MEMORY BANK OF FINITE CAPACITY FOR LLMS BASED MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory mechanisms are the core enabler for large language model based multi-agent systems (MAS) in downstream tasks. Existing works enhance the effectiveness of memory guided MAS through the integration and updating of memory for adaptation to constantly evolving tasks. Nevertheless, these strategies incur higher consumption of storage space, time, and token resources, which constrains the efficacy of memory mechanisms in long-term MAS operation. To solve these drawbacks, this paper proposes FiniteMem, a finite memory mechanism based on multi-dimensional memory replay for MAS. It builds a three-tier memory architecture to supply both macro and micro guidance, and designs the memory replay based on memory competition and inhibitory control, to substantially reduce resource consumption while improve the memory mechanism adaptability. Experiments on multiple publicly available datasets and MAS frameworks verify that FiniteMem achieves state-of-the-art performance, reducing storage space by 31.3%, token consumption by 26.9%, and runtime by 11.8%. FiniteMem offers a universal, high efficiency long-term memory management solution for balancing performance and resource overhead of MAS. Our code will be available at https://anonymous.4open.science/r/FiniteMem-5A3B.

1 INTRODUCTION

With the deepening research on Large Language Models (LLMs), Multi-Agent Systems (MAS) built upon LLMs exhibit unprecedented capabilities in perception Driess et al. (2023); Wei et al. (2024), planning Zhu et al. (2025); Erdogan et al. (2025), reasoning Liang et al. (2024); Zhang et al. (2026b), and action execution Li et al. (2024); Yang et al. (2024). To transform agents from static entities into dynamic ones for adapting to evolving tasks in downstream applications, researchers have designed memory mechanisms that mimic the physiological human knowledge acquisition process and perform continuous adaptive iterative optimization of agents using data generated from agent environment interactions Gao et al. (2025); Hu et al. (2025). These efforts have driven breakthroughs for agents in domains including code generation Qian et al. (2024a), personalized question answering Zhang et al. (2025), education Lv et al. (2025), and healthcare Chen et al. (2025a).

Analogous to human learning, memory updating constitutes the core of memory mechanisms. Early strategies simply append new memories to existing memory banks, disregarding latent dependencies and contradictions of memory entries, which leads to low efficiency in memory utilization Zhao et al. (2024); Park et al. (2023). To address these drawbacks, methods such as Matrix Liu et al. (2024a), G-memory Zhang et al. (2026a), and CAM Li et al. (2026) employ memory consolidation to periodically distill raw memory data into structured knowledge. Nevertheless, memory consolidation cannot eliminate erroneous and invalid memories within memory banks, prompting researchers to develop memory updating paradigms. One paradigm resolves internal memory conflicts and revises inconsistent records, as exemplified by MemGPT Packer et al. (2023) and Mem-α Wang et al. (2025). The other paradigm builds upon forgetting mechanisms, which actively purge outdated, redundant, or unreliable information via evaluation metrics including recency Jiang et al. (2026), access frequency Kang et al. (2025), and importance Yan et al. (2025), to prioritizes core knowledge.

Although these studies have enhanced memory mechanisms, they cannot resolve the persistent growth of the memory bank. Moreover, as MAS operate continuously, the accumulating memories

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.

inevitably incur increasing storage space, time, and token overhead, which prevents memory mechanisms from exerting persistent and stable performance during the long-term execution of MAS. To tackle this challenge, experience replay from continual learning offers a viable solution, which retains a small set of historical samples in memory cache to facilitate learning on new tasks Qiao et al. (2025). Drawing on this concept, we propose FiniteMem, a finite capacity memory mechanism based on multidimensional memory replay to eliminate unbounded memory expansion. Figure 1 illustrates the distinctions between FiniteMem and alternative approaches.

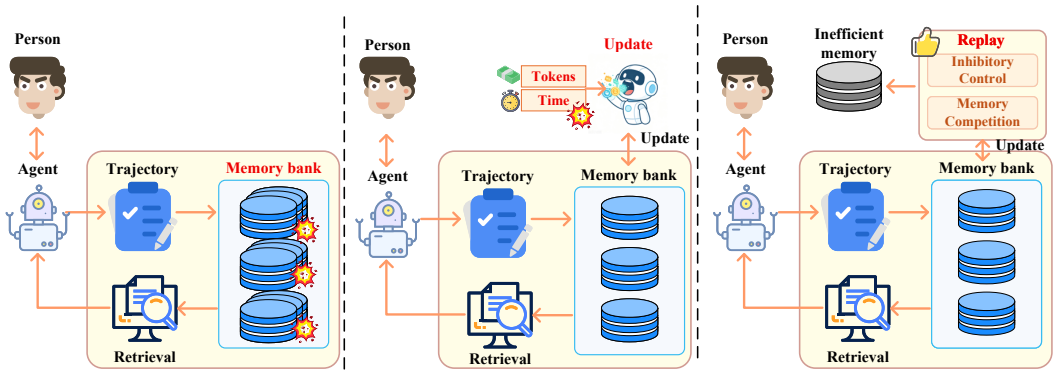

Figure 1: Differences between FiniteMem and other approaches. The left illustrates unbounded memory bank expansion; the middle illustrates massive token and time overhead; the right shows that FiniteMem eliminates inefficient memories without extra consumption.

To enhance the adaptability of FiniteMem across diverse MAS frameworks, we design a multi-layered memory system decoupled from MAS, which supports fine-grained management and utilization of each agent's memories. Three layers, namely the trajectory layer, knowledge layer, and perspicacity layer, are defined to store the interaction histories of MAS. The trajectory layer restructures agent interaction histories from Markov chains to graphs to excavate latent correlations and information embedded in memories, which further generates knowledge and perspicacity. Subsequently, customized replay algorithms are developed for memory data across different layers. The trajectory layer with fine-grained actions leverages a memory competition based replay algorithm, whereas the knowledge and perspicacity layers encoding high-level semantics adopt an inhibitory control based replay algorithm to dynamically update the finite capacity memory bank. Ultimately, microcosmic directional guidance derived from single step action generation is combined with macroscopic guidance from global reasoning logic and relevant knowledge to boost memory utilization efficiency.

The main contributions of this paper are summarized as follows:

(1) A universal finite capacity memory mechanism named FiniteMem based on multidimensional memory replay is proposed to mitigate the space, token and time overhead of memory mechanisms in the long-term operation of MAS.

(2) A memory replay based on memory competition and inhibitory control are proposed to optimize memory data with diverse hierarchies and structures under low resource consumption.

(3) Extensive experiments conducted on five benchmarks verify that FiniteMem achieves performance gains while substantially cutting memory storage space overhead, as well as reducing token consumption and runtime simultaneously.

2 RELATED WORKS

Memory in Agent System is the core driver for agents to accumulate experience and interact with environments for external exploration. Early agent memory research focused on simple single interaction memory mechanisms including Voyager Wang et al. (2023), MemoryBank Zhong et al. (2024), Generative Agents Park et al. (2023) and MemGPT Packer et al. (2023). As research progresses, agent memory frameworks shift toward brain-inspired cognitive design, extending memory scope from discrete single interactions to cross-episode interactions, such as ExpeL Zhao et al.

(2024) and Synapse Zheng et al. (2024). Meanwhile, memory management schemes evolve from rough text similarity matching to fine-grained knowledge abstraction and experience summarization, demonstrated by AMem Xu et al. (2026), Mem0 Chhikara et al. (2025) and MemInsight Salama et al. (2025). Recently, multi-agent specific memory mechanisms develop rapidly, where ChatDev Qian et al. (2024b), MAD-M^2 Tian et al. (2026) and G-Memory Zhang et al. (2026a) are customized to fit multi-agent system traits. However, existing works mostly pursue performance improvement through complex memory architectures yet overlook optimized memory pool management strategies, which results in unavoidable memory bank bloat under long-term agent running.

Memory evolution primarily maintains the conciseness, consistency, and adaptability of long-term knowledge through the synergistic interaction of memory consolidation and memory updating. Memory integration aims to transform fragmented short-term memories into long-term knowledge; for example, Matrix Liu et al. (2024a) and MOOM Chen et al. (2025b) use fusion methods to compress interaction trajectories and global context, thereby distilling general rules applicable across scenarios. Memory updating strategies focus on the interaction between new information and existing knowledge. One strategy emphasizes revising memories stored in external knowledge bases Packer et al. (2023) and model parameters Fang et al. (2026). Another strategy achieves active forgetting by designing evaluation metrics. XMem Cheng & Schwing (2022) is based on retrieval frequency, while TiM Liu et al. (2023) and MemTool Lumer et al. (2026) rely on the LLM's autonomous judgment. However, these methods lead to continuous growth in token consumption and training costs during long-term operation, posing a challenge for memory evolution.

3 METHODOLOGY

In this section, we introduce FiniteMem as illustrated in Figure 2. FiniteMem first constructs a multi-layer memory system to derive corresponding perspicacity and knowledge. It then runs memory replay to update the memory bank and sustain its efficiency within finite memory capacity.

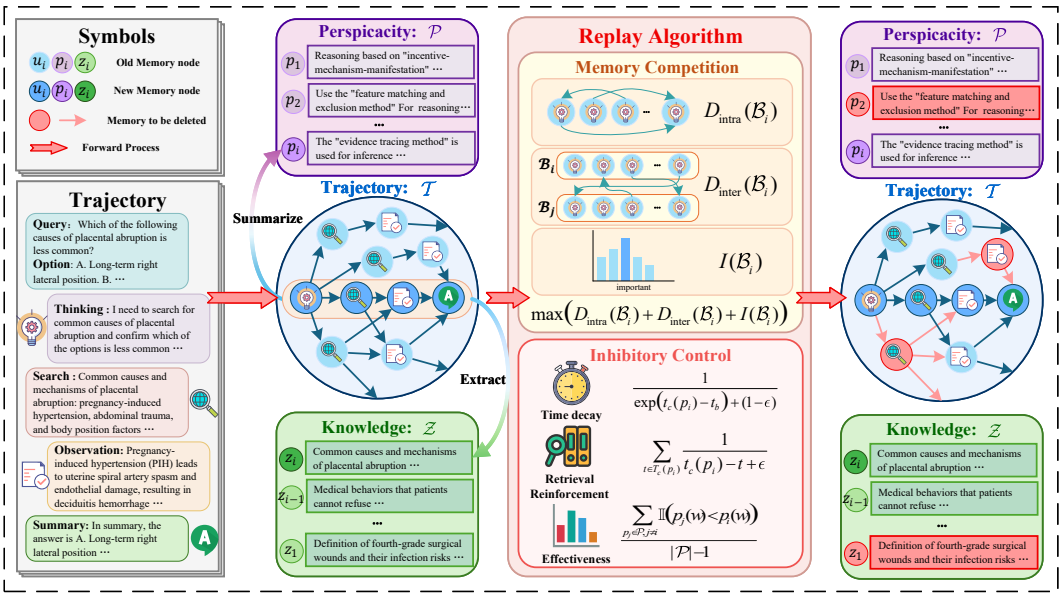

Figure 2: Overview of FiniteMem.

