LEARNING WHO KNOWS WHAT: LEARNED ROLE-AWARE AGENTIC MEMORY IN LLM-BASED MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Persistent memory helps LLM agents reuse experience across tasks, but experience useful to one role may be unsuitable for another. We study how agents should decide whether to use memory and whose experience to consult for the current task. We propose **LoRAM**, a role-aware memory framework that separates source eligibility from online source selection. A role compatibility gate identifies eligible peers, while an online contextual policy selects among bypassing memory, using local experience, consulting a single peer, or combining local and peer memories under a common token budget. Memories remain in stores owned by individual agents, and only consolidated procedures are eligible for transfer between agents. A transactive directory tracks source expertise and directed transfer outcomes to inform routing decisions. We evaluate LoRAM on sequential Research, Database, and Coding tasks from MultiAgentBench. LoRAM achieves the highest observed mean scores among the evaluated configurations across all three environments. On Research with GPT-4o-mini, it also uses 15.7% fewer tokens per task than the strongest baseline by mean score. Research ablations compare the full method with local memory, shared access, static role routing, and provenance-aware item routing. These results suggest that learning whose experience to reuse is a promising approach to memory management in teams of specialized agents.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems are now used for software engineering, scientific research assistance, database diagnosis, and long-horizon planning. Instead of solving a task with a single model call, these systems decompose work across agents with different roles, allow agents to exchange intermediate results, and use communication or tool interaction to refine the final answer. As such systems are deployed over long task sequences, *memory* becomes a key mechanism for lifelong improvement: by accumulating episodes and procedures, agents can avoid past mistakes and reuse successful strategies (Wu et al., 2026; Xu et al., 2025; Zhu et al., 2025).

Most multi-agent memory systems, however, determine the reach of persistent experience primarily through a predefined storage or organizational structure. In a local-memory design, each agent keeps its experience to itself. In a shared-memory design, all agents write to and read from a global store. In a hybrid design, information is partitioned into private and shared tiers (Wu et al., 2026). It helps to separate four decisions that these designs tend to bundle: *storage ownership* (who holds a record), *availability scope* (who is permitted to see it), *item retrieval* (which records are selected for a prompt), and *source selection* (whose experience supplies the final memory context). Recent systems have made item retrieval dynamic and even role-conditioned (Liu et al., 2025). Storage organization defines the candidate memory scope, but does not by itself specify an outcome-adaptive policy over access modes and sources. Item retrieval can use provenance to select among sources implicitly; we study making this source choice an explicit online decision.

Role specialization makes this choice consequential: a strategy useful to a Creator may be unsuitable for an Optimizer. Shared access permits useful transfer but can expose incompatible advice, whereas strictly local access excludes transferable peer experience. This motivates *task-conditioned, role-*

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

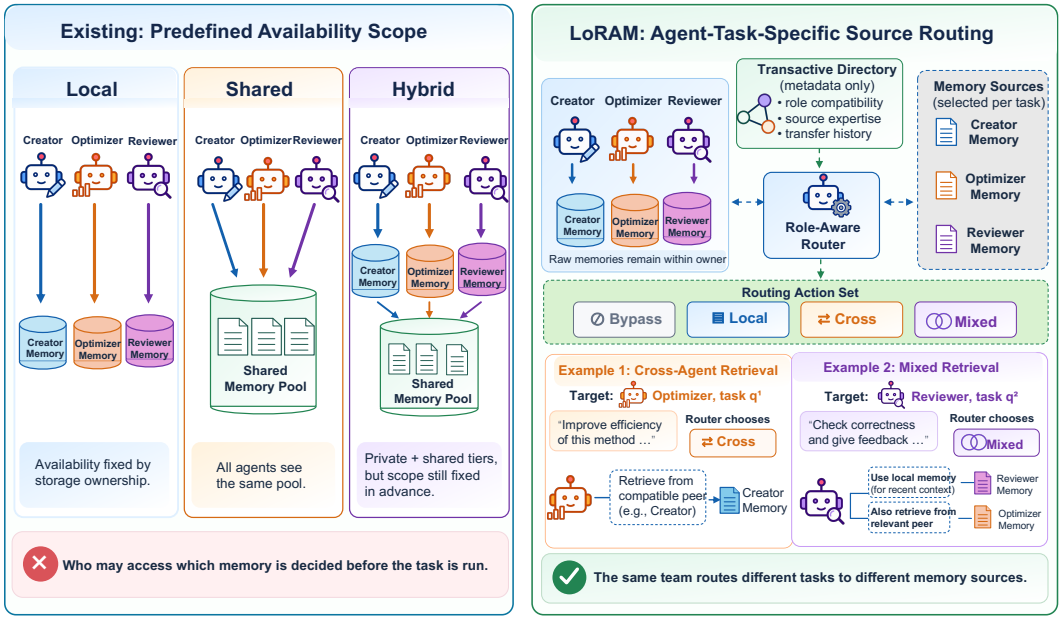

Figure 1: Memory availability and task-conditioned source selection. Local, shared, and hybrid organizations define candidate memory scopes. LoRAM explicitly selects an access mode and source for each target agent and task. Cross-agent and mixed access illustrate alternative decisions for the same team.

aware source selection: deciding whether to use memory and whose experience to consult, based on the task, source and target roles, and observed outcomes. Item retrieval and source selection are complementary: the former ranks records, while the latter chooses among bypass, local, peer, and mixed access. Figure 1 contrasts predefined memory scopes with these online choices over persistent agent-owned stores.

