H-SEAM: A Hierarchical Self-Evolving Agentic Memory System

Anonymous ACL submission

Abstract

Equipping LLM-based agents with evolving long-term memory remains a persistent challenge. Existing approaches predominantly rely on flat vectors or graphs with predetermined schemas, inevitably leading to semantic fragmentation or structural rigidity. To address this, we propose H-SEAM, a dynamic multi-level memory framework grounded in Dynamic Memory Evolution Theory. Unlike static hierarchies, H-SEAM features: 1) a multi-granular hierarchical structure that vertically integrates raw episodic fragments with semantic abstractions to bridge semantic gaps; 2) a density-driven evolutionary mechanism that dynamically distills high-level abstractions from information saturation, allowing the memory topology to adaptively grow in alignment with user interactions; and 3) a sufficiency-guided 2-hop expansion strategy that achieves an optimal accuracy-efficiency trade-off. Experiments on LOCOMO and LongMemEval demonstrate that H-SEAM consistently surpasses strong baselines (e.g., MemOS), **improving overall accuracy by 6.7%** while maintaining stable efficiency.

1 Introduction

Just as memory is the cornerstone of human cognition and identity (Craik and Jennings, 1992), it is equally central to intelligent agents, enabling them to maintain continuity in dialogue, personalization, and digital assistance. A key challenge for LLM-based agents is maintaining long-term, contextual memory across sessions.(Hu et al., 2025) Large language models (LLMs) now excel at language understanding, code generation, scientific reasoning, and interactive agents, with context windows reaching hundreds of thousands or even millions of tokens (Qian et al., 2023; Gao et al., 2023; Wang et al., 2024b). Yet long-context mechanisms still struggle with scalability, retention, and reasoning consistency (Maharana et al., 2024), making robust memory management crucial for coherent, grounded, and temporally consistent behavior in extended interactions. Recent work on agentic memory—such as Mem0 (Chhikara et al., 2025), A-Mem (Xu et al., 2025), and MemOS (Li et al., 2025)—moves toward persistent memory, often complementing retrieval-augmented generation (RAG) (Lewis et al., 2020) with structured long-term context and selective recall.

Limitations of existing approaches. Despite this progress, current agent memory frameworks still face two key limitations.

Passive accumulation and shallow consolidation. Frameworks such as MemGPT (Wu et al., 2023), LangMem (Wang et al., 2024a), and Generative Agents (Park et al., 2023b) introduce persistent storage, but largely lack active, hierarchical consolidation. Existing systems often treat memory as a flat log of interactions, relying on simplistic, recency-biased retrieval without periodically reflecting on or synthesizing past experiences. Cognitive studies show that such passive accumulation leads to fragmentation, where older but semantically relevant insights are buried under noise rather than being integrated into higher-level knowledge (Shin et al., 2017; Lesort et al., 2020).

Structural rigidity and fixed granularity. Most systems rely on flat vector stores or graph structures with predetermined schemas (e.g., fixed Entity-Relation layers). This structural rigidity prevents the memory from adapting to the evolving depth of user information.(Rasmussen et al., 2025; Xu et al., 2025; Chhikara et al., 2025; Kang et al., 2025; Li et al., 2025) Without explicit, dynamic abstraction layers, agents struggle to bridge fine-grained details with high-level context, weakening multi-step reasoning capabilities. Neuroscience models (Buzsáki, 2015; Lampinen et al., 2021) highlight that cross-level coordination is

crucial for abstraction and long-range prediction.

Motivation and intuition. Our work is fundamentally grounded in two cognitive principles: Dynamic Memory Evolution Theory and Hierarchical Association Theory. Dynamic Memory Evolution Theory posits that memory is not a static repository but a reconstructive process where traces are repeatedly reactivated, modified, and integrated over time (Dudai et al., 2015; Stoianov et al., 2022). Complementarily, Hierarchical Association Theory suggests that cognition relies on organizing information into nested levels of abstractionfrom specific episodes to general schemasto facilitate efficient retrieval and reasoning (Kounios et al., 2001; Lampinen et al., 2021). Motivated by these views, we argue that long-term agent memory should evolve hierarchically, utilizing structured associations to connect items, sequences, and contexts. Our model, H-SEAM (Hierarchical Agentic Memory), implements a density-driven, multi-level hierarchy, where new experiences are encoded as detailed traces and then autonomously abstracted into varying levels of granularity through iterative reflection.

To address these limitations, we propose H-SEAM. We argue that hierarchical memory should not be predefined or periodically summarized, but should **emerge as a function of semantic saturation and interaction dynamics.** Guided by this paradigm shift, our main contributions are:

Multi-Granular Vertical Integration. We construct a hierarchical structure that vertically integrates raw episodic fragments with semantic abstractions to bridge semantic gaps, ensuring both factual precision and conceptual depth.

Density-Driven Evolution. We introduce a mechanism that dynamically distills high-level abstractions from information saturation. This allows the memory topology to adaptively grow with user interactions rather than following a fixed schema.

Empirically Validated Navigation. We identify that a sufficiency-guided **2-hop expansion** offers the optimal trade-off between reasoning accuracy and computational overhead. Extensive experiments demonstrate that H-SEAM surpasses strong baselines (e.g., MemOS), improving overall accuracy by **6.7%** while maintaining stable efficiency.

We evaluate H-SEAM on two long-term memory benchmarks, LOCOMO (Maharana et al., 2024) and LongMemEval (Wu et al., 2024), and show that it improves accuracy and memory effectiveness while maintaining stable runtime, providing scalable and cognitively inspired long-term memory for LLM-based agents.

2 Related Work

Flat & Vector-based Memory. Pioneering works have established the foundation for persistent agent memory. Generative Agents (Park et al., 2023a) demonstrated the power of reflection over raw observation streams, yet they rely on manually scheduled reflection, incurring rigid overhead. MemGPT (Wu et al., 2023) effectively managed context limitations via an OS-style paging mechanism, and Mem0 (Chhikara et al., 2025) optimized retrieval through metadata filtering. While efficient for semantic matching, these vector-based approaches primarily treat memory as a flat sequence, facing challenges in complex multi-hop reasoning where disparate clues need structural connection to avoid semantic fragmentation.

Structured & Hierarchical Graph Memory. To capture complex dependencies, recent research has moved towards structured representations. A-Mem (Xu and et al., 2024) and HippoRAG (Gutiérrez et al., 2024) leverage associative algorithms (e.g., PageRank) on entity networks. Recent advancements like Zep (Rasmussen et al., 2025) introduce hierarchical architectures via community detection (e.g., Leiden). However, these approaches often prioritize topological connectivity over semantic coherence. For instance, Zep operates on fixed taxonomies or purely structural density, which can lead to semantic misalignment where communities form by connection density rather than conceptual relevance.

System-Level Memory Frameworks. Recognizing computational complexity, Memory OS paradigms manage the memory lifecycle. MemoryOS (Kang et al., 2025) organizes memory via fixed tiers (STM, MTM, LPM) governed by resource policies, while MemOS (Li et al., 2025) proposes schedulable MemCubes. These frameworks excel in system-level resource orchestration (i.e., *where* data is stored). H-SEAM complements this by optimizing the internal semantic topology (i.e., *how* knowledge is structured), enhancing the flexibility within these robust system