3.1 MULTI-LAYER MEMORY CONSTRUCTION

To maximize effectiveness and adaptability within finite memory capacity, we design a three level multi tier memory mechanism that realizes the decoupling of the memory mechanism from MAS.

Trajectory Layer. Trajectory layer records detailed operation steps of agents to provide fine-grained microscopic guidance for subsequent action generation. Prior research treats each full trajectory τ_q

as an independent storage unit Zhang et al. (2026a); Chhikara et al. (2025), which wastes storage and ignores implicit memory correlations. To solve this problem, we split trajectories into per-agent v_i within system G and build trajectory layer $\mathcal{T} = (U, E_U)$. Vertex set U contains all interactive steps $u_i = (q_i, v_i, r_i)$ of each agent for query q; edge set $E_U = \{\varepsilon_{ij} = \rho(u_i, u_j) = \varepsilon_{ji}\}$ quantifies pairwise step relevance.

Only trajectories τ_q that correctly answer query q are input to the trajectory layer. Following topological order π, we find each step u_i's predecessor $Pred(u_i)$ and successor $Succ(u_i)$, assign default correlation weight d as ε^p_{ii} and ε^s_{ii}, then store u_i into the trajectory bank. To fully exploit latent information among memories under finite memory capacity, we retrieve similar nodes U_r for every u_i via Equation 1:

$$U_r = \left\{ u_j \,\middle|\, sim(u_i, u_j) > \theta_u,\ v_j = v_i,\ q_j \neq q_i,\ j = 1, 2, \ldots, K_t \right\}, \tag{1}$$

where K_t is the retrieval quantity, θ_u the trajectory layer similarity threshold. If matching nodes are fewer than K_t, K_t equals the actual qualified count. The $q_j \neq q_i$ filter excludes same trajectory interference to ensure retrieved nodes are from different trajectories. Semantic similarity $sim(u_i, u_j)$ is calculated via Equation 2, where $Emb(u_j)$ denotes vector representation of step u_j.

$$sim(u_i, u_j) = \frac{Emb(u_i)^T Emb(u_j)}{\|Emb(u_i)\| \|Emb(u_j)\|}. \tag{2}$$

Analogous steps share potential linkages. Hence, we extract all $\mathrm{Pred}(u_j)$ and $\mathrm{Succ}(u_j)$ of retrieved u_j from π, and set default weight d for cross-node correlations ε^p_{ij} (between u_i and $\mathrm{Pred}(u_j)$) and ε^s_{ij} (between u_i and $\mathrm{Succ}(u_j)$). This converts original Markov-chain trajectory τ_q into a graph structure. Such graph modeling unlocks implicit memory connections, enriches candidate actions for downstream generation, and boosts the trajectory bank's information density under finite storage space.

Perspicacity Layer and Knowledge Layer. Trajectory memory provides fine-grained guidance for agents but lacks holistic guidance and supplementary knowledge. Prior work takes the full trajectory τ_q as context, where high-level information remains implicit. As the trajectory layer decomposes τ_q, we build a perspicacity layer and a knowledge layer to exploit this latent information. The perspicacity layer is defined as $\mathcal{P} = \{p \mid p = (q, r_p, w)\}$, where w denotes the validity score of each memory entry. After constructing the trajectory layer for τ_q, the query q and full trajectory are fed into the perspicacity agent v_p. This agent extracts reasoning logic, thought patterns, and error lessons from τ_q, encapsulating them into r_p to supply global reasoning guidance for new tasks. The knowledge layer $\mathcal{Z} = \{z \mid z = (q, r_z, w)\}$ shares the same structure. The knowledge extraction agent v_z extracts domain knowledge r_z from τ_q to fill agents' knowledge gaps. r_p and r_z are computed via Equation 2, with w initialized to the default value d. The prompts for v_p and v_z are shown in Appendix D.

Unlike agent-specific trajectory memories u_i, reasoning logic and knowledge extracted from different τ_q can overlap. Given limited memory bank capacity, we compute the semantic similarity between newly generated r_p, r_z and existing entries in their respective memory banks using Equation 2 before storage to remove redundancy. If the similarity exceeds threshold θ_p or θ_z, the new record is discarded as a duplicate memory entry.

3.2 Memory Replay Based on Memory Competition

The multi tier memory system of FiniteMem possesses distinct data structures and knowledge densities. To accommodate finite memory capacity and better fit the characteristics of multi level memories, we devise differentiated memory replay strategies. The trajectory layer contains detailed information regarding MAS execution and stores correlations among memories within the trajectory graph. To construct a high quality trajectory layer under constrained memory capacity, we introduce the memory competition concept from cognitive science, which accounts for human forgetting Lindsay & Norman (2013). Taking memory diversity and importance as competition criteria, a greedy algorithm is adopted for memory replay to reduce resource consumption.

Memory Competition. Three metrics are proposed to fully reflect the coverage range of memory entries: internal diversity, external diversity and importance. Internal diversity measures dissimilarity among samples inside the memory bank. Since grouping the whole trajectory graph $\mathcal{T}$ as one cluster would force memories into a narrow representation space, we split $\mathcal{T}$ into $K_{\mathcal{T}}$ disjoint memory clusters $\mathcal{B}_i$ and introduce external diversity to calculate inter-cluster node distances, broadening the overall representational scope and improving total memory diversity. The importance metric assesses each memory's practical value to agents. The updated memory cluster $\mathcal{B}'_i$ after replay satisfies the following criteria:

$$F(\mathcal{B}'_i) = \max\left(D_{intra}(\mathcal{B}'_i) + D_{inter}(\mathcal{B}'_i) + I(\mathcal{B}'_i)\right), \tag{3}$$

where $D_{intra}(\mathcal{B}_i)$ is internal diversity, computed as the sum of each sample's minimum Euclidean distance to other samples within cluster $\mathcal{B}_i$. $D_{inter}(\mathcal{B}_i)$ denotes external diversity. For any sample ψ, we compute its distance to $\mathcal{B}_i$ and average distances to all other clusters. $I(\mathcal{B}_i)$ means memory importance, quantified via normalized ranking scores based on each node's total connected correlation weights ε_{ij}. The calculation Equations for the above metrics are presented as follows, where I denote the index set of nodes contained, i.e., $I_b = \{i \mid u_i \in \mathcal{B}_i\}$ and $I_u = \{i \mid u_i \in U\}$.

$$D_{\text{intra}}(\mathcal{B}_i) = \sum_{\phi \in \mathcal{B}_i} \min_{\psi \in \mathcal{B}_i, \phi \neq \psi} \|\text{Emb}(\phi) - \text{Emb}(\psi)\|_2, \tag{4}$$

$$D_{\text{inter}}(\mathcal{B}_i) = \frac{1}{K_{\mathcal{T}} - 1} \sum_{\phi \in \mathcal{B}_i} \left(\sum_{\substack{j=1 \\ j \neq i}}^{K_{\mathcal{T}}} \min_{\psi \in \mathcal{B}_j} \left\|\text{Emb}(\phi) - \text{Emb}(\psi)\right\|_2 \right), \tag{5}$$

$$I(\mathcal{B}_i) = \sum_{j \in I_b} \frac{1}{|U|} \sum_{x \in I_u} \mathbb{I}\left(\sum_{\substack{k=1 \\ k \neq x}}^{|U|} \varepsilon_{xk} < \sum_{\substack{k=1 \\ k \neq j}}^{|U|} \varepsilon_{jk} \right), \tag{6}$$

$$\mathbb{I}(a < b) = \begin{cases} 1, & a < b, \\ 0, & a \geq b. \end{cases} \tag{7}$$

Greedy Algorithm. When the number of nodes in the trajectory layer first reaches the threshold $N_{\mathcal{T}}$, all nodes are evenly partitioned into $K_{\mathcal{T}}$ memory clusters $\mathcal{B}_i$ via K-means to construct initial clusters with low computational overhead. All new nodes generated by a τ_q form a new memory cluster $\mathcal{B}_{new}$, as shown in Equations 8, 9, 10, and a greedy filling strategy is applied to update memory clusters. We initialize the memory set first, to prevent excessive similarity to existing clusters, the memory point in $\mathcal{B}_{new}$ farthest from the buffer centers of all old clusters is selected as the initial element u_1, which forms the preliminary updated cluster $\mathcal{B}'_i = \{u_1\}$. For each unselected sample u^* from $\mathcal{B}_i$ and $\mathcal{B}_{new}$, we compute the marginal gain introduced by incorporating u^*. The sample delivering the maximum marginal gain is picked and appended to $\mathcal{B}'_i$ until termination conditions are met. This replay procedure is iteratively executed for all $K_{\mathcal{T}}$ memory clusters $\mathcal{B}_i$, and all unselected samples are discarded afterwards. We assign the remainder of $\frac{N_{\mathcal{T}}}{K_{\mathcal{T}}}$ to the first memory cluster.

$$u_1 = \arg\max_{u \in \mathcal{B}_{\text{new}}} \min_{j=1,\ldots,K_{\mathcal{T}}} \left\| \text{Emb}(u) - \frac{1}{|\mathcal{B}_j|} \sum_{\psi \in \mathcal{B}_j} \text{Emb}(\psi) \right\|_2, \tag{8}$$

$$u^* = \arg\max_{u \notin \mathcal{B}'_i} \left(F(\mathcal{B}'_i \cup \{u\}) - F(\mathcal{B}'_i)\right), \tag{9}$$

$$\mathcal{B}'_i \leftarrow \mathcal{B}'_i \cup \{u^*\}, \text{ until } |\mathcal{B}'_i| = \left\lfloor \frac{N_{\mathcal{T}}}{K_{\mathcal{T}}} \right\rfloor. \tag{10}$$

3.3 Memory Replay Based on Inhibitory Control

The knowledge layer and perspicacity layer store high-level abstract semantics distilled from trajectories by LLMs. Unlike the trajectory layer, they differ in text information density and data structure, requiring dedicated updating strategies. Inspired by the inhibitory control theory in cognitive science Mensink & Raaijmakers (1988), we propose a replay algorithm based on inhibitory control.