We instantiate this view in **LoRAM** (Learned Role-Aware Agentic Memory). Each agent owns a role specialized memory of episodes and consolidated procedures, while a team level transactive directory records ownership, expertise, role compatibility, and directed transfer history. For each agent task pair, LoRAM first restricts retrieval to role compatible sources and then learns online whether to use no memory, local experience, one compatible peer, or a budgeted local plus peer combination. The selected memories share one context budget, and the policy is updated from the observed outcome of the selected action.

Contributions. This paper makes three contributions:

- We formulate memory reuse in role-structured teams as an explicit online choice of access mode and source, separating candidate availability, source selection, and item retrieval over persistent agent-owned stores.
- We introduce LoRAM, which combines a role-compatibility gate, source-preserving candidates, and an outcome-adaptive policy over bypass/local/cross/hybrid actions under one memory-token budget. A transactive directory tracks source expertise and directed outcome history.
- We evaluate mean and anytime performance, cumulative advantage, and token use on MultiAgentBench. We report the highest observed mean scores among the evaluated configurations across three environments and compare local-only, shared, and static-role alternatives on Research.

2 Related Work

Agentic and multi-agent memory. LLM-agent memory has evolved from reflective and experiential memory (Shinn et al., 2023; Wang et al., 2024) to structured, procedural, and learned memory management (Xu et al., 2025; Ma et al., 2026; Jiang et al., 2026a;b). These methods increasingly learn what experience to retain, how to organize it, and which memories to retrieve, but primarily operate over memories within a predefined ownership or access scope. Multi-agent memory introduces an additional design choice: how persistent experience should be exposed across agents. LLMA-Mem (Wu et al., 2026) combines episodic, procedural, and transactive memory and studies local, shared, and hybrid configurations. Its hybrid design keeps episodes local while sharing procedures. LoRAM builds on this architectural setting by explicitly learning access-mode and source choices from task outcomes. Other multi-agent systems organize experience through shared, structured, or distributed stores; we review these approaches in Appendix D.

From item routing to source routing. Role-aware context routing provides the closest precedent to source-level memory routing. RCR-Router (Liu et al., 2025) dynamically selects role and task stage relevant items from a centralized shared memory under a token budget. Its routing unit is therefore the *memory item*: which entries from a common store should a target role receive? A complementary decision arises when persistent experience remains agent owned: whether long-term memory should be used at all and, if so, which memory source should be consulted. LoRAM addresses this *source-level* decision after constructing separate candidate sets from local and eligible peer stores: the policy selects which candidates supply the final memory context. This perspective is inspired by transactive memory—the team's meta knowledge of "who knows what" (Wegner, 1987) and casts source selection under partial feedback as an online contextual decision (Li et al., 2010; Auer et al., 2002). A broader comparison with experiential memory, shared and distributed memory, routing, communication, and memory evaluation appears in Appendix D.

3 Problem Formulation

We consider a lifelong multi-agent collaboration setting. A team of agents $A = \{a_1, \ldots, a_n\}$ processes a task sequence $Q = \{q_1, \ldots, q_T\}$. Each agent a_i has a role r_i and a profile encoding its specialization. For each task q_t, the system may retrieve memory and insert it into each agent's prompt before communication, tool use, or environment interaction begins. After execution, the system observes a task score TS_t, a coordination score CS_t, and token cost.

The goal is to maximize long-horizon performance under memory and context budget constraints. Following prior lifelong multi-agent evaluation, we define a unified score

$$\mathrm{J}_t = \frac{\mathrm{TS}_t + \mathrm{CS}_t}{2}. \tag{1}$$

Memory as a routing action. Each agent a_i owns a persistent local memory M_i; peer memories $\{M_j\}_{j \neq i}$ are owned by other agents. Rather than fixing availability scope in advance, for each agent task pair (a_i, q_t) LoRAM selects one *memory-access action* from

$$\mathcal{A}_{i,t} = \{\mathrm{bypass},\ \mathrm{local},\ \mathrm{cross}(j),\ \mathrm{hybrid}(i, j)\}, \tag{2}$$

where bypass injects no long-term memory, local retrieves from M_i only, $\mathrm{cross}(j)$ retrieves from a single compatible peer M_j, and $\mathrm{hybrid}(i, j)$ combines local memory with exactly one compatible peer. An action thus jointly specifies (a) whether long-term memory is used at all and (b) which ownership scope supplies it; it does not rewire the team's communication graph, which is held fixed throughout. All actions render their selected memories into compact cards under one memory-token budget B. Fixed local and single-peer hybrid policies are static special cases; a globally shared pool is evaluated separately as LoRAM-Shared. The chosen action yields normalized reward r_t derived from J_t.

