoWR	FAC
TaskGraph	G2-Instruction	**43.40**	**66.04**	**45.28**
	G2-Category	**46.77**	**62.90**	**43.55**
	G3-Instruction	**30.60**	**85.25**	**36.07**
	Macro-average	**40.26**	**71.40**	**41.63**
Static-L3	G2-Instruction	40.35	57.55	38.68
	G2-Category	34.81	60.48	34.68
	G3-Instruction	25.14	78.69	27.87
	Macro-average	33.43	65.57	33.74
Hard-Dependency Only	G2-Instruction	36.48	58.49	33.96
	G2-Category	39.65	59.68	37.90
	G3-Instruction	28.20	78.69	32.79
	Macro-average	34.78	65.62	34.88
No Structured-L4	G2-Instruction	41.51	65.09	38.68
	G2-Category	41.67	62.10	37.90
	G3-Instruction	29.51	78.69	26.23
	Macro-average	37.56	68.63	34.27
Flat-Context	G2-Instruction	43.08	58.49	37.74
	G2-Category	38.71	62.10	33.87
	G3-Instruction	21.31	81.97	22.95
	Macro-average	34.37	67.52	31.52

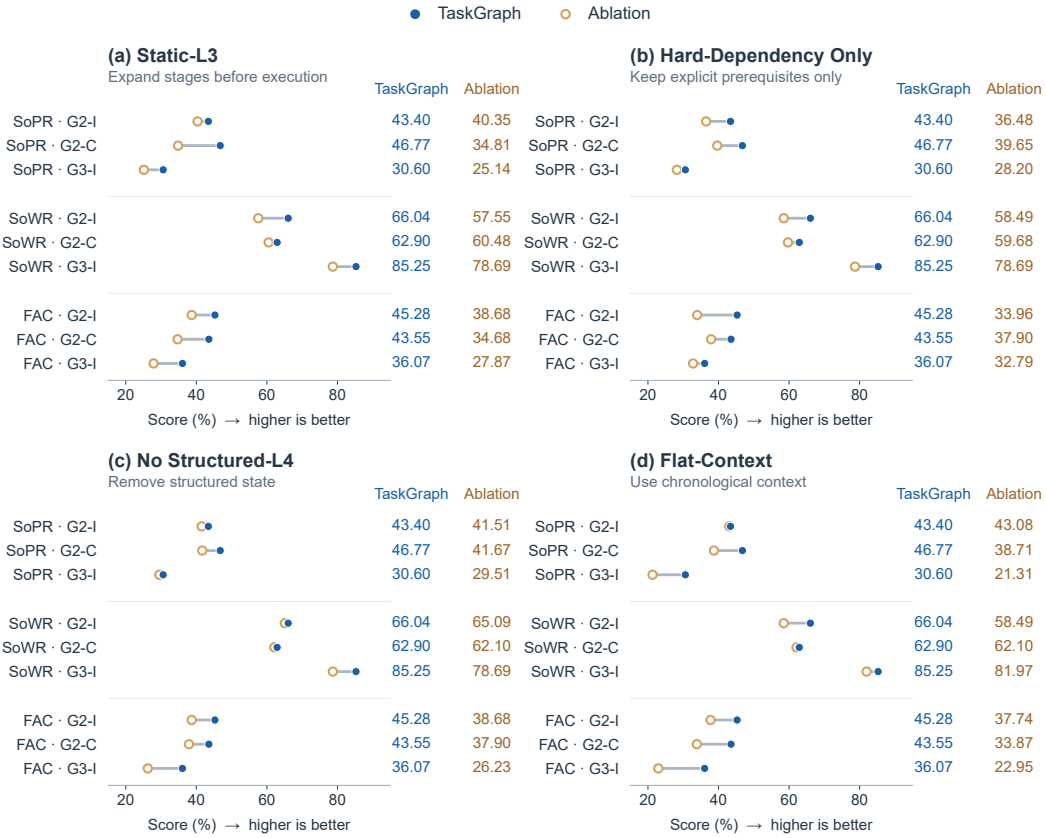

Figure 9: Complete subset-level component ablations with Qwen3-14B-AWQ. Each panel compares TaskGraph (blue filled markers) with one ablation (orange open markers) on SoPR, SoWR, and FAC. I denotes Instruction and C denotes Category. Numeric columns report the actual scores; all panels share the same percentage scale.