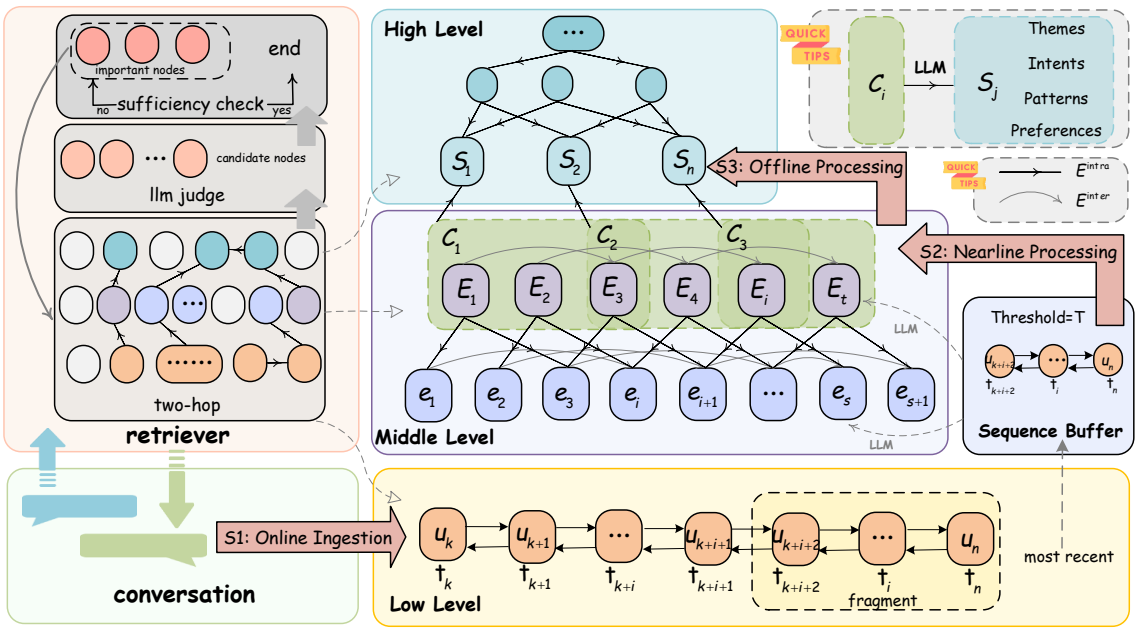

Figure 1: System overview of H-SEAM. The architecture integrates online ingestion (low-level), near-line correlation (middle-level), and offline consolidation (high-level).

architectures.

A Paradigm Shift in Hierarchy Construction. While prior works introduce hierarchy, they differ fundamentally from H-SEAM in *how hierarchy is formed*. Generative Agents rely on manual scheduling; HippoRAG constructs static graphs guided by topological centrality; and MemoryOS utilizes fixed system-level tiers. In contrast, H-SEAM introduces a **density-driven, content-adaptive evolution mechanism**. Our abstraction nodes are neither predefined nor periodically enforced but **self-organize** only when local semantic regions reach saturation. This design shifts hierarchy from a *design-time prior* to a *runtime outcome* of interaction dynamics, representing a qualitatively different paradigm where structure emerges from semantic density rather than rigid architectural constraints.

3 Method

3.1 H-SEAM Overview

Definition of Memory and Retrieval. We define memory as a structured and evolvable index I_M derived from dialogue history. Formally, given a chronological sequence $D = \{(t_1, d_1), \ldots, (t_N, d_N)\}$, the memory builder transforms D into multi-level representations I_M. A retrieval operation maps a query q to an answer a via the process $q \xrightarrow{\text{Retriever}} \mathcal{M}_q \xrightarrow{\text{Agent}} a$, where $\mathcal{M}_q \subseteq I_M$ denotes the specific subset of memory fragments extracted from the global index that is contextually relevant to q.

As illustrated in Figure 1, H-SEAM organizes memory into three synergistic functional levels:

Online Ingestion (Low-Level): Fine-grained conversational traces are continuously appended to a sequential buffer. This level maintains detailed temporal order and lexical granularity for short-term contextual retrieval.

Near-Line Correlation (Middle-Level): Once a complete session is finished or periodically, the system performs synthesis to link related sequences and identify recurrent entities. This level serves as a bridge, capturing relational regularities between raw tokens and abstract concepts.

Offline Consolidation (High-Level): During idle phases or periodically, H-SEAM aggregates semantically dense events to distill high-level abstractions. Topological restructuring occurs here to transform scattered episodic details into coherent user-specific themes.

3.2 Hierarchical Graph Memory

The memory consists of dynamic abstraction levels $L = \{0, 1, \ldots, H\}$, **visually corresponding to the Low, Middle, and High levels depicted in Figure 1.** Each level ℓ contains a node set $\mathcal{N}_\ell = \{n_{\ell,i}\}$, where each node is defined as a tuple $n_{\ell,i} = (\mathbf{v}_{\ell,i}, t_{\ell,i}, s_{\ell,i}, \text{type}_{\ell,i})$, comprising

its semantic embedding, timestamp, accumulated strength, and type (dialogue, entity, event, or context).

As illustrated in the figure's legend, edges are categorized into two types:

Intra-level edges E^{intra} (**depicted as solid black arrows**) modeling sequential or associative links within a layer; and **Inter-level edges** E^{inter} (**depicted as curved gray arrows**) representing abstraction or grounding relations between layers. The overall graph is denoted as $G = (\bigcup_{\ell} \mathcal{N}_{\ell}, E^{\text{intra}} \cup E^{\text{inter}})$.

Example. A four-level instance might contain: Level 0 (utterances), Level 1 (sessions), Level 2 (event/entity graphs), and Level 3 (context maps), connected via cross-level dependencies.

3.3 Index Builder and Evolving

3.3.1 Low-Level: Online Ingestion via Dynamic Slicing

Sequential Buffering and Linear Chaining. Recent utterances are continuously buffered and segmented into coherent dialogue slices σ via $\text{LLM}_{\text{slice}}$. As illustrated in the bottom layer of Figure 1, we strictly maintain a linear temporal graph where individual dialogue turns are instantiated as nodes and linked via directed chronological edges ($d_t \rightarrow d_{t+1}$). This chain-like topology preserves the strict temporal order and lexical fidelity of the conversation, serving as the grounded topological backbone for subsequent structured processing.

3.3.2 Middle-Level: Near-line Correlation and Graph Construction

Upon the completion of a dialogue slice, the system transitions to structured graph generation.

Entity Instantiation and Event Topology Construction. Transitioning from the linear sequence buffer to the **structured graph domain** (visualized in the **purple "Middle Level" block** of Figure 1), the system performs a topological transformation via a dual-phase pipeline.

First, **atomic entities** are disentangled from the dialogue slices using $\text{LLM}_{\text{extract}}$ (see Appendix C). As shown in the bottom row of the middle level, these entities are instantiated as **grounded nodes** (e_i), linked horizontally via semantic similarity to form the basal semantic layer.

Subsequently, **discrete events** are constructed as structured compositions of these entities and their predicates via $\text{LLM}_{\text{extract\_event}}$. These are visualized as the **higher-order event nodes** (E_i) in the figure, which aggregate the underlying entity interactions into coherent episodic units.

To capture **longitudinal dependencies**, causal relations (e.g., *precedes*, *causes*) are inferred by $\text{LLM}_{\text{relate\_event}}$. This process densifies the graph with **intra-level edges** (E^{intra}), weaving disjointed sessions into a continuous narrative fabric based on entity overlap and temporal adjacency.

LLM-Enhanced State Evolution and Conflict Resolution. To prevent redundant nodes and conflicting states during incremental graph construction, we implement a **two-stage "Recall-then-Verify" consolidation mechanism** that goes beyond simple deduplication. First, we perform a coarse retrieval to identify existing nodes with high semantic overlap (embedding similarity $> \tau_{\text{sim}}$). Instead of blindly merging based on vectors, we employ an LLM as a **Cognitive Arbiter** to compare the new fragment against these candidates. The LLM distinguishes whether the fragment represents a **state update** (increasing node strength $s_{\ell,i}$ and fusing the new information to **evolve** the entity's state) or a **distinct new concept** (creating a new node). Crucially, this agentic verification resolves **semantic ambiguity** (e.g., distinguishing "apple" the fruit from the brand) and handles **temporal conflicts** (e.g., updating "User lives in NYC" to "User moved to LA" instead of creating conflicting duplicates), ensuring the graph reflects the **true conceptual evolution** rather than just accumulating data.

Importantly, these LLM invocations operate under strictly bounded prompts and act as semantic validators rather than free-form generators.