This formulation differs from standard retrieval augmented generation in two ways. First, the decision is agent and role specific: a peer memory useful for one target agent can be harmful for another, so *negative transfer*—a peer memory that lowers J_t relative to the local or bypass counterfactual—is a first-class failure to avoid. Second, the decision is sequential: an action affects future estimates

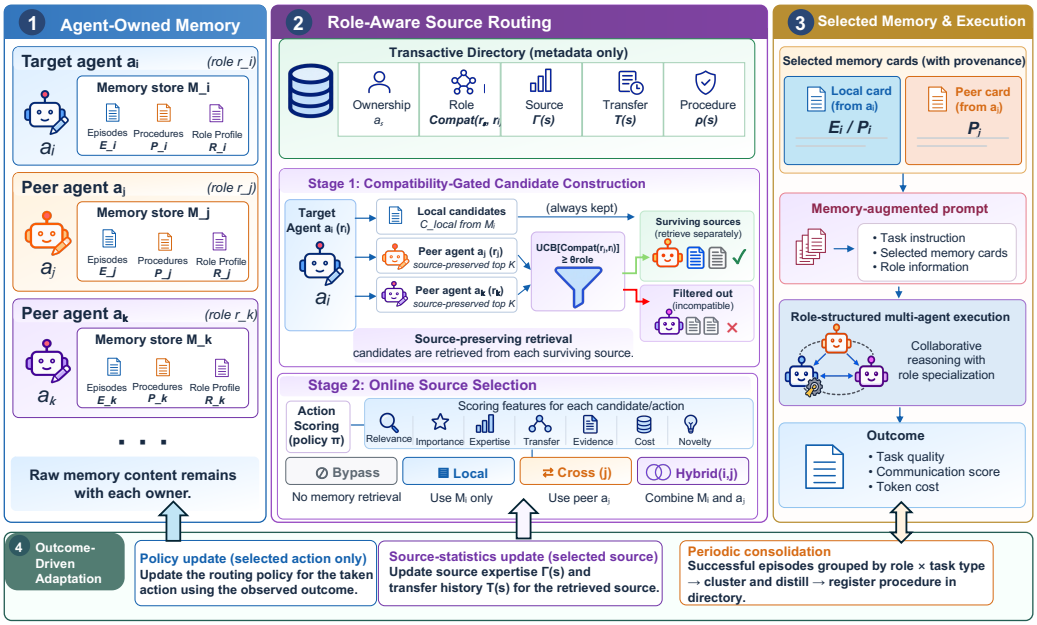

Figure 2: The LoRAM framework. Agent-owned stores retain episodes and procedures, while a transactive directory stores role profiles, routing metadata, and pointers. For each target agent and task, role compatibility gates peer sources, and candidates are retrieved separately from local and eligible peer stores. An online policy selects bypass, local, cross-agent, or hybrid access under one token budget. Outcomes update the policy and selected-source statistics, while periodic consolidation creates reusable procedures. The execution panel illustrates a hybrid selection.

of expertise and transfer utility. We therefore seek to improve performance over the task sequence, while recognizing that the observed team-level reward provides only approximate credit for each agent's memory decision.

4 The LoRAM Framework

LoRAM replaces a team wide, predefined availability scope with a memory source routing decision made for each agent and task. It combines native per agent memories (§4.1), a transactive directory, compatibility gated candidate construction (§4.3), and an online policy over the actions in Eq. 2 (§4.3). Memory content remains in its owner's store and is exposed to peers only through routed retrieval. Figure 2 summarizes the architecture and its outcome-driven updates.

4.1 Role-Specialized Memory and Transactive Directory

Each agent a_i with role r_i maintains a native store $M_i = \langle E_i, P_i \rangle$, while role profile proficiencies R_i reside at the team-level transactive directory. Local and peer memories share the same extraction and consolidation pipeline but differ in transfer eligibility: *episodic experiences E_i are strictly local-only*, whereas *only consolidated procedural strategies P_i are eligible for cross-agent transfer*.

Episodic memory E_i retains task-level experiences and extracted lessons; procedural memory P_i retains consolidated strategies with success/failure evidence. Consolidation (§4.4) grounds transferable procedures in repeated success. Role profiles track per-type proficiency $\Gamma_i(\tau)$ as a Beta posterior in the directory. Record definitions and reliability features appear in Appendices B.2 and B.4.