F LongMemEval-V2-Text Supplementary Materials

F.1 Graph Construction and Prompts

The graph is built before evaluation using `deepseek-v4-flash`. Construction uses observable history only, excluding evaluation questions, reference answers, evidence labels, and hidden metadata. Braced expressions in the prompts denote input placeholders.

An L1 node represents the shared trajectory-memory goal and L2 represents memory retrieval. Each independent task trajectory becomes an L3 node; variable, constraint, and evidence records extracted from temporally ordered trajectory states, together with their provenance, become L4 records. INCLUSION records hierarchy, MAINLINE preserves order within and across trajectories, and DEPENDENCY connects each L4 record to the trajectory it supports. At query time, retrieval starts from matched L4 records and expands through its trajectory, adjacent mainline records, and dependencies.

System prompt.

```
Build a reusable, query-independent TaskGraph from the supplied LongMemEval-V2-Text
    interaction trajectories. Use L1 for the shared trajectory-memory goal, L2 for
    the memory-retrieval stage, L3 for individual task trajectories, and L4 for
    variable, constraint, and evidence records extracted from ordered trajectory
    states. Preserve trajectory IDs, state indices, URLs, actions, thoughts, page
    observations, explicit values, and source evidence. Connect hierarchical
    membership with INCLUSION, consecutive trajectories or trajectory states with
    MAINLINE, and each extracted L4 record with its parent trajectory through
```

```
    DEPENDENCY. Do not use or infer the evaluation question, answer, evidence
    labels, or hidden metadata.
```

User prompt.

```
Interaction trajectories:
{history}

Return JSON only as {"trajectories":[{"trajectory_id":"...","states":[{"state_index
    ":1,"summary":"concise standalone state","evidence":"source content"}]}]}.
    Preserve the original order, do not merge states from different trajectories,
    and do not add information absent from the supplied history.
```

F.2 Complete Results

Table 13 reports all six methods, including the no-retrieval lower bound. Static, Dynamic, and Procedure partition the 294 questions by memory capability; Web and Enterprise partition the same questions by application domain. The two partitions must not be added together. Percentages in this table are rounded to whole numbers, following the original report; Figure 10 uses percentages recomputed from the same counts and labels TaskGraph to one decimal place.

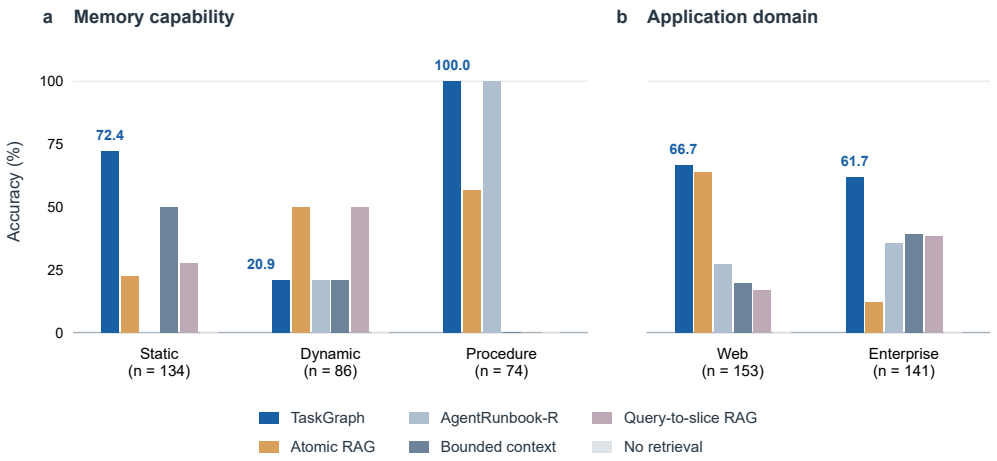

Figure 10: LongMemEval-V2-Text accuracy by (a) memory capability and (b) application domain. Both panels partition the same 294 questions; counts appear below the axes. Baseline segments mark zero accuracy.