3.3.3 High-Level: Abstraction via Density-Driven Emergence

Unlike static schemas that force information into fixed hierarchies, our abstraction layer H evolves dynamically through a **"Density-Driven Emergence"** paradigm. This mechanism is inspired by biological memory consolidation, where high-level concepts **organically emerge** only when the information density in a local region reaches saturation (i.e., local node density exceeds a capacity threshold $N_{\max}$).

Topological Clustering (Hotspot Identification): We apply K-Means not merely to group nodes, but to partition the dense graph regions into candidate **semantic hotspots** $\mathcal{C}(j)$, identi-

fying clusters where fragmented episodes share strong latent structures.

Cognitive Distillation: An LLM agent acts as a meta-reasoner to analyze the constituent episodes in $\mathcal{C}(j)$. Instead of simple summarization, it performs **pattern induction** to distill shared themes, behavioral patterns, or intent regularities (see Appendix C). It generates a synthesized insight S_j representing the **emergent high-level concept**.

Structural Instantiation: A new high-level node is instantiated with content S_j, establishing explicit dependency edges to its constituents. This creates a **vertical semantic anchor**, allowing the agent to reason over broad concepts before drilling down into details.

Entropy-Aware Graph Compression. To prevent graph entropy explosion, H-SEAM employs a Strength-Aware Greedy Merging strategy. The algorithm operates in three steps: (1) Candidate Scoring based on semantic and structural overlap; (2) Constraint Checking to ensure similarity $> \tau_\ell$ and strength preservation; and (3) Greedy Execution to merge optimal pairs, ensuring the memory topology remains sparse and efficient as interaction grows.

Theorem 1 (Bounded Stability)**.** *The merging algorithm guarantees: (1) Bounded Information Loss within* $(1-\epsilon)\mathcal{I}(G)$*; (2) Structural Stability converging to equilibrium density* ρ^**; and (3) Finite Termination in polynomial time. (Proof in Appendix A).*

3.4 Agentic Memory Retrieval

H-SEAM performs seed-driven hierarchical retrieval to efficiently explore the graph under cost constraints. We generalize the search into an k-hop multi-stage strategy (default $k = 2$). The retrieval process proceeds in rounds:

Local multi-hop batching. Starting from a frontier $\mathcal{F}_t$, we collect neighbor candidates up to k hops: $\mathcal{Z}^{(h)} = \text{Neighbors}_h(\mathcal{F}_t)$. This preserves local connectivity while expanding the horizon.

LLM-guided screening. An LLM judge evaluates the expected informativeness of candidates in $\mathcal{Z}^{(h)}$ regarding query q (see in Appendix C). Only informative nodes are retained in the retrieval set $\mathcal{R}_t$. The system then checks a sufficiency-of-answerability condition $\Phi(\mathcal{R}_t; q) \geq \tau_{\text{suff}}$; if satisfied, retrieval terminates early.

Frontier re-centering. If sufficiency is not met, the frontier is updated to the most informative retained nodes ($\mathcal{F}_{t+1} = \text{TopK}\,\Delta I_n$), and a new expansion round initiates until the budget is exhausted.

Theorem 2 (Optimality of Local Expansion)**.** *The sufficiency-guided retrieval strategy theoretically establishes: (1) Global Maximum of Return-on-Cost is achieved at expansion depth* $k = 2$*; (2) Diminishing Returns for* $k > 2$ *due to exponential noise accumulation; and (3) Optimal Trade-off between contextual breadth and computational overhead. (Proof in Appendix B).*

In summary, H-SEAM restructures conversational memory along three temporal scales, enabling abstraction at memory-time rather than test-time. This design shifts the role of LLMs from explicit reasoning to semantic consolidation and validation. As a result, downstream agents can operate with reduced reasoning depth while maintaining robust multi-hop performance.

4 Evaluation

We evaluate H-SEAM against competing baselines in terms of both effectiveness and efficiency.

4.1 Experimental Setup

Datasets. We employ two complementary benchmarks: **LOCOMO** (Maharana et al., 2024), which evaluates very long-term coherence across more than 35 dialogue sessions, and **LongMemEval** (Wu et al., 2024), a diagnostic benchmark designed to test five core memory capabilities, including updates and abstention.

4.1.1 LLM Settings and Implementation Details

We utilize a diverse set of language models to verify robustness across four strategic dimensions. Unless otherwise specified, **GPT-4.1-mini** serves as the default backbone, and all embeddings are generated using `text-embedding-3-small`. To ensure fair comparison, all baselines are evaluated under the same backbone and embedding configuration unless explicitly stated otherwise.

Specifically, we evaluate: (1) **Capability Thresholds** using the GPT-4.1 series (from Nano to Standard); (2) **Generational Consistency** by contrasting GPT-5-mini with GPT-4.1-mini; (3) **Open-Weights Viability** by benchmarking

Table 1: **Main Results on LOCOMO.** Performance and Efficiency comparison. H-SEAM achieves the best overall performance, particularly excelling in LLM-Judge scores across complex reasoning tasks, while maintaining competitive latency. All metric values are in percentage (%), and Latency is in seconds (s).

Method	Single Hop			Multi Hop			Temporal			Open Domain			Overall			Latency
	F1	BLEU	LLM	F1	BLEU	LLM	F1	BLEU	LLM	F1	BLEU	LLM	F1	BLEU	LLM	(s)
Mem0	48.6	42.0	71.4	40.1	30.3	68.2	39.2	33.2	56.9	23.7	17.7	47.9	43.5	36.5	66.3	3.53
Zep	49.3	38.9	76.7	42.0	34.5	49.8	35.7	23.3	61.7	19.4	18.4	41.4	41.0	33.9	66.0	9.77
MemOS	45.6	38.3	78.4	35.6	26.7	64.3	53.7	46.4	73.2	29.6	22.4	55.2	44.4	36.9	73.3	7.71
MemoryOS	42.2	47.8	69.0	30.6	42.4	48.3	24.5	35.7	58.9	16.8	21.4	49.0	34.9	42.8	61.6	2.61
A-Mem	42.9	38.0	64.6	29.9	20.0	58.6	39.8	34.1	64.8	15.0	13.1	42.7	38.1	32.3	62.1	3.93
full-context	61.4	53.4	86.9	44.2	33.7	77.2	47.5	40.0	74.2	28.4	22.2	56.6	53.3	45.0	80.6	5.80
Ours	47.3	40.2	**81.2**	33.6	25.6	**77.0**	65.4	53.3	**85.4**	29.7	27.2	**60.4**	47.4	39.5	**80.0**	4.11

Table 2: Performance on LongMemEval. Abbreviations denote: **S-Pref**: Single-session Preference, **S-Asst**: Single-session Assistant, **Temp**: Temporal Reasoning, **Multi**: Multi-session, **Update**: Knowledge Update, **S-User**: Single-session User.

Type	MemoryOS	Mem0	A-mem	Zep	Ours
S-Pref	30.0	60.0	47.7	53.3	**75.0**
S-Asst	64.3	41.1	**96.4**	75.0	78.6
Temp	32.3	40.2	47.9	54.1	**60.1**
Multi	31.0	46.2	48.9	47.4	**66.1**
Update	48.7	70.1	64.1	74.4	**76.9**
S-User	80.0	81.4	92.9	92.9	**95.7**

the DeepSeek ecosystem against the GPT family; and (4) **Reasoning Paradigms** by comparing **DeepSeek-V3** with the reasoning-enhanced **DeepSeek-R1**, to assess whether intrinsic Chain-of-Thought capabilities improve memory utilization.

Baselines. We compare H-SEAM against representative memory paradigms:

- **Flat / Vector-based: Mem0** (Chhikara et al., 2025);
- **Graph-based: Zep** (Rasmussen et al., 2025) (time-aware subgraphs) and **A-Mem** (Xu et al., 2025) (associative Zettelkasten);
- **System-level: MemOS** (Li et al., 2025) (resource scheduling) and **MemoryOS** (Kang et al., 2025) (hierarchical paging).