Task typing. Task type τ_t is assigned by deterministic embedding lookup without extra LLM calls or outcome labels. Research uses semantic anchors discovered from agent profiles; Database and Coding use active-team profiles (Appendix B.3).

Transactive directory. A team-level transactive directory G_T records the meta knowledge used by the router: memory ownership, task specific source expertise, procedure–task support, role compatibility, and directed transfer history. It stores pointers and sufficient statistics rather than memory content. *Role compatibility* is a structural eligibility estimate: with profile-embedding similarity $\mu_0 = \mathrm{sim}(d_s, d_t)$ rescaled to $[0, 1]$ and prior strength c, it is initialized as $\mathrm{Compat}(r_s, r_t) \sim \mathrm{Beta}(\alpha_0, \beta_0)$ with $\alpha_0 = c\mu_0$ and $\beta_0 = c(1 - \mu_0)$. To keep compatibility distinct from directed transfer utility, it is updated only when an explicit role-pair transfer label is available, not from every team reward. Compatibility determines which role pairs are eligible; the router estimates the task-conditioned utility of an eligible action.

4.2 Four Distinct Signals

LoRAM's routing rests on four signals that are easy to conflate but answer different questions, and it keeps them separate:

- *Semantic relevance*: does a memory match the current task? (embedding similarity between the memory and q_t.)
- *Source expertise*: has the owning agent demonstrated competence on this task type? (its proficiency posterior $\Gamma_o(\tau_t)$.)
- *Directed transfer history* ("trust"): what outcomes followed access from source a_j to receiver a_i? (a directed $\mathrm{Beta}(1, 1)$ posterior over observed transfer outcomes, used as a proxy rather than a causal estimate of benefit.)
- *Role compatibility*: is knowledge transfer between the two functional roles structurally permitted at all? (defined in the transactive directory above.)

Relevance, expertise, and transfer history enter the online router as normalized *features*; role compatibility acts as a hard *gate* on eligibility. Directed transfer history and proficiency are directed and per-pair/per-type, so they aggregate evidence across tasks even when any single task's credit signal is noisy.

4.3 Role-Aware Memory Source Routing

Source-preserving candidate construction. For target agent a_i and task q_t, LoRAM first retrieves the top-K local candidates from its full native store $M_i = \langle E_i, P_i \rangle$ by semantic relevance. It then determines the eligible peer sources through the hard role-compatibility gate: a peer a_j ($j \neq i$) is eligible iff its optimistic compatibility clears a threshold,

$$\mathrm{UCB}\big[\mathrm{Compat}(r_j, r_i)\big] \;=\; \hat{\mu} + \lambda\sqrt{\tfrac{\hat{\mu}(1-\hat{\mu})}{n+1}} \;\geq\; \theta_{\mathrm{role}}, \tag{3}$$

where $\hat{\mu}$ is the compatibility posterior mean and n its observation count; the optimism term allows uncertain role pairs to remain eligible. Its uncertainty decreases when additional role-pair labels are observed.

To limit peer exposure to unconsolidated execution traces, cross-agent transfer is restricted to consolidated strategies: from *each* eligible peer $a_j \in \mathcal{E}$ separately, LoRAM retrieves top-K candidates exclusively from that peer's procedural memory P_j, leaving raw episodic traces E_j local-only. Eligible peer candidates are retrieved using the same query embedding and depth K without pre-merging peers into a centralized pool. Ownership and provenance are strictly preserved throughout, ensuring every candidate carries its source role, Laplace-smoothed reliability $\hat{\rho}$, and evidence support $s + f$; empty candidate sets naturally fall back to the actions that remain feasible.

Online source selection. Given local and per peer candidates, LoRAM forms the action set $\mathcal{A}_{i,t}$ of Eq. 2 and selects one action with an online contextual policy. Each action is represented by a normalized vector $x_{i,t,a}$ containing relevance, memory importance, source expertise, directed transfer history, evidence support, negative retrieval cost, peer novelty relative to local memory, and action type. Agent a_i scores each action with a contextual upper-confidence bound,

$$\begin{aligned} U_{i,t}(a) &= \hat{\theta}_{i,t}^{\top} x_{i,t,a} + \lambda_{\mathrm{UCB}}\sqrt{x_{i,t,a}^{\top} A_{i,t}^{-1} x_{i,t,a}}, \\ \hat{\theta}_{i,t} &= A_{i,t}^{-1} b_{i,t}, \end{aligned} \tag{4}$$

and selects $a_i^\star = \arg\max_{a \in \mathcal{A}_{i,t}} U_{i,t}(a)$. The router may inject no memory or prefer a peer over local memory. Candidate retrieval precedes this choice, so bypass does not eliminate candidate-construction cost. Every mode is subject to the same memory-token cap B; a hybrid action splits that budget between local and peer cards. Selected cards label their owner and role.