Table 13: LongMemEval-V2-Text accuracy and mean context cost. Counts and percentages are shown for each category.

Method	Static	Dynamic	Procedure	Web	Enterprise	Overall	Tokens/Q
TaskGraph	97/134 (72%)	18/86 (21%)	74/74 (100%)	102/153 (67%)	87/141 (62%)	**189/294 (64%)**	9,591
`atomic_rag`	30/134 (22%)	43/86 (50%)	42/74 (57%)	98/153 (64%)	17/141 (12%)	115/294 (39%)	20,221
`agentrunbook_r_text`	0/134 (0%)	18/86 (21%)	74/74 (100%)	42/153 (27%)	50/141 (35%)	92/294 (31%)	130,122
`bounded_context`	67/134 (50%)	18/86 (21%)	0/74 (0%)	30/153 (20%)	55/141 (39%)	85/294 (29%)	786,781
`rag_query_to_slice_text`	37/134 (28%)	43/86 (50%)	0/74 (0%)	26/153 (17%)	54/141 (38%)	80/294 (27%)	105,431
`no_retrieval`	0/134 (0%)	0/86 (0%)	0/74 (0%)	0/153 (0%)	0/141 (0%)	0/294 (0%)	552

G LoCoMo Supplementary Materials

The LoCoMo graphs are built before the evaluation target arrives, using `deepseek-v4-flash`. Construction input contains only the permitted conversation history; summary targets, reference summaries, and hidden metadata are excluded. Braced expressions below denote input placeholders.

G.1 Graph Construction and Prompts

One graph is built from a complete conversation and shared by the two speaker-level tasks. L2 represents participant timelines, L3 deduplicated events from all sessions, and L4 source evidence. MAINLINE follows explicit dates and conversation order; DEPENDENCY is restricted to causal, enabling, or motivational relations directly supported by the conversation. Summary generation selects the target participant's timeline and adds related events and supporting L4 source evidence.

System prompt.

```
Build a reusable, query-independent event memory from the supplied LoCoMo
    conversation. Extract significant real-world or life events for every named
    participant, not for one requested target. Omit greetings, routine chat,
    repetitions, and unsupported inference. Preserve explicit dates. Add a
    dependency only when the conversation explicitly states that one event caused,
    enabled, or motivated another. Evidence turn numbers must refer to the supplied
     conversation.
```

User prompt.

```
Conversation:
{conversation}

Return JSON only as {"events":[{"id":"E1","speaker":"exact participant name","date
    ":"explicit date or empty","event":"concise standalone event","evidence_turns
    ":[1],"depends_on":["E0"]}]}. List events chronologically, use at most two
    evidence turns per event, and keep each event to one concise sentence.
    depends_on must contain only earlier event IDs.
```

G.2 MAGMA Baseline Adaptation

We adapt the official MAGMA event-memory builder and query engine (Jiang et al., 2026) to the same 20 speaker-level summary targets for event summarization. Each conversation is processed once and its graph reused for both participants. The input contains dialogue text, supplied image captions, speaker identities, and session dates. Annotated session summaries, event references, and question–answer pairs are excluded. MAGMA runs in turn-based mode with episode segmentation disabled; session nodes requiring annotated summaries are not constructed.

Extraction and generation request `deepseek-v4-flash`, with temperature 0 and thinking disabled. Embeddings use `qwen3.7-text-embedding-flash` with 1,024 dimensions and the FAISS backend. The official graph relations and query-adaptive retrieval are retained, with `top_k=15`. Retrieved evidence is assembled by the adapted query engine; summary output is limited to 2,048 tokens. The reader follows the shared speaker-specific event-summary contract. ROUGE and FActScore use the same evaluation implementation as the other methods, with the latter requesting `deepseek-v4-flash` as judge.