It evaluates memory value via three metrics: temporal decay, retrieval reinforcement, and validity. Low-value memories are suppressed and eliminated, equivalent to inhibiting their activation pathways in the MAS. Specifically, once memory counts in the perspicacity layer and knowledge layer reach their respective thresholds $K_{\mathcal{P}}$ and $K_{\mathcal{Z}}$, each memory's value is defined by:

$$S(p_i) = \alpha \frac{1}{\exp\left(t_c(p_i) - t_b\right) + (1-\epsilon)} + \beta \sum_{t \in T_c(p_i)} \frac{1}{t_c(p_i) - t + \epsilon} + \gamma s_e(p), \tag{11}$$

$$s_e(p_i) = \frac{1}{|\mathcal{P}| - 1} \sum_{p_j \in \mathcal{P}, j \neq i} \mathbb{I}\big(p_j(w) < p_i(w)\big). \tag{12}$$

where t_c denotes the current running round of the agent; t_b represents the round when the memory was created; T_c stands for the set of all historical rounds where this memory was retrieved. Since excessively remote retrieval histories exert negligible impacts on $S(p_i)$, only the latest 10 rounds are retained in T_c. α, β, and γ are weight coefficients to balance the contribution of each metric; ϵ denotes an infinitesimal positive constant to avoid division by zero; s_e refers to the validity score of the memory.

We define the first component in memory value calculation as the temporal decay metric. Referring to memory decay theory Ebbinghaus (2013), we argue that for agents, earlier insights can be superseded by later memories. The second component is the retrieval reinforcement metric. Frequently retrieved memories inherently possess generalizability, and elevating their value conforms to the "use it or lose it" principle. The final component quantifies memory validity using normalized rank scores. Ultimately, as in Equation13, we compute score S and retain $K_{\mathcal{P}}$ or $K_{\mathcal{Z}}$ memories to construct $\mathcal{P}'$ and $\mathcal{Z}'$, enabling efficient dynamic updates under fixed capacity constraints.

$$\mathcal{P}' = \underset{\substack{\mathcal{P}' \subseteq \mathcal{P}, \\ |\mathcal{P}'| = K_{\mathcal{P}}}}{\arg\max} \sum_{p_i \in \mathcal{P}'} S(p_i). \tag{13}$$

3.4 Memory Retrieval and Reasoning

Given a new input query q_{new}, we first retrieve $K_r^{\mathcal{P}}$ and $K_r^{\mathcal{Z}}$ memory entries from the perspicacity layer $\mathcal{P}$ and knowledge layer $\mathcal{Z}$ via semantic similarity, to provide macroscopic auxiliary information for agent reasoning. Before agent v_i executes the t-th round of actions, we retrieve the top-$K_r^{\mathcal{T}}$ relevant nodes M from trajectory graph $\mathcal{T}$ belonging to v_i, which are strongly correlated with q_{new} and the prior output $r_i^{(t-1)}$. All successor nodes of M are extracted as microscopic references to guide the t-th agent execution, with the computation defined in Equation14. Where K_r is the number of retrieved memories; N is the total number of stored memories; b denotes the text for similarity computation with memory r_i in the memory bank: $b = q_{\text{new}}$ for the perspicacity and knowledge layers, while $b = r_i^{(t-1)}$ for the trajectory layer. α_r and β_r are weights balancing different semantic similarity terms. After reasoning, the correlation coefficient ε or validity score w of retrieved memories increases if the agent yields positive feedback; otherwise, these values decrease.

$$M = \arg\max_{|M| = K_r} \sum_{i=0}^{N} \big((1-\alpha_r)\text{sim}(q_{\text{new}}, q_i) + \alpha_r \text{sim}(b, r_i)\big). \tag{14}$$

4 EXPERIMENTS

4.1 Experimental Setup

Datasets and Evaluation metrics. To comprehensively evaluate the effectiveness of FiniteMem, we adopt five widely used benchmarks: HotpotQA Yang et al. (2018), FEVER Thorne et al. (2018), AQuA Ling et al. (2017), CMB-Exam Wang et al. (2024) and MedQA Jin et al. (2021), covering multiple tasks and domains including commonsense QA, reasoning, mathematics and medical science, as well as diverse languages. Accuracy (Acc) is employed as the evaluation metric for all datasets. Detailed statistics are provided in Appendix B.1.

Table 1: Performance Comparison with Single/Multi-Agent Memory Architectures on Five Benchmarks

MAS	Memory	HotpotQA	FEVER	AQuA	CMB-Exam	MedQA	Avg.
AutoGen	non-memory	26.93	53.48	65.44	75.82	82.98	60.93
	Voyager	29.68	58.86	71.91	76.48	84.21	64.23
	MemoryBank	31.26	57.32	67.87	75.01	82.70	62.83
	Generative	29.60	55.81	71.48	76.51	83.95	63.47
	ChatDev	22.44	56.62	62.46	76.04	83.48	60.21
	MAD-M^2	33.28	61.81	75.97	76.84	84.31	66.44
	G-Memory	33.94	61.19	74.58	77.38	84.54	66.33
	FiniteMem	**34.19**	61.03	**76.19**	**79.15**	**85.53**	**67.22**
DyLAN	non-memory	29.13	56.22	66.12	76.45	82.65	62.11
	Voyager	30.33	57.55	73.02	78.01	84.72	64.73
	MemoryBank	27.81	54.63	60.77	71.77	80.19	59.03
	Generative	27.32	57.53	72.76	78.73	82.89	63.85
	ChatDev	20.44	55.28	58.30	70.88	81.09	57.20
	MAD-M^2	31.27	59.80	73.97	79.87	84.68	65.92
	G-Memory	32.14	60.16	74.53	79.43	84.73	66.20
	FiniteMem	**33.31**	**60.89**	**76.68**	**81.83**	**85.83**	**67.71**
MacNet	non-memory	27.02	56.54	67.58	76.54	82.99	62.13
	Voyager	31.03	58.49	73.31	79.02	85.08	65.39
	MemoryBank	31.76	56.98	66.39	75.29	83.22	62.73
	Generative	29.26	58.47	68.88	75.50	83.89	63.60
	ChatDev	27.86	56.16	68.34	78.14	81.31	62.36
	MAD-M^2	32.27	60.74	74.58	79.72	85.20	66.50
	G-Memory	32.75	60.35	74.01	80.42	84.32	66.37
	FiniteMem	**33.27**	**60.80**	**75.84**	**80.99**	**86.03**	**67.39**

Table 2: Ablation study on FiniteMem.

MAS	Method	HotpotQA	FEVER	AQuA	CMB-Exam	MedQA	Avg.
AutoGen	w/o $\mathcal{T}$	31.38	58.73	73.95	75.92	82.49	64.49
	w/o $\mathcal{P}$	33.23	60.86	75.80	78.17	84.64	66.54
	w/o $\mathcal{Z}$	32.94	59.56	74.29	77.58	83.91	65.66
	w/o $\mathcal{T}R$	32.76	59.93	74.69	77.78	84.37	65.91
	w/o $\mathcal{P}R$	33.21	60.33	75.15	78.58	84.60	66.37
	w/o $\mathcal{Z}R$	33.18	60.40	75.07	78.52	84.76	66.39

Baselines. To further demonstrate the superiority of FiniteMem, we select 7 baselines without memory capacity constraints, including four single-agent memory baselines: non-memory, Voyager Wang et al. (2023), MemoryBank Zhong et al. (2024) and Generative Agents Park et al. (2023), alongside three multi-agent memory implementations from ChatDev Qian et al. (2024b), MAD-M^2 Tian et al. (2026) and G-Memory Zhang et al. (2026a). Full details are listed in Appendix B.2.

Implementation details. We integrate FiniteMem and all baselines with three representative multi-agent frameworks: AutoGen Wu et al. (2023), DyLAN Liu et al. (2024b), and MacNet Qian et al. (2025). The open-source large language model Qwen3-32B Yang et al. (2025) instantiates these MAS frameworks, and BGE-large-v1.5 Xiao et al. (2024) serves as the encoding function. Notably, we cache memory embeddings at runtime for replay and retrieval to improve FiniteMem's computational efficiency. All experiments are repeated three times, and average results are reported. Further details are listed in Appendix B.3 and B.4, with additional experimental analyses in Appendix C.

4.2 Main results

Table1 fully compares the performance of various memory mechanisms on three widly-used MAS frameworks. Even with constrained memory capacity, FiniteMem boosts performance across all tasks and MAS setups, outperforming the optimal single-agent and multi-agent memory baselines by average margins of 4.10% and 1.71%. It especially attains substantial gains on professional medical datasets CMB and MedQA, as its replay algorithm preserves clinically salient medical

knowledge, diagnostic reasoning and operational procedures to deliver comprehensive macro and micro guidance for problem solving. Meanwhile, benchmarks including MemoryBank and ChatDev demonstrate that intricate memory structures do not guarantee better MAS performance. Overly complex memory systems store excessive low value content, without efficient memory update rules, such useless memories create misleading signals and degrade overall results. These results highlight the significant value of the multi-level memory bank and the finite memory capacity, which are the two core features of FiniteMem.

4.3 Ablation Studies

Table2 reports ablation experiments of FiniteMem. Two groups of ablations are conducted: removing each component of the multi-level memory system individually (w/o $\mathcal{T}$, w/o $\mathcal{P}$, w/o $\mathcal{Z}$), and disabling the replay algorithm of each layer separately (w/o $\mathcal{T}R$, w/o $\mathcal{P}R$, w/o $\mathcal{Z}R$). All component removals cause steady performance degradation. The trajectory layer $\mathcal{T}$ exerts the strongest influence, its elimination reduces performance by 4.23%. As the core provider of fine-grained micro guidance for agents, removing $\mathcal{T}$ or receiving low quality action instructions impairs overall results; the perspicacity and knowledge layers only offer high level macro guidance and cannot substitute $\mathcal{T}$. Accordingly, disabling the trajectory layer's replay algorithm triggers a larger performance drop than shutting down other layer replay modules. The ablation outcomes prove that both the multi-level memory architecture and hierarchical replay algorithms jointly boost model performance.