After the task, a normalized reward $r_t \in [0, 1]$ derived from J_t updates the agent's shared policy parameters using only the selected action's observed feature–reward pair,

$$A_{i,t+1} = A_{i,t} + x_{i,t,a_i^\star} x_{i,t,a_i^\star}^\top, \qquad b_{i,t+1} = b_{i,t} + r_t \, x_{i,t,a_i^\star}, \tag{5}$$

so unobserved alternatives are not assigned counterfactual rewards, although their predictions can change through the shared parameters. The same team outcome updates selected-source proficiency and directed transfer statistics; these are outcome associations, not isolated causal effects of memory transfer.

4.4 Consolidation and Bounded Maintenance

Every N tasks, successful episodes are grouped by task type and role, clustered, and distilled into procedures supported by repeated success. The same pipeline runs for every agent. Confidence-based pruning, duplicate merging, and per-key capacity bounds maintain bounded stores; these do not constitute a learned retention policy. Full settings appear in Appendix B.5.

The complete end-to-end execution loop of LoRAM, from task typing and role-aware routing to posterior updates and consolidation, is summarized in Algorithm 1 in Appendix B.9.

5 Experiments

Our evaluation is organized around three research questions. **RQ1 (overall performance):** does LoRAM improve mean and anytime team performance over no-memory and existing memory methods? **RQ2 (cross-agent access):** how does the full method compare with its local-only variant (LoRAM-Local)? **RQ3 (routing design):** how does the full method compare with static role routing, unstructured shared access, and provenance-aware item routing? RQ1 is answered by the main performance results; RQ2–RQ3 are addressed by the Research ablations (§5.3).

5.1 Experimental Setup

Benchmark and tasks. We evaluate LoRAM on three task-oriented environments from MultiAgentBench (Zhu et al., 2025): *Research*, *Database*, and *Coding*, using 100 sequential tasks from each environment (300 tasks in total). In the *Research* environment, agents with complementary researcher profiles collaborate to produce a new research proposal conditioned on a provided paper introduction. In the *Database* environment, five agents collaboratively diagnose the root cause of performance anomalies in a simulated PostgreSQL database through tool use and communication. In the *Coding* environment, agents with complementary software-development roles collaborate on implementation, execution, debugging, testing, revision, and code review. Together, the three environments span open-ended generation, interactive diagnosis, and executable problem solving. We preserve the same task order within each environment across memory configurations to support paired comparison and lifelong analysis.

Baselines. We compare against three existing memory configurations and our proposed method. All four share the same underlying agent graph, prompts, and tools, differing only in how cross-task memory is maintained:

- **No Memory** (baseline). Per-task shared scratchpad only; memory is wiped between tasks. This isolates the contribution of cross-task memory from the underlying multi-agent collaboration loop.
- **A-MEM** (Xu et al., 2025) (baseline). Per-agent ChromaDB vector store with LLM-driven memory evolution; retrieves top-k past memories and injects them into the agent prompt.
- **LLMA-Mem** (Wu et al., 2026) (baseline). Episodic-consolidation memory: each agent's per-task episodes are summarised at consolidation interval N and made available to subsequent tasks.

- LoRAM(ours). Native per-agent memory with role-aware online routing over distributed stores (§4).

We additionally compare local-only, shared-access, and static-role variants of LoRAM (§5.3). These comparisons evaluate alternative access and routing designs under a common memory pipeline.

Backbones. We report Research results with two backbones and cross-environment results with GPT-4o-mini:

- **Qwen2.5-7B-Instruct**, served locally with vLLM (Kwon et al., 2023); serving configuration in Appendix B.
- **GPT-4o-mini** (OpenAI et al., 2024) accessed through the Azure API.

Both backbones receive identical prompts and identical k, consolidation interval, and evolution settings.

Evaluation metrics. For Research we use the benchmark joint score

$$\begin{aligned} \mathrm{TS}_t &= 20\,\mathrm{mean}(\mathrm{innovation}_t, \mathrm{safety}_t, \mathrm{feasibility}_t), \\ \mathrm{CS}_t &= 20\,\frac{\overline{\mathrm{plan}}_t + \overline{\mathrm{comm}}_t}{2}, \\ J_t &= (\mathrm{TS}_t + \mathrm{CS}_t)/2. \end{aligned}$$

Rubric dimensions use a 1–5 scale; $20x$ maps a valid mean x to $[20, 100]$. Coding uses the mean of four rubric dimensions; Database uses diagnosis accuracy. Judge settings, rubric definitions, and the alternative map $25(x-1)$ appear in Appendix E. Iteration-level -1 signals are excluded from rubric means; failed tasks receive zero (Appendix B.7). These $S_{e,t}$ are not pooled across environments. With the same failure rule,

$$\mathrm{AP}_{e,t} = \frac{1}{t}\sum_{s=1}^{t} \widetilde{S}_{e,s}, \quad \mathrm{AIP}_e = \frac{1}{T}\sum_{t=1}^{T} \mathrm{AP}_{e,t}, \quad \mathrm{CMA}_{e,t}^{(m)} = \sum_{s=1}^{t} (\widetilde{S}_{e,s}^{(m)} - \widetilde{S}_{e,s}^{(\mathrm{NoMem})}).$$

Implementation. All experiments use the MARBLE framework. Unless otherwise stated, LoRAM uses the same retrieval depth, memory budget, and consolidation schedule across environments and backbones, and every memory-access mode is rendered under one shared memory-token budget so that modes differ in *source*, not in context allowance. Full implementation details, hyperparameters, memory schemas, routing features, prompts, and failure-handling rules are provided in Appendix B.