4.2 Main Results

4.2.1 General Performance on LOCOMO

Table 1 summarizes generation quality on LOCOMO. We report LLM Judge Scores (hereafter referred to as Judge Scores) throughout this section. **H-SEAM achieves the strongest overall performance among all evaluated methods**, attaining an **Overall Judge Score of 80.0%** and outperforming the strongest baseline, MemOS (73.3%), by a substantial margin. H-SEAM consistently excels across tasks requiring complex reasoning, validating the effectiveness of hierarchical semantic organization.

Efficacy of Adaptive Abstraction in Complex Reasoning. On *Multi-hop* and *Open-domain* tasks, H-SEAM achieves Judge Scores of **77.0%** and **60.4%**, respectively, substantially outperforming structured baselines such as Zep and A-Mem. This performance gap highlights the benefit of density-driven abstraction: by constructing semantic bridges that connect related preferences across distant timestamps, H-SEAM overcomes the limitations of localized retrieval. The hierarchical structure reduces the effective reasoning path length, enabling the model to synthesize information that is semantically related but temporally distant.

Precision in Chronological Reconstruction. For *Temporal Reasoning*, H-SEAM achieves a Judge Score of **85.4%**, significantly exceeding system-level approaches such as MemOS (73.2%). This result demonstrates that a content-adaptive topology yields stronger narrative continuity than rigid resource scheduling or fixed paging strategies. Rather than enforcing arbitrary memory boundaries, H-SEAM dynamically merges events based on semantic density, aligning memory organization with the natural flow of dialogue and enabling finer-grained temporal reconstruction.

From Keyword Recall to Logical Coherence. We observe a clear divergence between lexical recall and reasoning quality. Flat architectures

Table 3: Results of Ablation Studies Across Different Categories. All values are reported in percentage (%).

Category	Full Model			w/o Expansion			w/o Fragment			w/o Abstraction		
	BLEU	F1	LLM	BLEU	F1	LLM	BLEU	F1	LLM	BLEU	F1	LLM
Multi-hop (1)	25.6	33.6	**77.0**	24.8	31.6	75.2	**27.5**	**34.5**	75.5	24.5	32.5	71.6
Temporal (2)	**53.3**	**65.6**	**85.4**	49.6	61.7	80.4	50.1	62.0	77.9	50.0	62.2	77.9
Open-domain (3)	27.2	29.7	**60.4**	25.5	27.6	51.0	**26.4**	**28.5**	60.4	24.2	26.7	60.4
Single-hop (4)	40.2	47.3	81.2	40.8	47.6	77.2	38.2	44.0	77.2	**40.8**	**47.6**	**83.2**
Overall	**39.5**	**47.4**	**80.0**	38.7	46.3	75.8	38.0	45.1	76.0	38.7	46.5	78.6

such as Mem0 achieve competitive F1 scores (e.g., 40.1% on Multi-hop) but lag behind in Judge Scores (68.2% versus H-SEAM's **77.0%**). While surface-level recall may suffice for F1-based metrics, answering complex queries requires preserving the structural logic surrounding entities. H-SEAM's hierarchical synthesis maintains these reasoning contexts, ensuring that retrieved information satisfies logical coherence rather than mere lexical overlap. This further supports our design choice of prioritizing structural coherence over surface recall.

4.2.2 Fine-grained Analysis on LongMemEval

To analyze specific memory capabilities, we compare H-SEAM with strong baselines on LongMemEval (Table 2). H-SEAM achieves the highest Judge Score in five out of six categories, maintaining near-perfect performance on basic recall tasks while demonstrating superior robustness in complex scenarios.

Robustness in Preference Tracking. On *Single-session Preference*, H-SEAM achieves a Judge Score of 75.0%, significantly outperforming Zep and A-Mem. This suggests that associative baselines struggle with conflicting updates. In contrast, H-SEAM consolidates evolving preferences into unified high-level representations, effectively prioritizing the users most recent state.

Stability in Multi-session Consistency. In *Multi-session* settings, H-SEAM attains a Judge Score of 66.1%, notably higher than all baselines. This result highlights the role of vertical abstraction in bridging long temporal gaps. Unlike graph systems that rely primarily on horizontal links, H-SEAM anchors disjoint sessions to persistent high-level abstractions (e.g., long-term career goals), enabling consistent recall across extended interactions.

Adaptability in Knowledge Evolution. For *Knowledge Update*, H-SEAM achieves 76.9%, outperforming OS-based methods. While recency-biased retrieval can handle simple updates, H-SEAM benefits from dynamic upward abstraction. Instead of merely appending new information, it distills evolving knowledge into salient nodes, ensuring that critical updates are structurally prioritized.

4.3 Efficiency

Table 1 reports end-to-end latency on LOCOMO. Although H-SEAM (4.11s) incurs slightly higher latency than lightweight vector-based methods such as Mem0 (3.53s) due to structured retrieval, it remains **significantly more efficient** than complex structured systems, including Zep (9.77s) and MemOS (7.71s).

We observe that H-SEAM achieves a **1.4× speedup** compared to full-context processing (5.80s). This result indicates that the proposed two-hop expansion strategy effectively prunes the retrieval space, allowing H-SEAM to achieve comparable reasoning quality (80.0% versus 80.6%) with improved inference speed, thereby supporting its scalability for real-time interaction.

4.4 Sensitivity to Model Capabilities

Model Scaling and Generational Evolution. As shown in Table 5, performance does not scale linearly with model size. **GPT-4.1-mini** equipped with H-SEAM already demonstrates strong competence, indicating that architectural design can outweigh parameter scaling. Moreover, **GPT-5-mini** exhibits performance comparable to GPT-4.1-mini, suggesting that H-SEAM generalizes robustly across model generations rather than relying on model-specific emergent behaviors.

Reasoning Paradigms and Open-Weights Viability. Table 5 further shows that **DeepSeek-V3** performs competitively with proprietary GPT models, supporting the open-weights viability of H-SEAM. In contrast, the reasoning-enhanced

Table 4: Ablation study on graph expansion hops. Diffusing 2 hops outward achieves the best trade-off between performance and efficiency. Increasing to 3 hops introduces noise and significantly raises computational costs.

k-Hop	Single hop	Multi hop	Temporal	Open domain	Overall	Token	Time
0	77.2	75.2	80.4	51.0	75.8	4,161,458	3.08s
1	78.5	76.2	80.4	54.2	77.0	4,996,208	3.46s
2	**81.2**	**77.0**	**85.4**	**60.4**	**80.0**	**6,617,888**	**4.11s**
3	81.1	77.0	76.0	62.5	78.1	16,017,443	10.12s

Table 5: Overall LLM Scores of different Models

Model	Overall
GPT-4.1-mini	**80.0**
GPT-4.1-nano	67.3
GPT-5-mini	78.5
GPT-4.1	79.3
DeepSeek-V3	76.2
DeepSeek-R1	69.2

DeepSeek-R1 does not yield consistent improvements and, in some cases, performs worse than V3. We attribute this to a paradigm mismatch: H-SEAM shifts the primary cognitive burden from test-time reasoning to memory-time abstraction by consolidating fragmented dialogues into coherent structured representations. When such abstraction is already in place, additional runtime Chain-of-Thought reasoning may become redundant or introduce unnecessary deliberation noise, reducing its marginal benefit.

4.5 Ablation Study

Component-wise ablations are reported in Table 3, with all metrics presented as percentages consistent with the main results.

Impact of Low-level Grounding. Removing fragment-level representations reduces the Judge Score to **76.0%**. Without fine-grained grounding, the model is prone to hallucination, capturing only coarse semantic content while failing to verify specific entities.

Impact of Retrieval Scope. Restricting retrieval to a single hop leads to substantial performance degradation in *Temporal Reasoning* (**85.4%** $\rightarrow$ **80.4%**) and Open-domain tasks. Single-hop retrieval confines reasoning to local neighborhoods, whereas H-SEAM's multi-hop expansion ($k = 2$) effectively bridges disjoint dialogue turns.

Impact of High-level Abstraction. Removing abstraction layers causes the most pronounced decline in Multi-hop Reasoning. High-level nodes function as semantic connectors; without them, the agent must traverse long chains of raw fragments, increasing the likelihood of reasoning failures.