4.4 Further Analysis

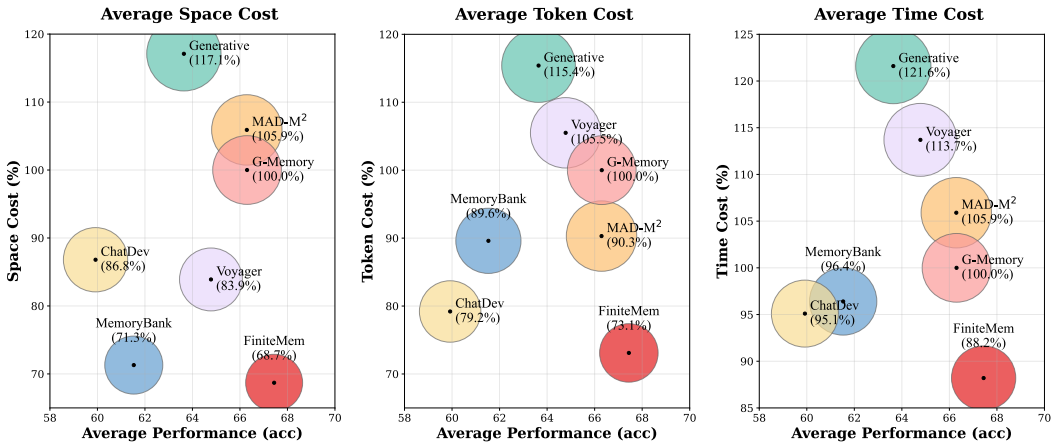

Figure 3: Cost analysis. The horizontal axis represents the average performance of each method across all datasets and MAS configurations.

Cost Analysis. We measured token, space and time overhead of all methods to verify FiniteMem's resource efficiency, taking the optimal baseline G-Memory as reference to compute average resource consumption ratios, with results illustrated in Figure3. FiniteMem cuts all three resource costs notably, including a 31.3% storage space reduction. Though memories are stored as vectors, its bounded capacity memory bank offsets extra vector space overhead. For tokens, FiniteMem avoids LLM driven memory updates and full trajectory text input, slashing token usage by 26.9%. While the replay algorithm adds minor computation load, vectorized memory space accelerates runtime. Baseline memory banks keep expanding with agent iterations, slowing retrieval and lengthening per-round runtime continuously. By contrast, FiniteMem's computation time plateaus once its memory hits capacity, delivering an overall 11.8% time saving.

Influence of Memory Bank Capacity. Figure4 shows how memory capacity affects model performance. For the trajectory bank, insufficient capacity discards valid memories, while excessive capacity introduces noise. Both degrade single-step reasoning, increase interaction rounds and reasoning text, and raise resource overhead. The knowledge and perspicacity banks exhibit similar trends under capacity reduction, though their high-level guidance makes performance far less

volatile. Greatly expanding memory volume improves high-level matching accuracy and reduces redundant interactions, yet it brings negligible performance gains with heavy storage space costs. We adopt fixed capacity limits: $N_{\mathcal{P}} = 100$, $N_{\mathcal{Z}} = N_{\mathcal{T}} = 500$. With this setup, FiniteMem achieves an average score of 65.53 across all datasets and MAS configurations, only 0.77 below the state-of-the-art baseline G-Memory, and reduces space complexity from $O(N)$ to $O(1)$. Thus, FiniteMem has clear advantages and broad practical value for storage-constrained deployment scenarios.

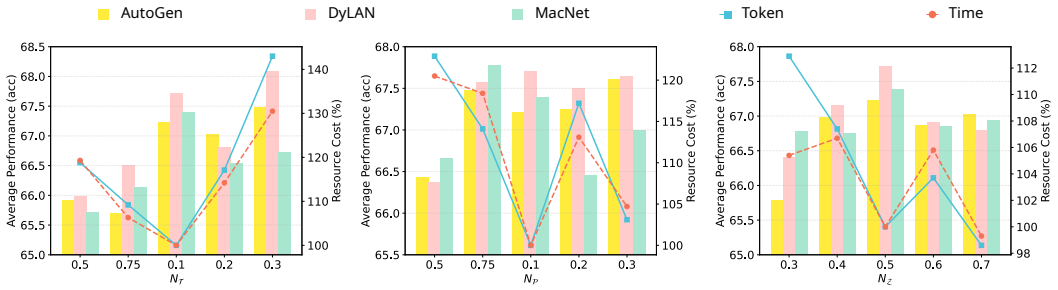

Figure 4: where N denotes the ratio relative to the data quantity $|D|$.

Influence of Agent Running Rounds. Using the MedQA, we analyze the performance and resource consumption of FiniteMem, G-Memory, and MAD-M^2 across the full execution cycle. As shown in Figure5, FiniteMem underperforms the baselines in early execution , yet its performance improves substantially through memory updates in later stages. For resource consumption, the baseline's average runtime increases linearly with agent rounds, as expanding memory banks increase retrieval overhead. In contrast, FiniteMem's average runtime stays nearly constant after hitting its memory capacity limit. All methods exhibit stable token consumption, which depends only on MAS performance, degraded MAS performance triggers longer reasoning texts or extra steps, thereby increasing resource usage. Overall, FiniteMem shows growing advantages in both performance and resource efficiency as the MAS execution rounds increase.

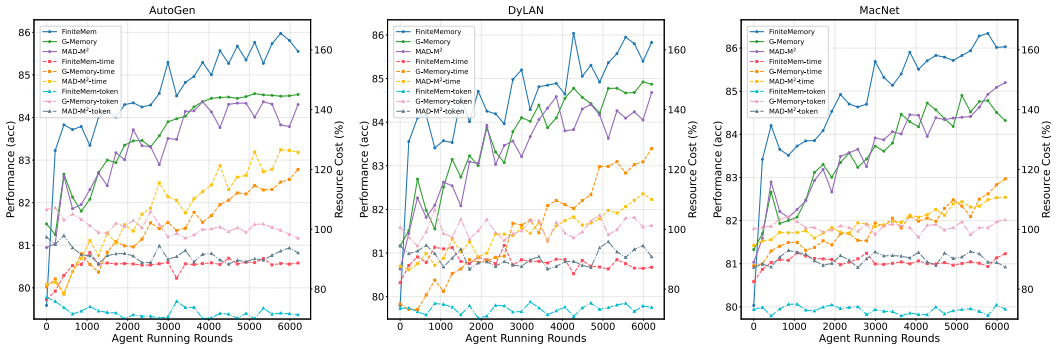

Figure 5: Influence of Agent Running Rounds. Using G-Memory as reference to compute average resource consumption ratios.

5 Conclusion

To tackle the efficacy of memory mechanisms in long-term MAS operation, we propose FiniteMem, a finite memory mechanism based on multidimensional representation. Its multi-layer memory system extracts task-critical memory information, while two replay algorithms enable efficient memory updates under fixed capacity without extra resource overhead. Extensive experiments verify that FiniteMem can be natively embedded into mainstream MAS architectures and greatly boost their autonomous evolution ability. Still, FiniteMem has room for optimization in runtime cost and algorithm stability, which will be explored in future work.

AI USE STATEMENT

Generative AI tools were used during manuscript preparation for grammar polishing and wording refinement. No generative AI was used for research idea formulation, method design, experimental implementation, data analysis, or deriving the core conclusions of this work. All text and results were reviewed and verified manually by the authors.

ETHICS STATEMENT

This work to tackle the efficacy of memory mechanisms in long-term MAS operation. Our experiments are conducted entirely on publicly available, community-standard multi-agent simulation benchmarks. No human subjects, personally identifiable information, private or sensitive datasets are collected, processed, or released in this study. All data used for model training and evaluation have been previously published with open access and have been widely adopted within the multi-agent research community.

The proposed approach aims to improve resource efficiency without modifying task objectives or introducing biased, discriminatory, or harmful agent behaviors. The method is developed solely for academic research and does not target high-risk real-world deployment. We have carefully checked the experimental setup to avoid unintended unfair outcomes. There are no conflicts of interest, privacy concerns, or legal compliance issues associated with this work. All experiments adhere to standard academic research integrity guidelines.

REFERENCES

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5): 679–684, 1957.

Kai Chen, Ji Qi, Jing Huo, Pinzhuo Tian, Fanyu Meng, Xi Yang, and Yang Gao. A self-evolving framework for multi-agent medical consultation based on large language models. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025a.

Shuze Chen and Tianyi Peng. Multi-agent markov entanglement. *Advances in Neural Information Processing Systems*, 38:9826–9865, 2026.

Weishu Chen, Jinyi Tang, Zhouhui Hou, Shihao Han, Mingjie Zhan, Zhiyuan Huang, Delong Liu, Jiawei Guo, Zhicheng Zhao, and Fei Su. Moom: Maintenance, organization and optimization of memory in ultra-long role-playing dialogues, 2025b. URL https://arxiv.org/abs/2509.11860.

Ho Kei Cheng and Alexander G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), *Computer Vision – ECCV 2022*, pp. 640–658, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19815-1.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Hermann Ebbinghaus. [image] memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4):155, 2013.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. *arXiv preprint arXiv:2503.09572*, 2025.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. Lightmem: Lightweight and efficient memory-augmented generation. In *International Conference on Learning Representations*, volume 2026, pp. 98706–98729, 2026.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.

Tao Jiang, Zichuan Lin, Lihe Li, Yi-Chen Li, Cong Guan, Lei Yuan, Zongzhang Zhang, Yang Yu, and Deheng Ye. Multi-agent in-context coordination via decentralized memory retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 22363–22371, 2026.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14), 2021. ISSN 2076-3417. doi: 10.3390/app11146421. URL https://www.mdpi.com/2076-3417/11/14/6421.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25972–25981, 2025.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li E Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.