5.2 Main Results

Lifelong performance on Research. Table 1 reports average task performance, anytime performance, and token cost on the Research environment. LoRAM achieves the highest observed mean J under both backbones. With GPT-4o-mini, LoRAM reaches 87.62, compared with 87.38 for LLMA-Mem, 86.38 for A-MEM, and 86.01 for No Memory. With Qwen2.5-7B, the absolute differences are smaller: LoRAM reaches 82.90, compared with 82.70 for LLMA-Mem, 82.43 for A-MEM, and 82.47 for No Memory. On GPT-4o-mini, LoRAM uses 15.7% fewer tokens per task than LLMA-Mem, while No Memory retains the lowest token count. The mean-score margins over LLMA-Mem are small; these results describe single task streams rather than variation across independent runs.

Performance across environments. Table 2 compares all methods across the three MultiAgentBench environments using GPT-4o-mini. LoRAM achieves the highest observed mean in all three settings. The improvement is modest on Research (87.62 vs. 87.38 for LLMA-Mem), where baseline performance is already high, but substantially larger on Database (40.0 vs. 35.0). On Coding, LoRAM reaches 71.51, compared with 70.28 for No Memory, 70.14 for A-MEM, and 69.01 for LLMA-Mem.

Backbone sensitivity. Memory gains are smaller with Qwen2.5-7B than with GPT-4o-mini; with only two backbones, this is a robustness observation (Appendix C).

Table 1: Main results on 100 sequential Research tasks. J is the joint score $(\mathrm{TS} + \mathrm{CS})/2$; **Bold** marks the best result per column within each backbone and underline the runner-up. Lower token count indicates higher efficiency.

Backbone	Method	J (mean)	Tokens/task ↓	AIP ↑
Qwen2.5-7B	No Memory	82.47	**214,487**	82.37
	A-MEM	82.43	220,716	82.43
	LLMA-Mem	82.70	236,159	82.41
	LoRAM (ours)	**82.90**	227,851	**82.72**
GPT-4o-mini	No Memory	86.01	**144,189**	85.79
	A-MEM	86.38	173,983	86.82
	LLMA-Mem	87.38	175,018	86.79
	LoRAM (ours)	**87.62**	147,511	**87.30**

Table 2: Performance across three role-structured MultiAgentBench environments using GPT-4o-mini. Each column reports the benchmark's primary task metric.

Method	Research J ↑	Database Acc. ↑	Coding Score ↑
No Memory	86.01	32.00	70.28
A-MEM	86.38	32.00	70.14
LLMA-Mem	87.38	35.00	69.01
LoRAM	**87.62**	**40.00**	**71.51**

Cumulative memory advantage. Figure 3 shows when performance differences relative to No Memory accumulate along the ordered task streams. LoRAM finishes with the largest CMA in each environment. Research shows a gradually increasing advantage, whereas Database exhibits larger fluctuations. In Coding, A-MEM's early advantage diminishes and LLMA-Mem ends below No Memory, while LoRAM retains a positive cumulative margin. With the same task count and scoring rule, final CMA equals T times the difference in mean scores; it is not independent evidence beyond the corresponding means. The trajectories describe the timing of observed gains but do not establish that the router improves with experience or rule out gains concentrated on a few tasks.

5.3 Routing-Design Ablations

The ablations compare routing designs rather than perform an exhaustive component sweep. The LoRAM variants share the native memory representation and maintenance pipeline, varying how cross-agent memory is made available and selected; we additionally evaluate adapted item routing:

- **LoRAM-Local**: restricts the action space to $\{\mathrm{bypass}, \mathrm{local}\}$, removing cross-agent transfer.
- **LoRAM-Shared**: pools peer memories into a single unstructured source, removing role-aware source preservation and selection.
- **Adapted Item-Routing**: selects from a flat candidate pool retaining ownership and role metadata under the same 1,000-token cap (Appendix B.1).
- **LoRAM-StaticRole**: retains role-compatible source filtering but uses a fixed routing policy, removing online adaptation from downstream outcomes.
- **LoRAM**: the full role-constrained online memory-source router.

LoRAM-Local provides a local-only comparison (RQ2), while LoRAM-StaticRole and LoRAM-Shared provide static-role and unstructured shared-access comparisons; adapted item routing supplies an additional comparison for RQ3. These variants differ in their available source sets or routing policies, so their performance contrasts should not be interpreted as isolated causal effects of individual components.