4.6 Impact of Graph Expansion Depth

We analyze the effect of expansion depth (k) on performance and efficiency (Table 4), observing a clear **inverted U-shaped trend**. Restricting retrieval to local neighborhoods ($k < 2$) fails to capture distant but semantically related connections, causing significant performance drops in Multi-hop and Open-domain tasks. Expanding to **two hops** ($k = 2$) yields the global optimum (Overall Judge Score **80.0%**), effectively bridging indirect neighbors while maintaining a focused context window. However, further increasing depth to $k = 3$ leads to diminishing returns (score drops to 78.1%) and incurs substantial overhead, with token usage increasing by $\approx 2.4\times$ and latency nearly doubling. This confirms that excessive expansion degrades the signal-to-noise ratio, justifying our default choice of $k = 2$ as the optimal trade-off between reasoning breadth and computational efficiency.

5 Conclusion

We introduced H-SEAM, a dynamic multi-level memory framework specifically designed to resolve longhorizon reasoning instability in LLM agents.

By coupling density-driven evolution with optimal graph expansion, H-SEAM effectively addresses the limitations of semantic fragmentation and structural rigidity.

Experiments on LOCOMO and LongMemEval demonstrate that H-SEAM consistently outperforms stateoftheart baselines, significantly improving overall accuracy by 6.7% while maintaining highly competitive computational efficiency.

These results suggest that content-adaptive topological organization is essential for longterm consistency, offering a scalable paradigm for future agentic memory systems beyond simple information accumulation.

6 Limitations

Our experiments demonstrate the robustness of H-SEAM in long-context retrieval and reasoning. However, we identify two areas where the current paradigm can be further expanded:

Dependency on Instruction-Following Capabilities. H-SEAM's bottom-up construction relies on the precision of the underlying LLM to extract entities and events (as detailed in the Index Builder). Our ablation studies suggest that while models like **GPT-4.1-mini** and **DeepSeek-V3** perform exceptionally well, extremely lightweight or non-instruction-tuned models may struggle with the complex formatting constraints required for structured graph generation. The system's performance in its current instantiation suggests a practical lower bound on the required model capability., and future work could explore distilling these extraction capabilities into smaller, specialized encoders to further reduce deployment costs.

Static Hyperparameter Sensitivity. In the current implementation, key thresholds for node merging (e.g., similarity threshold τ_{sim}) and cluster synthesis (e.g., density threshold $N_{\max}$) are set empirically. While these settings generalize well across LOCOMO and LongMemEval, real-world interactions with highly varying information densities (e.g., a fast-paced debate vs. casual chitchat) might benefit from adaptive hyperparameters. Future iterations will investigate **dynamic thresholding mechanisms** that adjust the abstraction granularity in real-time based on the entropy of the incoming dialogue stream.

AI Writing Assistance

In accordance with the conference policy on AI-assisted writing, we acknowledge the use of Large Language Models (specifically **GPT-5** and **Gemini**) to assist with linguistic polishing, LaTeX formatting, and refining the clarity of certain text passages. We certify that the scientific conceptualization, experimental design, and data analysis are the original work of the authors. All AI-generated suggestions were manually reviewed and verified by the authors to ensure accuracy and prevent plagiarism.

References

György Buzsáki. 2015. Hippocampalneocortical dialogue and memory consolidation. *Science*, 347:1054–1059.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Fergus IM Craik and Janine M Jennings. 1992. Human memory.

Yadin Dudai, Avi Karni, and Jan Born. 2015. The consolidation and transformation of memory. *Neuron*, 88(1):20–32.

Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. 2023. S3: Social-network simulation system with large language model-empowered agents. *arXiv preprint arXiv:2307.14984*.

Bernal Jimenez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*. NeurIPS 2024.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2025. Evaluating memory in llm agents via incremental multi-turn interactions. *Preprint*, arXiv:2507.05257.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. *Preprint*, arXiv:2506.06326.

John Kounios, Roderick W. Smith, Wei Yang, Peter Bachman, and Mark D'Esposito. 2001. Cognitive association formation in human memory revealed by spatiotemporal brain imaging. *Neuron*, 29(1):297–306.

Andrew K. Lampinen and 1 others. 2021. Towards mental time travel: hierarchical memory for reinforcement learning agents. *arXiv preprint arXiv:2105.14039*.

Timothée Lesort and 1 others. 2020. Generative replay for continual learning: A survey. *Neural Networks*, 125:126–142.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, and 1 others. 2025. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*.

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

Joon Sung Park and 1 others. 2023b. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 6(3):1.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *NeurIPS*.

Ivilin Stoianov, Domenico Maisto, and Giovanni Pezzulo. 2022. The hippocampal formation as a hierarchical generative model supporting generative replay and continual learning. *Progress in Neurobiology*, 217:102329.

Hanlin Wang and 1 others. 2024a. Langmem: Long-term memory for language model based agents. *arXiv preprint arXiv:2403.05969*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024b. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*.

Xinyang Wu and 1 others. 2023. Memgpt: Towards lifelike memory in llm agents. *arXiv preprint arXiv:2310.08518*.

Haotian Xu and et al. 2024. Agentic memory: Structuring long-term memory for language agents. In *arXiv preprint arXiv:2404.07307*.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

A Additional Details on Hierarchical Memory Compression and Theoretical Guarantees

A.1 Strength-Aware Edge-Density-Constrained Greedy Merging

To balance scalability and fidelity, H-SEAM performs periodic graph compression using a **Strength-Aware Edge-Density-Constrained Greedy Merging** strategy. This process selectively merges semantically redundant, weakly connected, and low-strength nodes while preserving high-importance nodes and maintaining uniform structural density. It thereby achieves an optimal trade-off among *semantic compactness*, *behavioral relevance*, and *structural uniformity*.

Each iteration evaluates candidate node pairs $(n_{\ell,i}, n_{\ell,j})$ according to their similarity, node strength, and local edge density deviation. Merging is executed only when: (1) semantic similarity exceeds a threshold τ_ℓ; (2) local degree and average edge weight are below structural thresholds; and (3) cumulative node strength remains within a preservation bound. The newly formed composite node inherits aggregated embeddings, strengths, and edge sets, followed by density-aware rebalancing to maintain global topological consistency.

Information Preservation and Uniformity Guarantee. Let $\mathcal{I}(G_\ell)$ denote the information content of the level-ℓ graph, estimated by the average pairwise dissimilarity of node embeddings weighted by their strengths:

$$\mathcal{I}(G_\ell) = \frac{1}{|\mathcal{N}_\ell|^2} \sum_{i,j} s_{\ell,i} s_{\ell,j} \big(1 - \mathrm{sim}(n_{\ell,i}, n_{\ell,j})\big).$$

The merging process is designed such that

$$\mathcal{I}(G'_\ell) \geq (1 - \epsilon)\mathcal{I}(G_\ell),$$

ensuring that the expected information loss per compression cycle is bounded by a tunable factor ϵ. Simultaneously, density regularization guarantees $\rho(G'_\ell) \approx \rho^*_\ell$, maintaining global and local edge density uniformity. Thus, the hierarchical memory remains both information-sufficient and structurally stable throughout compression.

Theorem 3 (Bounded Information Loss and Structural Stability)**.** *Let $G_\ell = (\mathcal{N}_\ell, E_\ell)$ be a level-ℓ memory graph satisfying degree bounds $d_{\min} \leq \deg(n) \leq d_{\max}$ and target density ρ^*_ℓ. If*

the greedy merging process obeys the similarity, strength, and density constraints described above, then the post-merge graph G'_ℓ satisfies:

$$\mathcal{I}(G'_\ell) \geq (1-\epsilon)\mathcal{I}(G_\ell), \quad |\rho(G'_\ell) - \rho^*_\ell| \leq \epsilon_\rho, \quad \textit{and} \quad \frac{\sum_i s'_i}{\sum_i s_i} \geq \eta_s.$$

In particular, $\mathcal{I}(G_\ell)$ is non-increasing with each merge and converges to a finite fixed point, ensuring both bounded information degradation and structural equilibrium.