Rui Li, Zeyu Zhang, Xiaohe Bo, Zihang Tian, Xu Chen, Quanyu Dai, Zhenhua Dong, and Ruiming Tang. Cam: A constructivist view of agentic memory for llm-based reading comprehension. *Advances in Neural Information Processing Systems*, 38:113381–113406, 2026.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 17889–17904, 2024.

Peter H Lindsay and Donald A Norman. *Human information processing: An introduction to psychology*. Academic press, 2013.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 158–167, 2017.

Jiale Liu, Yifan Zeng, Malte Højmark-Bertelsen, Marie Normann Gadeberg, Huazheng Wang, and Qingyun Wu. Memory-augmented agent training for business document understanding. In *AI Agent for Information Retrieval Workshop*, pp. 39–53. Springer, 2024a.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. Think-in-memory: Recalling and post-thinking enable llms with long-term memory, 2023. URL https://arxiv.org/abs/2311.08719.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration, 2024b. URL https://arxiv.org/abs/2310.02170.

Elias Lumer, Anmol Gulati, Vamse Kumar Subbiah, Pradeep Honaganahalli Basavaraju, and James A. Burke. Memtool: Optimizing short-term memory management for dynamic tool retrieval and invocation in llm agent multi-turn conversations. In Ricardo Campos, Adam Jatowt, Yanyan Lan, Mohammad Aliannejadi, Christine Bauer, Sean MacAvaney, Avishek Anand, Zhaochun Ren, Suzan Verberne, Nan Bai, and Masoud Mansoury (eds.), *Advances in Information Retrieval*, pp. 237–253, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-21300-6.

Rui Lv, Qi Liu, Weibo Gao, Haotian Zhang, Junyu Lu, and Linbo Zhu. Genal: Generative agent for adaptive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 577–585, 2025.

Ger-Jan Mensink and Jeroen G Raaijmakers. A model for interference and forgetting. *Psychological Review*, 95(4):434, 1988.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Chen Qian, Yufan Dang, Jiahao Li, Wei Liu, Zihao Xie, Yifei Wang, Weize Chen, Cheng Yang, Xin Cong, Xiaoyin Che, et al. Experiential co-learning of software-developing agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5628–5640, 2024a.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large language model-based multi-agent collaboration. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 41488–41505, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/66a026c0d17040889b50f0dfa650e5e0-Paper-Conference.pdf.

Ziyue Qiao, Junren Xiao, Qingqiang Sun, Meng Xiao, Xiao Luo, and Hui Xiong. Towards continuous reuse of graph models via holistic memory diversification. In *International Conference on Learning Representations*, volume 2025, pp. 73190–73211, 2025.

Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. Meminsight: Autonomous memory augmentation for llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33124–33140, 2025.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, 2018.

Hongduan Tian, Xiao Feng, Ziyuan Zhao, Rolan Yan, and Bo Han. Multi-agent debate with memory masking. In *International Conference on Learning Representations*, volume 2026, pp. 5122–5143, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. In *Intrinsically-Motivated and Open-Ended Learning Workshop @NeurIPS2023*, 2023. URL https://openreview.net/forum?id=nfx5IutEed.

Xidong Wang, Hardy Chen, Dingjie Song, Zhiyi Zhang, Zhihong Chen, Qingying Xiao, Feng Jiang, Jianquan Li, Xiang Wan, Benyou Wang, et al. Cmb: A comprehensive medical benchmark in chinese. In *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)*, pp. 6184–6205, 2024.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-{\alpha}: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025.

Yuxi Wei, Zi Wang, Yifan Lu, Chenxin Xu, Changxing Liu, Hao Zhao, Siheng Chen, and Yanfeng Wang. Editable scene simulation for autonomous driving via collaborative llm-agents. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15077–15087. IEEE, 2024.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 641–649, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Cilin Yan, Jingyun Wang, Lin Zhang, Ruihui Zhao, Xiaopu Wu, Kai Xiong, Qingsong Liu, Guoliang Kang, and Yangyang Kang. Efficient and accurate prompt optimization: the benefit of memory in exemplar-guided reflection. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 753–779, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yijun Yang, Tianyi Zhou, Kanxue Li, Dapeng Tao, Lusong Li, Li Shen, Xiaodong He, Jing Jiang, and Yuhui Shi. Embodied multi-modal agent trained by an llm from a parallel textworld. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26265–26275. IEEE, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *Advances in Neural Information Processing Systems*, 38:12988–13018, 2026a.

Xinliang Frederick Zhang, Nick Beauchamp, and Lu Wang. Prime: Large language model personalization with cognitive dual-memory and personalized thought process. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33695–33724, 2025.

Yuxuan Zhang, Jianzhou Feng, Yiming Xu, Ziqi Wang, Tianyu Yang, and Xiaohuan Wang. Emrdcm: An experience-enhanced multi-role debate method based on compromise mechanisms. *Journal of Intelligent Information Systems*, 64(3):1305–1327, 2026b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 19632–19642, 2024.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 19036–19066, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/52f050499cf82fa8efb588e263f6f3a7-Paper-Conference.pdf.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 19724–19731, 2024.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. Knowagent: Knowledge-augmented planning for llm-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3709–3732, 2025.

A Agent System Formalization

We define an agent system as a directed graph $G = (V, E)$. The size of vertex set $|V| = N$ equals the number of agents in the system, and edge set $E \subseteq V \times V$ denotes interaction links between agents. For any agent $v_i \in V$ in G, we define it via a triple:

$$v_i = (Model_i, Role_i, Tool_i), \tag{15}$$

where $Model_i$ denotes the underlying LLM instance the agent relies on; $Role_i$ stands for the assigned role configuration of the agent; $Tool_i$ refers to the set of tools (e.g., web search) accessible to this agent.

After the system receives a user query q, it generates a topological order $\pi = [\pi_1, \pi_2, \cdots, \pi_N]$ for all nodes over t interaction rounds. A directed edge from π_i to π_j implies $i < j$. This constraint ensures each agent executes its own task only after all predecessor agents finish computation. For agent v_i in the topological sequence, its output at round t is as follows:

$$r_i^{(t)} = v_i\big(Role_i, q, Mem_i, \{r_h^{(t)} | v_h \in \mathcal{N}^-(v_i)\}\big), \tag{16}$$

where $r_i^{(t)}$ denotes the response generated by agent v_i at round t, Mem_i represents the memories utilized to produce the response, and $\mathcal{N}^-(v_i)$ is the set of in-neighbor nodes of v_i; outputs from neighboring nodes serve as contextual inputs for v_i. Prior studies Bellman (1957); Chen & Peng (2026) have demonstrated that this process satisfies the Markov property. Let the agent state at round t be $S_t = (q, Role_t, Mem_t)$:

$$P\left(r_i^{(t+1)} | S_t, r_i^{(t)}, S_{t-1}, r_i^{(t-1)}, \cdots, S_0, r_i^{(0)}\right) = P\left(r_i^{(t+1)} | S_t, r_i^{(t)}\right). \tag{17}$$

The predefined iteration upper limit is reached or an early termination condition is triggered after T rounds of interaction, yielding the final output for query q. All interaction records generated during execution are defined as a trajectory τ_q. We construct the multi-layer memory based on trajectory τ.

B Experimental Details

B.1 Dataset Descriptions

In this section, we introduce the datasets adopted in our experiments. The statistics of these datasets are summarized in Table3.

HotpotQA Yang et al. (2018). This Wikipedia-based question answering dataset requires models to perform multi-step reasoning across multiple documents, with diverse question types unbound by fixed knowledge bases. It provides sentence-level supporting facts to supervise reasoning and result interpretation, and additionally includes comparative fact questions to specifically evaluate models' capabilities of fact extraction and comparative reasoning.

Table 3: The statistics of all datasets

Dataset	Num
HotpotQA	7,405
FEVER	9,999
AQuA	2,801
CMB-Exam	11,200
MedQA	6,112

FEVER Thorne et al. (2018). A knowledge-intensive dataset designed for fact verification tasks, consisting of claims paraphrased from Wikipedia sentences. Each claim is annotated with three labels: SUPPORTS, REFUTES, and NOT ENOUGH INFO, paired with corresponding evidence sentences. All critical evidence required to reach each verdict is fully recorded in the dataset.

AQuA Ling et al. (2017). A mathematical reasoning benchmark for algebraic word problems, containing multiple-choice quantitative math questions spanning various computational scenarios. Each sample is equipped with complete natural-language step-by-step derivations, which can be used to assess models' stepwise reasoning and mathematical explanation skills.

CMB-Exam Wang et al. (2024). A Chinese medical multiple-choice dataset covering six major medical disciplines and 28 subfields, used to measure large language models' medical knowledge reserves and complex clinical reasoning performance.

MedQA Jin et al. (2021). A medical QA dataset formatted as multiple-choice questions, whose test items are sourced from medical licensing examinations in the United States, mainland China and Taiwan, China.

B.2 Baseline Setup

In this section, we elaborate on all baselines adopted for comparison. It is worth noting that although the following methods support memory update operations, none impose a hard upper bound on memory bank capacity, allowing memory entries to accumulate indefinitely.

Voyager Wang et al. (2023). Voyager is the first lifelong learning agent powered by large language models, capable of autonomous exploration, skill acquisition and novel discovery without human intervention. Its architecture consists of three core modules: an automatic exploration curriculum, an elastically expandable code skill library, and an iterative prompting mechanism integrating environmental feedback and self-verification. The stored skills are extensible, interpretable and composable, which steadily boost agent performance and mitigate catastrophic forgetting. Since Voyager's memory design is originally tailored for single-agent settings, we implement agent-specific history retrieval based on each agent's visible dialogue context to adapt it to multi-agent scenarios. Other single-agent memory baselines are modified following the same adaptation paradigm.

MemoryBank Zhong et al. (2024). MemoryBank is a dedicated memory mechanism for large language models, devised to address the inherent limitations of LLMs: the lack of persistent long-term memory and poor user adaptation in continuous interactive workflows. Its core design enables the model to retrieve relevant historical memories and iteratively update memory storage. By aggregating cumulative interaction records, the model gradually perceives and adapts to individual user characteristics. Inspired by the Ebbinghaus forgetting curve, a memory update strategy is integrated to strengthen valuable memories and selectively discard trivial ones according to time decay and memory importance. This human-like memory retention logic sustains long sequential interaction tasks.