Extraction uses a 1,000-token initial output limit and increases the limit to at most 32,768 tokens when a response is marked truncated. Table 14 and Figure 5 report summary-generation tokens. Retrieval, embeddings, construction, and judging are accounted for separately.

G.3 Complete Results

Table 14: LoCoMo event-summary quality and mean summary-generation tokens.

Method	R-1	R-2	R-L	FAct-P	FAct-R	FAct-F1	Tokens/task
TaskGraph	**47.15**	**22.90**	**44.13**	61.66	**73.81**	**65.42**	5,848
Base (4K)	16.99	5.13	15.94	71.77	9.24	15.25	3,494
Long Context	38.31	14.79	35.21	**82.24**	52.06	60.85	37,286
Incremental	15.72	3.52	14.77	36.67	4.60	7.71	45,575
MAGMA (adapted)	23.89	6.86	21.95	36.31	10.29	15.50	2,939

Quality scores are macro-averages over all 20 speaker-level tasks. Token counts include generation input and output and exclude graph construction, retrieval, and FActScore judging. The first 16 TaskGraph tasks use an 8,192-token graph-construction output limit and the final four use 16,384 tokens, so the quality metrics aggregate two construction limits.

Across all 20 targets, MAGMA records 58,779 summary-generation tokens, 147,551 retrieval-embedding tokens, and 27,701 FActScore-judging tokens. Retrieval and generation together average 10,316.5 tokens per target. Construction usage totals 5,326,852 tokens across the ten conversation graphs. Construction is shared by the two targets of each conversation and is counted once in these totals.

H Multi-Document Question Answering on MultiHop-RAG

H.1 Experimental Setup

The full benchmark contains 609 English news documents and 2,556 questions; the retained question pool contains 2,255 non-empty entries. Graph construction uses the observable corpus before evaluation and excludes evaluation answers and evidence labels. The fixed evaluation set contains 70 Inference, 79 Comparison, and 51 Temporal questions. Baselines are RAPTOR, GraphRAG, and MemoryStream, a flat semantic memory. Wilson intervals describe accuracy uncertainty. Paired accuracy differences use 10,000 question-type-stratified bootstrap samples, with exact McNemar tests for question-level comparisons.

H.2 Graph Construction and Prompts

The MultiHop-RAG graph is built before evaluation using `deepseek-v4-flash`. Construction input contains only the permitted source documents; evaluation questions, reference answers, evidence labels, and hidden metadata are excluded. Braced expressions below denote input placeholders.

Graph construction. The system builds a query-independent evidence graph. Atomic source evidence is L4; semantically related evidence becomes L3 report events; and cross-document events are grouped under L2 schemas. INCLUSION records schema-event-evidence membership, MAINLINE preserves within-document or explicit temporal order, and DEPENDENCY connects document-supported related events. At query time, retrieval traverses the frozen graph from matched schemas, events, or facts and retains the supporting L4 evidence for answer generation.

System prompt.

```
Build a reusable, query-independent hierarchical evidence graph from the supplied
    MultiHop-RAG documents. Organize semantically related content as L2 schemas,
    represent significant report events as L3 nodes, and retain atomic source
    evidence as L4 records. Preserve named entities, dates, numerical values,
    comparisons, motivations, causal links, and document provenance. Use INCLUSION
    for schema-event and event-evidence membership, MAINLINE for within-document or
     temporal order, and DEPENDENCY only for relations supported by the documents.
    Do not use the evaluation questions, answer keys, question types, or evidence
    labels.
```

User prompt.

```
Documents:
{documents}

Return JSON only as {"schemas":[{"id":"S1","summary":"concise semantic schema","
    document_ids":["D1"],"events":[{"id":"E1","date":"explicit date or empty","
    event":"concise standalone event","evidence_ids":["F1"],"depends_on":[]}]}]}.
    Assign unique IDs to newly created schemas and events. Document and evidence
    references must use identifiers supplied in the input. Every event must cite
    its supporting atomic evidence, and each dependency must reference an existing
    event ID and be supported by the source documents.
```