A.2 Proof of Theorem

Proof. Consider a single merge of nodes (n_i, n_j) satisfying the eligibility constraints: $\mathrm{sim}(n_i, n_j) \geq \tau_\ell$, $\min(s_i, s_j) \leq s_{\mathrm{th}}$, and $|\rho_{\text{local}} - \rho^*_\ell| \leq \epsilon_\rho$.

(1) Bounded Information Loss. Since $\mathrm{sim}(n_i, n_j) \geq \tau_\ell$, the merged embedding $\mathbf{v}' = \frac{w_i \mathbf{v}_i + w_j \mathbf{v}_j}{w_i + w_j}$ satisfies

$$1 - \mathrm{sim}(\mathbf{v}', \mathbf{v}_k) \leq \max\{1 - \mathrm{sim}(\mathbf{v}_i, \mathbf{v}_k),\, 1 - \mathrm{sim}(\mathbf{v}_j, \mathbf{v}_k)\} + O(1 - \tau_\ell).$$

Thus, the aggregate change in $\mathcal{I}(G_\ell)$ due to the merge is upper-bounded by

$$\Delta\mathcal{I} \leq \frac{2 s_i s_j}{|\mathcal{N}_\ell|^2}(1 - \tau_\ell) + O\left(\frac{d_{\max}}{|\mathcal{N}_\ell|^2}\right),$$

and accumulating over all merges yields $\mathcal{I}(G'_\ell) \geq (1-\epsilon)\mathcal{I}(G_\ell)$ for $\epsilon = C(1 - \tau_\ell)$ with a small constant C dependent on $d_{\max}$.

(2) Density Stability. Each merge updates edges locally within the neighborhoods of n_i and n_j. Since $|\rho(G_\ell) - \rho^*_\ell| \leq \epsilon_\rho$ before merging, and rebalancing adds or prunes edges to restore target density, it follows that $|\rho(G'_\ell) - \rho^*_\ell| \leq \epsilon_\rho$ by construction.

(3) Strength Preservation. The merged nodes strength is $s'_k = s_i + s_j$. Hence, the total strength remains unchanged except for negligible rounding or decay factors: $\sum_i s'_i = \sum_i s_i$. In practice, a small decay coefficient δ_s may be applied to balance drift, giving $\sum_i s'_i / \sum_i s_i \geq 1 - \delta_s = \eta_s$.

(4) Convergence. Because each merge reduces $|\mathcal{N}_\ell|$ by one while maintaining non-increasing $\mathcal{I}(G_\ell)$ and bounded density deviations, the sequence $\{G^{(t)}_\ell\}$ forms a monotone, bounded process. By the monotone convergence theorem, it converges to a finite fixed point G^∞_ℓ satisfying the inequalities in the theorem. □

A.3 Complexity and Convergence Analysis of Greedy Merging

Preliminaries. Let $G_\ell = (\mathcal{N}_\ell, E^{\mathrm{intra}}_\ell)$ be the level-ℓ graph with $n_\ell = |\mathcal{N}_\ell|$ nodes and $m_\ell = |E^{\mathrm{intra}}_\ell|$ edges. The greedy procedure repeatedly selects a merge pair (i, j) maximizing the marginal gain $\Delta\mathcal{J}$ under hard constraints until the node budget B_ℓ is met or no feasible pair remains. Denote $M^\star_\ell = \max\{n_\ell - B_\ell, 0\}$ as the *maximum number of merges* needed at level ℓ.

Candidate maintenance and per-iteration cost. We maintain a candidate set $\mathcal{C}$ of feasible pairs filtered by the three eligibility conditions (similarity, degree/weight, strength). Two practical implementations are assumed:

1. **Exact screening.** Compute all pairwise similarities with a threshold τ_ℓ; this induces at most $P_\ell \leq \min\{\binom{n_\ell}{2}, \#\{(i,j) : \mathrm{sim}(i,j) \geq \tau_\ell\}\}$ candidates. Initial build is $O(n^2_\ell d)$ for dense embeddings (dimension d).
2. **Approximate screening (recommended).** Use ANN (e.g., HNSW/IVF-PQ) to get top-K neighbors per node above τ_ℓ. This yields $P_\ell = O(n_\ell K)$ expected candidates with index build time $\tilde{O}(n_\ell d)$ and query time $\tilde{O}(K)$ per node (the $\tilde{O}$ hides log factors).

We store $\Delta\mathcal{J}$ for all $(i,j) \in \mathcal{C}$ in a max-heap $\mathcal{H}$ of size $|\mathcal{C}| = O(P_\ell)$. Each iteration performs: (i) extract-max in $O(\log P_\ell)$; (ii) **feasibility recheck** (degree/density/strength) in amortized $O(\deg(i) + \deg(j) + R_{loc})$ time, where R_{loc} is the number of affected local density cells; (iii) **merge-update**: create n', relink edges, update local/global density statistics, update strengths and embedding, and *lazy* refresh of impacted pairs in $\mathcal{H}$.

Let $\bar{\Delta}$ be the average number of pairs impacted by one merge (i.e., pairs incident to i or j or to neighbors whose density/degree penalties change). Using adjacency lists and cell-wise density bookkeeping, we obtain the amortized per-merge cost:

$$T_{\text{iter}} = O\big(\log P_\ell + \deg(i) + \deg(j) + R_{loc} + \bar{\Delta} \log P_\ell\big).$$

With degree bounds $d_{\max}$ and bounded local neighborhoods per cell, this simplifies to $T_{\text{iter}} = \tilde{O}\big(1 + \bar{\Delta}\big)$.

Overall time complexity. Over at most $M_\ell^\star$ merges,

$$T_{\text{total}} = T_{\text{build}} + \sum_{t=1}^{M_\ell^\star} T_{\text{iter}}^{(t)} = \begin{cases} O(n_\ell^2 d) + \tilde{O}\big(M_\ell^\star(1+\bar{\Delta})\big), & \text{exact} \\ \tilde{O}(n_\ell d) + \tilde{O}\big(M_\ell^\star(1+\bar{\Delta})\big), & \text{ANN} \end{cases}$$

where T_{build} is the one-off similarity/index construction. In practice, ANN with small K and degree/density caps yields nearly linear behavior in n_ℓ for fixed B_ℓ.

Space complexity. We store node attributes, embeddings, and adjacency lists: $O(n_\ell d + m_\ell)$. Candidate heap and ANN index contribute $O(P_\ell)$ and $\tilde{O}(n_\ell)$ respectively. Thus $S_{\text{total}} = O(n_\ell d + m_\ell + P_\ell)$.

Convergence and termination. We define a potential function combining the (negative) objective and feasibility slack:

$$\Phi_t = -\sum_{(i,j)\in\mathcal{C}_t} \big[\Delta\mathcal{J}_{ij}\big]_+ + \lambda_{\text{feas}}\,\Xi_t,$$

where Ξ_t aggregates violations of hard constraints (degree/density/budget), and $[\cdot]_+$ truncates at zero. Each accepted merge strictly reduces $|\mathcal{N}_\ell|$ by 1 and does not increase Ξ_t (merges violating any hard constraint are rejected). Therefore:

1. **Finite termination.** The process halts in at most $M_\ell^\star$ merges since budget B_ℓ or candidate exhaustion is reached.
2. **Monotone feasibility.** Because feasibility is checked before committing, Ξ_t is non-increasing; once $\Xi_t = 0$, it stays 0.
3. **No cycling.** Node identifiers are not reused in a way that recreates pre-merge states; the merge operator is idempotent on pairs not reintroduced, hence no cycles arise.