Generative Agents Park et al. (2023). Generative Agents maintain two tiers of memory: raw observational memory and high-level reflective memory. The reflective memory captures abstract insights generated by agents through reflection, delivering more structured, conceptualized representations of past experience.

ChatDev Qian et al. (2024b). ChatDev's memory system incorporates intra-trial memory and cross-trial memory. At the start of each round, intra-trial memory is transmitted from the central or initiator agent to furnish guidance derived from prior interactions. The cross-trial memory module adopts a simpler design, storing solutions to historical queries for future retrieval.

MAD-M^2 Tian et al. (2026). MAD-M^2 proposes an innovative memory masking mechanism that allows agents to refine and filter contextual memory at the start of every iteration. The agent first independently scores each memory entry from the previous round, then converts these evaluation scores into a binary mask vector. All entries marked with zero values are discarded, and the agent only utilizes its filtered memory subset $\widehat{\mathcal{M}}_r^{(i)}$. While this method provides a memory filtering pipeline, it does not enforce any size limit on the filtered memory set $\widehat{\mathcal{M}}_r^{(i)}$.

G-Memory Zhang et al. (2026a). Inspired by organizational memory theory, G-Memory is a hierarchical memory system built exclusively for LLM-based multi-agent systems. It establishes a three-layer graph architecture consisting of insight graph, query graph and interaction graph to centrally manage long-term multi-agent interaction records. When processing new user queries, bidirectional memory traversal is performed to simultaneously retrieve generalizable high-level insights and compact fine-grained collaborative trajectories, facilitating cross-task knowledge reuse.

B.3 Multi-agent Systems Setup

In this section, we elaborate on the three multi-agent systems frameworks adopted in our experiments: AutoGen, DyLAN and MacNet.

AutoGen Wu et al. (2023). AutoGen is a mainstream multi-agent coordination framework. Based on its built-in A3 decision-making structure, we construct a cooperative solving pipeline with three dedicated agents: the Solver Agent produces initial solutions under the prompt of a task-solving specialist; the Ground Truth Agent checks its outputs against standard answers to detect biases and logical errors; the Executor Agent transforms verified valid solutions into executable commands. This modular design makes multi-agent collaboration transparent, verifiable and deployable, and stably ensures the reliability and operability of the whole solving process.

DyLAN Liu et al. (2024b). DyLAN is a debate-based multi-agent framework sharing partial structural similarities with LLM-Debate, yet distinguished by a more efficient agent-level early-stopping mechanism embedded within multi-turn interactions. This mechanism cuts redundant computational costs while maintaining sufficient information exchange between agents. Additionally, DyLAN adopts an unsupervised filtering strategy named Agent Importance Score, which identifies agents making the most substantial task contributions to optimize the agent group participating in subsequent rounds. In our implementation, three agents are deployed for the debate module.

MacNet Qian et al. (2025). As a representative work on distributed and scalable multi-agent systems, the core innovation of MacNet lies in abandoning the centralized agent paradigm adopted by conventional architectures and introducing edge agents instead. Edge agents are dynamically invoked during inter-agent communication. Based on outputs from preceding agents, they generate actionable guidance for subsequent agents to enable efficient decentralized collaboration. We retain the original base architecture and prompt templates proposed by MacNet in our reproduction.

B.4 Implementation details

We utilize the bge-large-v1.5 Xiao et al. (2024) model series to implement the embedding function $Emb(\cdot)$ in Equation 2. Specifically, BGE-Large-zh-v1.5 is deployed for Chinese texts, while BGE-Large-en-v1.5 is used for English data. Qwen3-32B Yang et al. (2025) serves as the base large language model for all baselines and multi-agent frameworks. We retain the default hyperparameters of Qwen3-32B, with the temperature parameter solely set to 0.1.

We set the upper bound of memory nodes for each layer as follows: the trajectory layer capacity $N_{\mathcal{T}} = 0.1|V||D|$, the perspicacity layer capacity $N_{\mathcal{P}} = 0.1|D|$, and the knowledge layer capacity $N_{\mathcal{Z}} = 0.5|D|$. Here, $|D|$ denotes the total number of dataset samples, and $|V|$ represents the number of agent roles within the multi-agent systems. The default correlation score for the trajectory layer and the default importance score for both the perspicacity and knowledge layers are fixed at $d = 5$. During trajectory layer construction, the number of retrieved nodes $K_t = 3$ and the similarity

threshold for retrieval $\theta_u = 0.8$. The similarity thresholds for the perspicacity and knowledge layers are set identically to $\theta_p = \theta_z = 0.8$.

In the replay algorithm based on memory competition, the number of memory clusters $\mathcal{B}_i$ is set to $K_{\mathcal{T}} = 5$. Note that $N_{\mathcal{T}} = 0.1|V||D|$. When computing $\frac{N_{\mathcal{T}}}{K_{\mathcal{T}}}$ across different datasets, remainders may occur. We allocate the remainder of $\frac{N_{\mathcal{T}}}{K_{\mathcal{T}}}$ to the first memory cluster to guarantee the stable operation of the replay algorithm.

In the replay algorithm based on inhibitory control, the weights of the temporal decay metric, retrieval reinforcement metric, and validity metric are set to $\alpha = 0.1$, $\beta = 0.5$, and $\gamma = 0.4$, respectively, and an infinitesimal positive constant $\epsilon = 10^{-6}$ is adopted to avoid division by zero. For similarity calculation in memory retrieval, the weight $\alpha_r = 0.8$. The retrieval numbers are configured as $\alpha_r = 0.8$ for the trajectory layer, $K_r^{\mathcal{P}} = 1$, for the perspicacity layer, and $K_r^{\mathcal{Z}} = K_r^{\mathcal{T}} = 5$ for the knowledge layer.

All experiments are repeated three times, and average results are reported. The experimental implementation is built on Python 3.11 and the PyTorch 2.1.0 framework, and all experiments are conducted on an Ubuntu server equipped with eight RTX 3090 GPUs and one Intel(R) Xeon(R) CPU.

Moreover, dedicated smoothing rules are proposed for cold-start scenarios or samples with sparse retrieval histories in the replay algorithm based on inhibitory control: Given the node set, if nodes with empty retrieval history T_c exist and their count is less than n (the number of nodes exceeding cache capacity), the second term in the importance score calculation of such nodes is substituted with the average value of this term from nodes with non-empty retrieval histories. This mechanism prevents excessive unreasonable suppression of nodes lacking retrieval records caused by a zero-valued second term. If the number of nodes with empty retrieval histories is no less than n, this smoothing operation is skipped, and the original Equation is applied directly for score computation.

C Additional Experiment Results

C.1 Sensitivity Analysis of Number of Retrieved Nodes K_t

K_t denotes the number of nodes sharing the same agent v_i and similar response r_i to be retrieved for each newly created node during the construction of the trajectory layer, and initial node correlations are established based on the retrieved results. Accordingly, v exerts a direct impact on the memory replay and reasoning pipeline of FiniteMem. We analyze the sensitivity of FiniteMem to this hyperparameter by evaluating model performance with $K_t \in \{1, 2, 3, \ldots, 10\}$, and the experimental outcomes are illustrated in Figure6.

As observed, FiniteMem achieves optimal overall performance at $K_t = 3$. When $K_t < 3$, sparse edges are constructed between trajectory nodes, which restricts the excavation of latent correlations among memory nodes. Consequently, insufficient diverse microscopic action references are retrieved in downstream reasoning, resulting in degraded performance. For $3 < K_t < 6$, the number of inter-node connections rises substantially, yet latent correlations have already been fully captured by the established links; further expanding connections yields no performance gains. By contrast, when $7 < K_t \leq 10$, redundant edges introduce substantial noise, which injects misleading microscopic guidance during reasoning and deteriorates model performance. Nevertheless, the diversity of candidate actions remains unaffected by such noise, so the overall performance still outperforms the cases with small K_t.

C.2 Sensitivity Analysis of Retrieval Similarity Thresholds θ_u,θ_p and θ_z

θ_u,θ_p and θ_z are similarity thresholds used to retrieve analogous texts during the construction of the multi-layer memory bank. Specifically, they are adopted to build initial node correlations within the trajectory layer and eliminate redundant similar texts in the perspicacity and knowledge layers. In FiniteMem, we set $\theta_u = \theta_p = \theta_z = 0.8$ To further investigate the sensitivity of these similarity thresholds, we adjust one threshold while keeping the other two fixed for ablation tests, with experimental results presented in Figure7. The results reveal that retrieval similarity thresholds exert a considerable influence on overall performance. For θ_u, increasing the threshold reduces the number

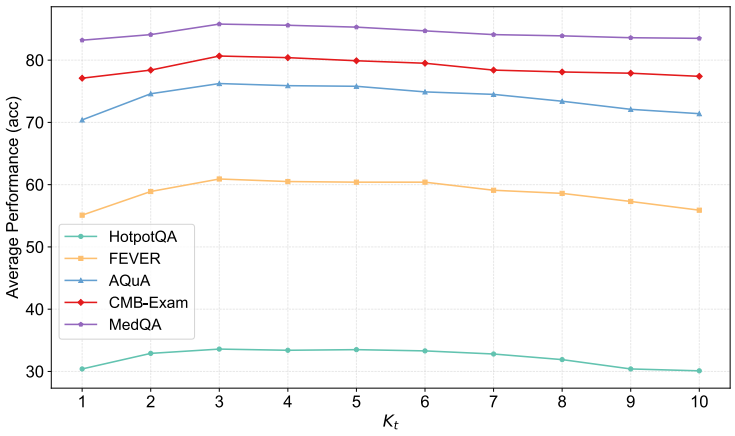

Figure 6: Sensitivity experimental results of hyperparameter K_t. All reported values correspond to the average performance across three types of MAS on all datasets.

of established node correlations, which limits the microscopic guidance available during reasoning and leads to obvious performance degradation. Conversely, lowering θ_u introduces excessive noise into the reasoning process and impairs the performance of MAS. In comparison, performance degradation induced by adjusting θ_p and θ_z is milder. This is because θ_p and θ_z only affect the construction speed and diversity of the memory bank in the early operation stage of MAS. As the agent accumulates more running rounds, the memory bank becomes progressively complete, and model performance gradually recovers without drastic fluctuations.