Ablation results. Table 3 shows that LoRAM-Local reaches 87.05 on Research, while LoRAM-Shared and LoRAM-StaticRole reach 87.12 and 87.34, respectively. Adapted Item-Routing reaches

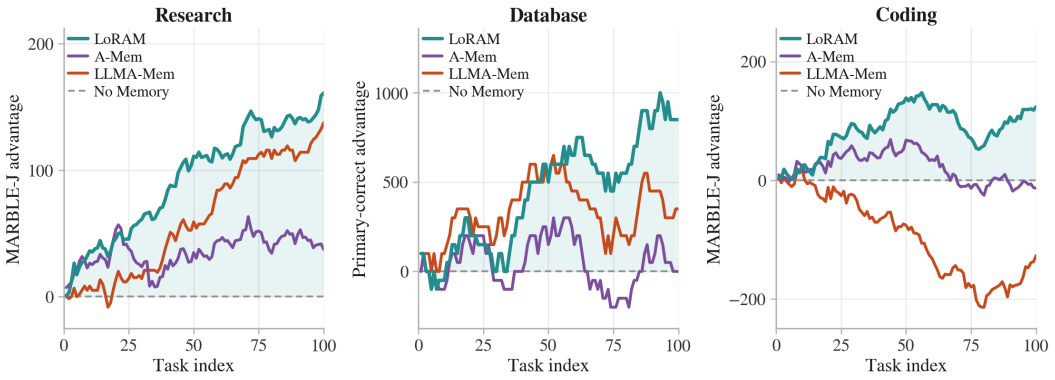

Figure 3: Cumulative memory advantage (CMA) on GPT-4o-mini across the Research, Coding, and Database environments. Each panel compares No Memory, A-MEM, LLMA-Mem, and LoRAM. Higher CMA indicates a larger accumulated performance gain over the No-Memory baseline along the task sequence.

Table 3: Routing-design ablations on 100 Research tasks with GPT-4o-mini. Final CMA equals $100(J - J_{\text{NoMem}})$ for this stream and restates the mean-score difference. Higher is better for J and final CMA; lower is better for token count.

Variant	$J \uparrow$	Final CMA $\uparrow$	Tokens/task $\downarrow$
LoRAM-Local	87.05	+104	142k
LoRAM-Shared	87.12	+111	162k
Adapted Item-Routing	87.28	+127	158k
LoRAM-StaticRole	87.34	+133	151k
LoRAM (full)	**87.62**	**+161**	148k

87.28, compared with 87.62 for full LoRAM, and uses 158k rather than 148k tokens per task (approximately 6.8% more, using rounded counts). Full LoRAM thus achieves a higher mean score with lower token use than this adapted baseline in the evaluated stream. Both methods use the same role-compatibility gate; this comparison does not isolate learned scoring from access-mode selection or budget allocation. Full LoRAM uses more tokens than LoRAM-Local. Peer access (cross or hybrid) increases from 20.0% to 63.2% across the first and last task windows (Appendix F); these frequencies do not isolate policy learning.

6 Conclusion

We introduced LoRAM, which formulates persistent multi-agent memory reuse as online, task-conditioned source routing over agent-owned memories. Rather than fixing memory access through a local, shared, or hybrid organization, LoRAM selects whether memory should be used and which agent-owned source should supply it for each agent-task pair.

Across the Research, Database, and Coding environments of MultiAgentBench, LoRAM achieves the highest observed mean scores among the evaluated memory configurations, with gains of different magnitudes across environments. The Research ablations are consistent with benefits from the full routing design over local-only, unstructured shared, and static-role alternatives. Their generalization to additional task orders and random seeds remains open.

More broadly, our results motivate treating multi-agent memory not only as a problem of what information to store and retrieve, but also as a problem of deciding *whose* experience should be reused for a given agent and task.

AI USE STATEMENT

In this work, we used generative AI tools for editing the manuscript to improve readability and grammatical correctness. We have thoroughly reviewed all AI-assisted text, and we take full responsibility for the final content of this work, including all text, claims, and results.

ETHICS STATEMENT

This work studies memory routing among cooperating LLM-based agents on publicly available multi-agent benchmarks. No human-subjects data are collected. The evaluation stack (MultiAgentBench, GPT-4o-mini API, Qwen2.5-7B-Instruct served locally) is used within the terms of its respective licenses. We do not release trained models or user data; the released code exposes only routing configurations, memory schemas, and evaluation scripts. Potential misuse considerations of memory-augmented multi-agent systems (e.g., accumulation of unsafe procedures) are addressed through the compatibility gate and the bounded-maintenance mechanisms described in §4.4.