Stability of density and strength. Let $\rho(G_\ell)$ be global density and ρ_r cell-wise densities with targets ρ_ℓ^*, ρ_r^*. The post-merge update enforces hard tolerance $|\rho(G_\ell) - \rho_\ell^*| \le \epsilon_\rho$ and bounds local deviations via the penalty U_ρ. Hence, the sequence $\{\rho(G_\ell^{(t)})\}_t$ remains in a closed interval around ρ_ℓ^*; similarly, the cumulative strength ratio $\sum_i s_i' / \sum_i s_i \ge \eta_s$ is preserved by construction. Together, these yield *structural stability*: degrees, densities, and total strength remain within pre-specified bounds throughout.

Information-retention bound (sketch). Recall the information functional

$$\mathcal{I}(G_\ell) = \frac{1}{n_\ell^2}\sum_{i,j} s_{\ell,i}s_{\ell,j}\big(1-\text{sim}(i,j)\big),$$

which is Lipschitz in pairwise similarities. A single merge of two τ_ℓ-similar, low-strength nodes perturbs at most $O(\deg(i)+\deg(j))$ pairwise terms. With degree cap $d_{\max}$ and strength preservation $s_k' = s_i + s_j$, the merge induces a bounded change $|\mathcal{I}(G_\ell') - \mathcal{I}(G_\ell)| \le L \cdot \epsilon_{\text{sim}}$ for some constant L depending on $d_{\max}$ and strength scaling; accumulating over at most $M_\ell^\star$ merges and imposing the density tolerance ensures $\mathcal{I}(G_\ell') \ge (1-\epsilon)\mathcal{I}(G_\ell)$ with tunable ϵ (via $\tau_\ell, \epsilon_\rho, \eta_s$). Thus information loss is controlled while sparsity increases.

Quality guarantee under submodularity (optional). If the negative of the objective can be written as a *monotone submodular* set function over the complement node set, and the constraints reduce to a (partitioned) knapsack or matroid, then the classic greedy selection achieves a $(1-1/e)$-approximation to the optimum. Concretely, when

$$\Delta\mathcal{J}((i,j)\mid G_\ell) = f(\text{sim}) - \alpha_s(s_i+s_j) - \alpha_\rho\Delta U_\rho - \alpha_{\deg}\Delta\text{Var}_{\deg}$$

is *approximately* separable and the penalty terms are convex and Lipschitz so that the resulting gain exhibits diminishing returns, a curvature-adjusted bound $1-e^{-1/\kappa}$ (with total curvature $\kappa \in [1,\infty)$) applies. This provides a principled near-optimality guarantee for the greedy merge sequence. (When the conditions do not hold, our guarantees revert to finite termination and bounded information loss above.)

Incremental and multi-level runtime. Applying the above per level and summing over $\ell \in L$ gives

$$\sum_{\ell\in L}\Big(T_{\text{build}}^{(\ell)} + \tilde{O}\big(M_\ell^\star(1+\bar{\Delta}_\ell)\big)\Big),$$

where $\bar{\Delta}_\ell$ is typically small under degree/density caps. Inter-level reconnection E^{inter} touches only boundary nodes of the merged pair, costing $\tilde{O}(|\partial(i)\cup\partial(j)|)$ per merge and thus linear in local degree.

The merging algorithm (i) *terminates* in at most $M_\ell^\star$ steps, (ii) maintains *structural feasibility* (degree, density, strength) throughout, (iii) offers *controlled information loss* with tunable ϵ, and (iv)

achieves near-linear practical runtime with ANN-based screening and bounded local updates. Under mild approximate-submodularity, the greedy rule enjoys a constant-factor *approximation* to the optimum.

B Additional Theoretical Result on m-Hop Expansion

Theorem 4 (Conditional Optimality of k for 4-Layer Graph)**.** *In a hierarchical memory graph with four abstraction levels $\{0,1,2,3\}$, assuming an effective branching factor $b > 1$, geometric decay of informativeness with hop distance ($r \in (0,1)$), and convex LLM screening cost per hop ($\mathrm{Cost}_h \propto b^h \phi_h$ with $\phi_1 > 1$, $\phi_2 \approx 1$, $\phi_{h\geq 3} > 1$), the expected return-on-cost ratio ROC_h first increases and then decreases with hop count.*

Therefore, the per-round optimal expansion depth that maximizes the informativenesscost trade-off satisfies:

$$\boxed{k^\star = 2.}$$

That is, expanding up to two hops per round yields the best overall efficiency in a 4-layer hierarchical graph.

B.1 Analysis of Optimal k in a 4-Layer Graph

Setup. We analyze the *k-hop batching* depth per round on a 4-layer hierarchical graph (levels $0,1,2,3$). Let a round start from a frontier of size K with effective branching factor $b > 1$. Assume:

1. **Candidate growth.** The number of hop-h candidates is $N_h \approx K\, b^h$.

2. **Informativeness decay.** The expected (per-node) informativeness gain decays geometrically with hop distance: $g_h = g_0\, r^h$ with $0 < r < 1$ (semantic relatedness attenuates with distance). A retained fraction $p_h \in (0,1]$ (post-LLM screening) gives hop-h total gain $\Delta I_h \approx p_h N_h g_h \;=\; K\, p_h\, g_0\, (br)^h$.

3. **LLM screening cost.** Descriptor length is capped at $L_{\max}$ per node. Batched screening uses per-call token cap $T_{\max}$ and convex per-call cost $f(T) = c_0 + c_1 T + c_2 T^\alpha$ with $\alpha > 1$.

Thus hop-h incurs

$$\mathrm{Cost}_h \;\approx\; \left\lceil \frac{N_h L_{\max}}{T_{\max}} \right\rceil \cdot f(T_{\max})$$

$$\approx\; \frac{K\, b^h L_{\max}}{T_{\max}} \cdot f(T_{\max}) \;\propto\; b^h,$$

where the proportionality absorbs $K, L_{\max}, T_{\max}, f(T_{\max})$.

Define the hop-h *return-on-cost* (ROC):

$$\mathrm{ROC}_h \;\triangleq\; \frac{\Delta I_h}{\mathrm{Cost}_h}$$

$$\approx\; \frac{K\, p_h\, g_0\, (br)^h}{C_0\, b^h} \;=\; \frac{K\, g_0}{C_0}\, p_h\, r^h,$$

where $C_0 > 0$ is a constant. Hence, absent other effects, $\mathrm{ROC}_h \propto p_h\, r^h$ decreases with h because $r < 1$ and typically p_h nonincreasing in h.

Including convex compute and multi-call overhead. The previous expression assumes perfectly *balanced* batches at $T_{\max}$. In practice, for small h we may underfill $T_{\max}$ (wasting per-call overhead c_0), while for large h we overflow and need many calls (convex penalty $c_2 T^\alpha$ accumulates across calls). We capture this with a hop-dependent *cost multiplier* $\phi_h \geq 1$:

$$\mathrm{Cost}_h \;\approx\; C_0\, b^h\, \phi_h,$$

$$\text{with } \phi_1 > 1 \text{ (underfill)},$$

$$\phi_2 \approx 1,\; \phi_{h\geq 3} > 1 \text{ (overflow)}.$$

Then

$$\mathrm{ROC}_h \;\approx\; \frac{K\, g_0}{C_0}\, \frac{p_h\, r^h}{\phi_h}.$$

Selecting k in a 4-layer graph. Per round we may include hops $h = 1,2,3$ (since the graph has 4 layers). The *optimal* $k^\star$ maximizes cumulative net value under a per-round budget, which, in the marginal sense, is equivalent to choosing the largest h such that ROC_h remains nondecreasing:

$$k^\star \;=\; \arg\max_{m\in\{1,2,3\}} \sum_{h=1}^{k} (\Delta I_h - \lambda\, \mathrm{Cost}_h) \;\iff$$

$$\text{include hop } h \text{ iff } \frac{\Delta I_h}{\mathrm{Cost}_h} \geq \tau_{\mathrm{ROC}},$$

for some implicit threshold τ_{ROC} tied to λ and remaining budget.