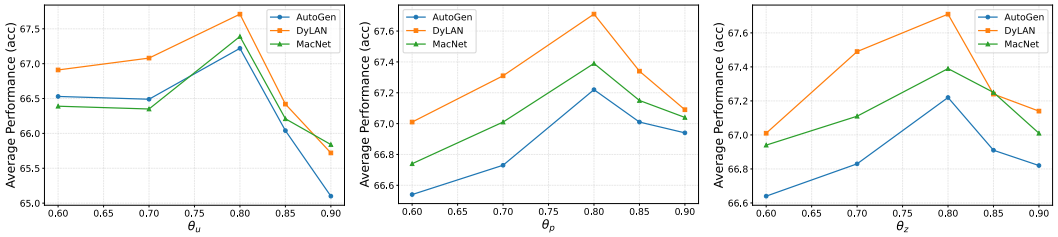

Figure 7: Sensitivity experiment results of threshold, where the data is the average of all datasets on three MAS.

C.3 Sensitivity Analysis of Number of Memory Clusters $K_\mathcal{T}$

When the trajectory layer of FiniteMem executes the replay algorithm based on memory competition, the memory buffer is partitioned into $K_\mathcal{T}$ memory clusters to calculate the intra-cluster diversity, inter-cluster diversity, and importance within each cluster. The number of memory clusters affects the performance of the replay algorithm. Thus, we evaluate FiniteMem under $K_\mathcal{T} = \{2, 4, 5, 8, 10\}$, with experimental results listed in Table4. The overall optimal performance of FiniteMem is achieved at $K_\mathcal{T} = 5$. This is attributed to reduced effectiveness of inter-cluster diversity in the replay algorithm when the number of memory clusters is excessively small. An insufficient number of clusters leads to converged inter-cluster diversity values across all nodes, invalidating inter-cluster diversity as a competition metric and increasing the weight of intra-cluster diversity, which degrades memory buffer quality. Conversely, an overabundance of memory clusters amplifies the impact of inter-cluster importance while suppressing intra-cluster importance, also impairing memory buffer quality. Nevertheless, dataset sizes vary drastically. As demonstrated by results on the CMB-Exam dataset, datasets requiring numerous agent training rounds may benefit from more memory clusters to improve memory buffer quality. This weak trend remains an avenue for future in-depth investigation.

Table 4: Results of the sensitivity analysis for $K_{\mathcal{T}}$. The data shown represent the average results across different datasets for the three MAS.

$K_{\mathcal{T}}$	HotpotQA	FEVER	AQuA	CMB-Exam	MedQA
2	32.94	59.95	75.21	79.38	84.40
4	33.22	60.09	75.37	79.68	84.75
5	**33.59**	**60.91**	**76.24**	80.66	**85.80**
8	32.96	59.91	74.97	**80.91**	84.07
10	32.89	59.64	74.42	80.72	83.94

C.4 Hyperparameter Analysis of a Replay Algorithm Based on Inhibitory Control

We assign weights to distinct metrics in the replay algorithm based on inhibitory control. To further analyze how weights of individual metrics affect the replay algorithm performance, we conduct experiments by adjusting one weight while keeping the other two fixed, and the experimental results are illustrated in Figure8. The weight α of the temporal decay metric imposes negligible influence on model performance. As agent training rounds accumulate, the rounds when perspicacity and knowledge in the memory buffer are generated deviate substantially from the current round, yielding low temporal decay values. Consequently, adjusting the corresponding weight exerts limited effects on the model. Nevertheless, the temporal decay metric cannot be eliminated. At the early training stage where the memory buffer is underdeveloped, the stored memory data remain insufficient; eliminating outdated memories in this stage accelerates the optimization of the memory buffer.

The retrieval enhancement metric and validity metric exert pronounced impacts on model performance. The retrieval enhancement metric primarily captures memories that are easily retrieved under the current task environment, which typically represent implicit shared features within the task. Hence, appropriately prioritizing the retrieval enhancement metric facilitates performance improvement. However, overemphasizing this metric suppresses the validity metric and retains numerous low-efficiency memories. By contrast, excessive emphasis on the validity metric hinders the retention of newly generated memories and reduces memory diversity. Therefore, balancing the retrieval enhancement metric and validity metric with a moderate bias toward the retrieval enhancement metric yields optimal performance.

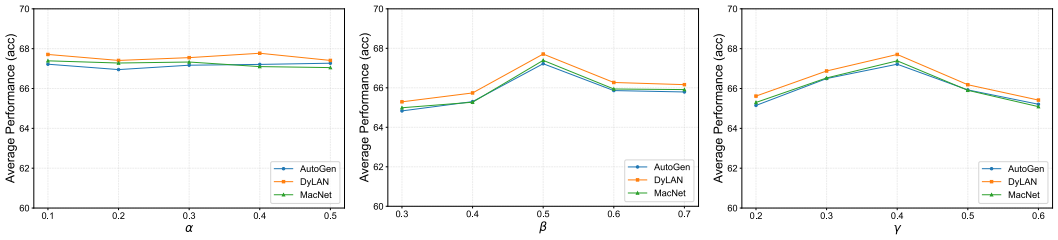

Figure 8: Results of the hyperparameter experiment for the replay algorithm based on inhibition control.

C.5 Sensitivity Analysis of Hyperparameter Selection

The memory retrieval weight α_r primarily balances the proportions of the new query q_{new} and memory information r_i during retrieval. Accordingly, the sensitivity analysis of α_r reveals how retrieving similar queries and similar trajectories impacts model performance, with experimental results summarized in Table 5. The performance degradation induced by reducing α_r is substantially more severe than the performance loss caused by increasing α_r. A larger α_r facilitates the retrieval of similar queries and their corresponding memories; similar queries generally correspond to analogous solutions and reasoning paradigms, thereby boosting model performance. Nevertheless, overprioritizing similar queries diminishes the diversity of retrieved results and impairs model performance. In contrast, excessive emphasis on similar trajectories introduces noise into retrieval

Table 5: Results of the sensitivity analysis for α_r. The data shown represent the average results across different datasets for the three MAS.

α_r	HotpotQA	FEVER	AQuA	CMB-Exam	MedQA
0.6	32.80	59.82	74.61	79.31	84.27
0.7	33.02	59.85	75.07	79.23	84.36
0.8	**33.59**	**60.91**	**76.24**	**80.66**	**85.80**
0.9	33.31	60.59	75.81	80.16	85.41
1.0	33.01	60.37	75.76	80.24	85.40

Table 6: Sensitivity experiment results for $K_r^{\mathcal{T}}$, $K_r^{\mathcal{P}}$ and $K_r^{\mathcal{Z}}$. The data presented here represent the average results across different datasets for the three MAS.

Parameter	Num	HotpotQA	FEVER	AQuA	CMB	MedQA
$K_r^{\mathcal{P}}$	1	**33.59**	**60.91**	**76.24**	**80.66**	**85.80**
	2	33.12	60.03	75.23	79.84	84.83
	3	33.11	59.89	74.98	79.39	84.33
	4	32.91	59.68	74.69	78.78	84.04
$K_r^{\mathcal{T}}$	1	32.83	59.74	74.66	79.21	84.26
	2	32.95	59.91	75.08	79.38	84.47
	5	**33.59**	**60.91**	**76.24**	**80.66**	**85.80**
	7	33.17	60.19	75.63	79.84	85.09
	10	33.08	59.97	75.25	79.75	84.60
$K_r^{\mathcal{Z}}$	1	33.51	60.64	75.36	79.19	84.16
	2	33.44	60.76	75.78	79.88	84.72
	5	**33.59**	**60.91**	76.24	80.66	**85.80**
	7	33.10	60.44	**76.43**	**80.89**	85.76
	10	33.06	59.89	75.90	80.57	85.59

outputs by incorporating irrelevant operations and reasoning logics, which degrades model performance.

Furthermore, the retrieval count within multi-level memory imposes effects on model performance. We conduct a sensitivity analysis on retrieval counts of memories at distinct layers, with experimental results presented in Table 6. It is observed that a larger retrieval count K_r^p for the perspicacity layer does not consistently yield superior performance. Data in the perspicacity layer macroscopically guide the reasoning and execution logic of agents. Retrieving excessive perspicacity memories introduces discrepancies among multiple perspicacity entries, disrupting regular agent reasoning and triggering erroneous inference chains.

For trajectory-layer memories that microscopically steer concrete agent operations, setting the retrieval count to $K_r^t = 5$ constitutes a rational configuration. An overly small K_r^t restricts the volume of reference memories, compelling agents to rigidly mimic stored memory content and generate erroneous execution steps. Although Equation 14 guarantees the relevance of retrieved trajectory memories, the agent ultimately receives all successor nodes of the K_r^t retrieved trajectory nodes as memory input. An excessively large K_r^t inevitably incorporates substantial noise and deteriorates agent performance. Regarding the knowledge-layer retrieval count K_r^z, moderately increasing this parameter improves the retrieval of relevant knowledge in medical and mathematical domains, where partial test samples rely on overlapping knowledge. In contrast, commonsense reasoning exhibits weak generalization of associated knowledge, rendering models insensitive to variations in K_r^z. Nevertheless, continuous growth of K_r^z still introduces extraneous noise and degrades overall model performance.

C.6 Case study

Figure 9 illustrates the specific memory cues supplied by FiniteMem across diverse tasks. For instance, under the MedQA + AutoGen configuration, given the task query "What is the predominant

mechanism underlying type II respiratory failure in patients with bronchial asthma? A. Decreased alveolar ventilation. ... E. None of the above", FiniteMem first retrieves the analogous perspicacity memory "Clinical reasoning follows the logic of definition anchoring, pathological localization and direct causal linkage" to clarify the overall reasoning framework for the query. It then extracts the relevant knowledge memory "Type II respiratory failure is essentially characterized by hypoxemia combined with hypercapnia", which delineates the subsequent analytical orientation. Finally, the model retrieves the associated trajectory memory "Reflection: The mechanism of bronchial asthma should be analyzed with emphasis on its definition", offering actionable guidance for the ongoing task. Supported by these multi-type memories, AutoGen generates the reasoning step: "Reflection: The pathophysiological mechanism of type II respiratory failure in bronchial asthma patients shall be analyzed, focusing on the definition of type II respiratory failure and the specific manifestations of hypoxemia-hypercapnia in asthma". Further analysis attributes hypoxemia and hypercapnia to insufficient alveolar ventilation (VA), yielding the correct answer "A. Decreased alveolar ventilation". This case demonstrates the effective support of FiniteMem's hierarchical memory system for intelligent agents.