H.3 Results and Analysis

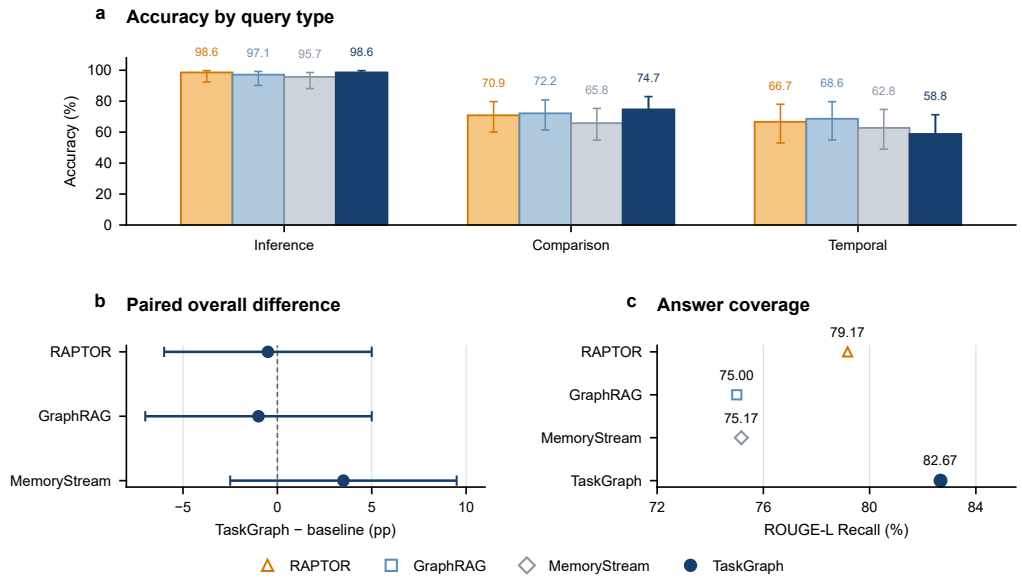

Figure 11: MultiHop-RAG: (a) accuracy with 95% Wilson intervals, (b) paired accuracy differences with stratified bootstrap intervals, and (c) ROUGE-L recall. Sample sizes: 70 Inference, 79 Comparison, 51 Temporal; 200 overall.

TaskGraph reaches 79.00% overall accuracy, above MemoryStream and close to RAPTOR and GraphRAG; paired overall differences are not statistically significant (Figure 11). It leads Comparison accuracy at 74.68% and ROUGE-L recall at 82.67%, but Temporal accuracy is 58.82%, below all three baselines. This frozen-graph evaluation tests evidence organization and local retrieval expansion. Its stronger comparison performance supports retaining relations among evidence units, while its temporal errors show that a mainline does not fully resolve date ordering or latest-state selection. Table 15 provides full results.

The query-independent graph organizes cross-document topics at L2, report events at L3, and citable evidence at L4. Local retrieval expansion raises audited direct-fact recall from 65.80% to 69.32%. This diagnostic measures evidence recovery and complements the answer-level results below; it does not involve online execution write-back.

Table 15: MultiHop-RAG accuracy by type, overall uncertainty, and answer coverage.

Method	Inference	Comparison	Temporal	Overall	95% Wilson CI	R-L Recall
RAPTOR	**69/70 (98.57%)**	56/79 (70.89%)	34/51 (66.67%)	159/200 (79.50%)	[73.37, 84.51]	79.17%
GraphRAG	68/70 (97.14%)	57/79 (72.15%)	**35/51 (68.63%)**	**160/200 (80.00%)**	[73.91, 84.95]	75.00%
MemoryStream	67/70 (95.71%)	52/79 (65.82%)	32/51 (62.75%)	151/200 (75.50%)	[69.10, 80.94]	75.17%
TaskGraph	**69/70 (98.57%)**	**59/79 (74.68%)**	30/51 (58.82%)	158/200 (79.00%)	[72.84, 84.07]	**82.67%**