Using the model above,

$$\frac{\mathrm{ROC}_{h+1}}{\mathrm{ROC}_h} \;=\; \frac{p_{h+1}}{p_h} \cdot \frac{r}{\phi_{h+1}/\phi_h}.$$

Thus ROC_h decreases after hop h when

$$r \;<\; \frac{\phi_{h+1}}{\phi_h} \cdot \frac{p_h}{p_{h+1}}.$$

Empirically, $p_{h+1} \le p_h$ and $\phi_2 \approx 1$ while $\phi_1 > 1$, $\phi_3 > 1$. Hence:

(A) $\dfrac{\mathrm{ROC}_2}{\mathrm{ROC}_1} \;\approx\; \dfrac{p_2}{p_1} \cdot \dfrac{r}{\phi_2/\phi_1} \;\approx\; \dfrac{p_2}{p_1} \cdot r \cdot \phi_1$
(often ≥ 1 since $\phi_1 > 1$ corrects underfill);

(B) $\dfrac{\mathrm{ROC}_3}{\mathrm{ROC}_2} \;\approx\; \dfrac{p_3}{p_2} \cdot \dfrac{r}{\phi_3/\phi_2} \;\approx\; \dfrac{p_3}{p_2} \cdot \dfrac{r}{\phi_3}$
(often < 1 since $\phi_3 > 1$ and $p_3 \le p_2$).

Conclusion (typical regime). In common settings (moderate b; $r \in [0.4, 0.7]$; balanced batching at hop-2; convex cost with $\alpha > 1$):

$$\mathrm{ROC}_2 \;\gtrsim\; \mathrm{ROC}_1 \quad \text{and} \quad \mathrm{ROC}_3 \;<\; \mathrm{ROC}_2.$$

Therefore the optimal per-round depth on a 4-layer graph is

$$\boxed{k^\star = 2}$$

i.e., include up to 2 hops per round.

When can $k^\star$ differ?

- $k^\star = 1$ if r is very small or the frontier already saturates the token cap ($\phi_1 \approx 1$), making hop-2 under-informative.
- $k^\star = 3$ only if informativeness decays slowly ($r \approx 1$), screening stays effective ($p_3 \approx p_2$), and batching remains balanced at hop-3 ($\phi_3 \approx 1$), which is uncommon because $N_3 \propto b^3$ typically forces multiple calls and raises ϕ_3.

Rule-of-thumb test (closed-form check). Using $p_h \approx p$ (flat retention) and $\phi_1 = \gamma > 1$, $\phi_2 = 1$, $\phi_3 = \eta > 1$:

$$\mathrm{ROC}_1 \propto \frac{r}{\gamma}, \quad \mathrm{ROC}_2 \propto r^2, \quad \mathrm{ROC}_3 \propto \frac{r^3}{\eta}.$$

Then $k^\star = 2$ iff

$$r^2 \ge \max\left(\frac{r}{\gamma}, \frac{r^3}{\eta}\right) \iff r \ge \frac{1}{\gamma} \quad \text{and} \quad r \le \sqrt{\eta}.$$

With mild underfill $\gamma \in [1.2, 1.5]$ and overflow $\eta \in [1.3, 2.0]$, this condition holds for $r \in [0.5, 1)$, matching typical semantic decay, hence $k^\star = 2$.

C LLM Prompts and Implementation Details

To ensure reproducibility, we provide the exact prompt templates used in H-SEAM. We utilize `GPT-4.1-mini` (or equivalent) to guarantee deterministic output. The system employs a multi-stage prompt pipeline covering entity extraction, timeline construction, high-level pattern mining, and retrieval-augmented reasoning.

C.1 Memory Construction Prompts

1. Middle-Level Entity & Relationship Extraction. This prompt extracts stable entities and their relationships while handling deduplication against existing memory.

2. Middle-Level Timeline Extraction (Events, States, Contexts). This prompt extracts dynamic episodes from dialogue slices, distinguishing between discrete events, ongoing states, and background contexts.

C.2 Abstraction and Reasoning Prompts

3. High-Level Pattern Mining (Cluster Synthesis). This prompt implements the **LLM-Driven Conceptual Synthesis**. It takes a cluster of related events/states and synthesizes high-level patterns, preferences, and behavior rules.

4. Retrieval and Final Answer Generation. This prompt handles the final reasoning step, including relative time resolution, conflict handling, and seed node selection for graph expansion.

System Instruction: You are an expert in information extraction. Extract entities and relationships from the conversation fragment below.

Existing Entities:
`{{EXISTING_ENTITIES_LIST}}`

Extraction Rules:
• **Entities:** Extract people, organizations, places, and personalized concepts.
• **Constraint:** Focus on **BEING vs. DOING** (e.g., extract "supportive person", ignore "is running").
• **Action Logic:**
  - `create_new`: For completely new entities.
  - `update_existing`: If a new entity matches an existing one (same ID), supplement its info.
  - `link_to_existing`: If a new entity relates to an existing one.
• **Relationships:** Extract STABLE ties (e.g., "friends with", "works at") rather than temporary interactions.

Input Fragment: `{{FRAGMENT_TEXT}}`

Output Format (JSON):

```
{
  "entities": [
    { "name": "Name", "content": "Static description", "action": "create_new" },
   { "name": "Name", "content": "Updated info", "action": "update_existing", "existing_entity_id":
"id_1" }
  ],
  "relationships": [
    { "source": "A", "target": "B", "content": "Desc", "action": "create_new" }
  ]
}
```

Table 6: Prompt 1: Entity & Relationship Extraction

System Instruction: Extract timeline information (events, states, contexts) from the conversation fragment.

Completeness Requirement: Extract ALL timeline-relevant information explicitly mentioned, including casual actions (purchases, plans).

Classification Guidelines:
• **EVENT:** Explicit actions or occurrences (e.g., "bought a figurine", "meetings").
• **STATE:** Persistent conditions or emotions (e.g., "loves art").
• **CONTEXT:** Background framing information.

Input Fragment: `{{FRAGMENT_TEXT}}`
Session Time: `{{SESSION_TIME}}`

Output Format (JSON):

```
{
  "events": [
    {
      "content": "Description including time/location",
      "participants": ["..."],
      "conversation_time": "...",
      "relative_time": "e.g., last Friday",
      "action": "create_new"
    }
  ],
  "states": [ ... ],
  "contexts": [ ... ]
}
```

Table 7: Prompt 2: Timeline Extraction

System Instruction: Analyze the following clustered events, states, and contexts to identify high-level patterns.

Input Clusters:
• Events: `{{EVENTS_TEXT}}`
• States: `{{STATES_TEXT}}`
• Contexts: `{{CONTEXTS_TEXT}}`

Analysis Tasks:
1. **Event Cluster Summary:** Brief summary (1-2 sentences) of the larger activity.
2. **Patterns:** Recurring behaviors (e.g., "Researches before buying").
3. **Preferences:** Likes/Dislikes (e.g., "Prefers fruity desserts").
4. **Behavior Rules:** Decision-making logic (e.g., "Considers guests when hosting").

Output Format (JSON):

```
{
  "event_cluster": {
    "description": "User explored Ethiopian cuisine...",
    "significance": "high"
  },
  "patterns": [ { "person": "...", "pattern_type": "...", "description": "..." } ],
  "preferences": [ ... ],
  "behavior_rules": [ ... ]
}
```

Table 8: Prompt 3: High-Level Abstraction

System Instruction: Based on the knowledge graph information below, provide a precise answer to the question.

Core Principles:
• **Time Resolution:** Compute absolute time using `conversation_time` + `relative_time`. Always prefer the most recent information (latest timestamp or highest node ID).
• **Relevance Verification:** Ensure retrieved info refers to the *exact concept* (e.g., "pineapple" != "apple").
• **Insufficient Info:** If context is missing, output "Insufficient information" but still select 3-5 relevant seed node IDs for expansion.

Question: `{{QUERY}}`
Knowledge Graph Context: `{{CONTEXT}}`

Output Format (JSON):

```
{
  "reason": "Brief explanation of sources used",
  "answer": "Precise, direct answer",
  "seed_node_ids": ["node_id_1", "node_id_2", ...] // Select 3-5 most relevant IDs
}
```

Table 9: Prompt 4: Graph-Based Reasoning & Answer Generation