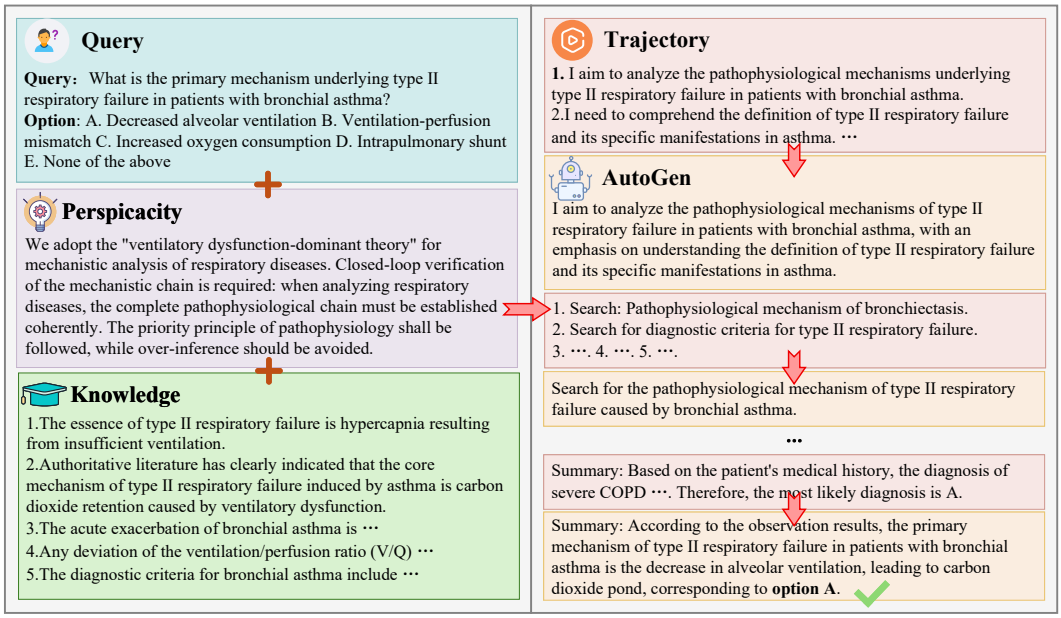

Figure 9: Case study of FiniteMem.

C.7 Performance of fewer parameter models

To further validate the performance of FiniteMem on small-scale large language models, the open-source large language models Qwen3-14B and Qwen3-7B Yang et al. (2025) are adopted as base models for evaluation, with experimental results summarized in Tables 7 and 8. Significant performance degradation is observed as model parameter size decreases, and the performance margin of FiniteMem over the baseline gradually narrows. This phenomenon arises from the weakened instruction-following capability of small-scale large language models: smaller models struggle to comprehend the function of instructions, which hinders effective memory utilization. Nevertheless, the trajectory layer of FiniteMem provides fine-grained operational guidance, enabling small-scale large language models to directly imitate trajectory memories; consequently, FiniteMem still outperforms the baseline. This experiment also indicates that boosting the efficacy of memory modules for small-parameter LLMs remains a critical research direction for future work.

Table 7: Performance comparison with single- and multi-agent memory architectures across five benchmarks. The underlying LLM backbone is Qwen3-14b.

MAS	Memory	HotpotQA	FEVER	AQuA	CMB-Exam	MedQA	Avg.
AutoGen	non-memory	25.97	51.32	62.94	72.91	80.01	58.63
	Voyager	28.66	56.84	69.09	73.63	80.91	61.83
	MemoryBank	29.94	55.21	65.18	72.12	79.71	60.44
	Generative	28.53	53.64	68.88	73.81	80.65	61.10
	ChatDev	21.73	54.61	60.09	73.10	80.58	58.02
	MAD-M^2	31.99	**59.37**	73.01	74.14	81.24	63.95
	G-Memory	32.81	59.02	71.84	74.71	81.37	63.95
	FiniteMem	**32.95**	58.91	**73.43**	**76.37**	**82.40**	**64.81**
DyLAN	non-memory	28.14	53.96	63.57	73.54	79.43	59.73
	Voyager	29.10	55.27	70.24	75.19	81.54	62.27
	MemoryBank	26.73	52.50	58.43	69.31	77.29	56.85
	Generative	26.36	55.28	70.17	75.69	79.78	61.46
	ChatDev	19.88	53.30	56.17	68.34	78.23	55.18
	MAD-M^2	30.23	57.48	71.05	76.85	81.46	63.41
	G-Memory	30.84	58.12	71.93	76.42	81.67	63.80
	FiniteMem	**32.04**	**58.81**	**73.83**	**78.72**	**82.56**	**65.19**
MacNet	non-memory	26.05	54.36	65.12	73.87	79.89	59.86
	Voyager	29.77	56.53	70.61	75.94	81.76	62.92
	MemoryBank	30.43	54.93	64.06	72.48	79.97	60.37
	Generative	28.13	56.51	66.35	74.83	80.94	61.35
	ChatDev	26.64	54.18	65.89	75.27	78.22	60.04
	MAD-M^2	31.14	58.34	71.80	76.76	82.24	64.06
	G-Memory	31.69	58.03	71.28	77.32	81.37	63.94
	FiniteMem	**32.19**	**58.48**	**73.10**	**78.05**	**82.72**	**64.91**

Table 8: Performance comparison with single- and multi-agent memory architectures across five benchmarks. The underlying LLM backbone is Qwen3-7b.

MAS	Memory	HotpotQA	FEVER	AQuA	CMB-Exam	MedQA	Avg.
AutoGen	non-memory	22.10	43.38	53.22	61.66	67.73	49.62
	Voyager	24.25	48.11	58.42	62.28	68.42	52.30
	MemoryBank	25.35	46.71	55.11	60.99	67.44	51.12
	Generative	24.13	45.36	58.27	62.46	68.20	51.68
	ChatDev	18.42	46.21	50.82	61.81	68.19	49.09
	MAD-M^2	27.05	50.19	61.73	62.74	68.73	54.09
	G-Memory	27.77	49.94	60.77	63.23	68.83	54.11
	FiniteMem	**27.88**	**50.85**	**62.12**	**64.63**	**69.71**	**55.04**
DyLAN	non-memory	23.81	45.62	53.76	62.20	67.17	50.51
	Voyager	24.60	46.73	59.42	63.61	68.97	52.67
	MemoryBank	22.67	44.39	49.42	58.65	65.39	48.10
	Generative	22.30	46.74	59.37	64.01	67.48	51.98
	ChatDev	16.85	45.09	47.52	57.82	66.26	46.71
	MAD-M^2	25.59	48.61	60.08	65.32	68.91	53.70
	G-Memory	26.07	49.19	60.87	64.64	69.09	53.97
	FiniteMem	**27.10**	**49.77**	**62.45**	**66.57**	**69.82**	**55.14**
MacNet	non-memory	22.04	45.97	55.09	62.52	67.57	50.64
	Voyager	25.17	47.85	59.73	64.21	69.13	53.22
	MemoryBank	25.72	46.47	54.22	61.31	67.62	51.07
	Generative	23.79	47.83	56.13	63.33	68.49	51.91
	ChatDev	22.50	45.84	55.75	63.68	66.15	50.78
	MAD-M^2	26.35	49.33	60.73	64.93	69.59	54.19
	G-Memory	26.83	49.07	60.39	65.39	68.86	54.11
	FiniteMem	**27.25**	**49.45**	**61.84**	**66.03**	**69.95**	**54.90**

D PROMPT SET

In this section, we will provide a detailed introduction to the prompts used in FiniteMem. The prompt for the perspicacity agent v_p, which extracts perspicacity, is shown in Figure 10; the prompt for the knowledge extraction agent v_z, which extracts useful knowledge, is shown in Figure 11; and the prompt ultimately used for reasoning is shown in Figure 12.

Perspicacious Agent

Prompt:
You are a proficient summarizing agent. Your task is to generate insightful experience summaries based on the given execution records of agent tasks. These summaries help the agent effectively utilize existing information, avoid redundant reasoning, and ensure task continuity.
Summary Requirements:
1. Reasoning Logic Summary: Summarize the overall reasoning logic of the execution records, which shall be instructive and generalizable.
2. Experience Summary: Conclude the lessons learned throughout the reasoning process, clarifying the key reasoning steps to emphasize and the errors to avoid in subsequent reasoning.
3. Clarity and Conciseness: Compose experience summaries with clear and accurate wording, free of unnecessary details and complex formatting.
4. Information Correction and Filtering: Filter or correct incorrect intermediate states and irrelevant information to improve the accuracy of summaries.
5. Format Reference: Generate experience summaries by following the given examples, with no limit on the number of summaries.

Figure 10: Prompt for the Perspicacity Agent v_p.

Knowledge Extraction Agent

Prompt:
You are an expert tasked with extracting all professional knowledge contained in the given execution records. The extraction requirements are as follows:
Evidence-based professional knowledge identification: The extracted knowledge shall be core and sophisticated knowledge from the execution records.
1. Task relevance: Summarize only content related to the provided execution records without expansion or extrapolation.
2. Clarity and conciseness: Summarize information with clear and accurate language, and exclude unnecessary details and irrelevant content.
3. Finally output the knowledge in list format. There is no lower limit on the number of entries, while the total number shall not exceed five.

Figure 11: Prompt for the Knowledge Extraction Agent v_z.

Inference

Prompt:

You are a problem-solving intelligent agent.

Experience Summary

Below are key problem-solving points accumulated from similar past tasks, which can help improve the accuracy of your current response:

{insight_text}

Reference Knowledge

Below is reference knowledge accumulated from similar past tasks, which can help improve the accuracy of your current response:

{knowledge_text}

Possible Next Operations

Below are potential operations summarized from similar past tasks. You may refer to them to generate reasoning steps for the next move and boost the accuracy of your current response:

{trajectory_text}

Figure 12: Prompts for reasoning.