REPRODUCIBILITY STATEMENT

To support reproducibility, §5.1 specifies the benchmark, backbones, baselines, and evaluation metrics; Appendix B collects the full implementation details, including Table 4 with all fixed LoRAM hyperparameters; Algorithm 1 describes the end-to-end execution loop; and §4.1–§4.4 document the memory schema, transactive directory, candidate construction, online router, and consolidation pipeline. Appendix E specifies judge settings, scoring rubrics, scale conventions, and evaluation provenance. The failure-handling rule is specified in Appendix B.7. We will release the memory schema, router implementation, evaluation scripts, and routing logs as anonymous supplementary material.

REFERENCES

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1575. URL https://aclanthology.org/2025.acl-long.1575/.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. MAGMA: A multi-graph based agentic memory architecture for AI agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36848–36865, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1709. URL https://aclanthology.org/2026.acl-long.1709/.

Dongming Jiang, Yi Li, Guanpeng Li, Qiannan Li, and Bingzhe Li. Hage: Harnessing agentic memory via rl-driven weighted graph evolution, 2026b. URL https://arxiv.org/abs/2605.09942.

Dongming Jiang, Yi Li, Songtao Wei, Jinxin Yang, Ayushi Kishore, Alysa Zhao, Dingyi Kang, Xu Hu, Feng Chen, Qiannan Li, and Bingzhe Li. Anatomy of agentic memory: Taxonomy and

empirical analysis of evaluation and system limitations, 2026c. URL https://arxiv.org/abs/2602.19320.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pp. 661–670. ACM, April 2010. doi: 10.1145/1772690.1772758. URL http://dx.doi.org/10.1145/1772690.1772758.

Jun Liu, Zhenglun Kong, Changdi Yang, Fan Yang, Tianqi Li, Peiyan Dong, Joannah Nanjekye, Hao Tang, Geng Yuan, Wei Niu, Wenbin Zhang, Pu Zhao, Xue Lin, Dong Huang, and Yanzhi Wang. Rcr-router: Efficient role-aware context routing for multi-agent llm systems with structured memory, 2025. URL https://arxiv.org/abs/2508.04903.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of llm agent memory mechanisms, 2026. URL https://arxiv.org/abs/2605.06716.

Wenquan Ma, Jiayan Nan, and WenLong Wu. What deserves memory: Adaptive memory distillation for LLM agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34789–34812, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1607. URL https://aclanthology.org/2026.acl-long.1607/.

Bodhisattwa Prasad Majumder, Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. CLIN: A continually learning language agent for rapid task adaptation and generalization, 2024. URL https://openreview.net/forum?id=d5DGVHMdsC.

Yanki Margalit, Nurit Cohen-Inger, Erni Avram, Ran Taig, and Oded Margalit. Governed shared memory for multi-agent llm systems, 2026. URL https://arxiv.org/abs/2606.24535.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun,

Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. URL https://arxiv.org/abs/1908.10084.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

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. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Yiqi Wang, Zihao Yan, Jiaqi Zhang, Zhangkai Wu, Mingkai Zheng, Zequn Sun, Yanming Zhu, and Taotao Cai. Map-graph: Provenance-aware shared memory for multi-agent workflows, 2026. URL https://arxiv.org/abs/2608.10509.

Yu Wang and Xi Chen. Mirix: Multi-agent memory system for llm-based agents, 2025. URL https://arxiv.org/abs/2507.07957.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=NTAhi2JEEE.

Daniel M. Wegner. *Transactive Memory: A Contemporary Analysis of the Group Mind*, pp. 185–208. Springer New York, New York, NY, 1987. ISBN 978-1-4612-4634-3. doi: 10.1007/978-1-4612-4634-3_9. URL https://doi.org/10.1007/978-1-4612-4634-3_9.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory, 2025. URL https://arxiv.org/abs/2410.10813.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Shanglin Wu, Yuyang Luo, Yueqing Liang, Kaiwen Shi, Yanfang Ye, Ali Payani, and Kai Shu. Scaling teams or scaling time? memory enabled lifelong learning in llm multi-agent systems, 2026. URL https://arxiv.org/abs/2604.03295.

Naen Xu, Hengyu An, Shuo Shi, Jinghuai Zhang, Chunyi Zhou, Changjiang Li, Tianyu Du, Zhihui Fu, Jun Wang, and Shouling Ji. When agents "misremember" collectively: Exploring the mandela effect in llm-based multi-agent systems, 2026. URL https://arxiv.org/abs/2602.00428.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. Comas: Co-evolving multi-agent systems via interaction rewards, 2026. URL https://arxiv.org/abs/2510.08529.

Chenfei Yan, Zeyang Yue, Feifei Zhao, Erliang Lin, Lu Jia, Haibo Tong, Mingyang Lyu, Chengyi Sun, and Yi Zeng. When truth is distributed: Misinformation derails collective fact recovery in llm-based multi-agent systems, 2026a. URL https://arxiv.org/abs/2608.03421.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

Hanqing Yang, Jingdi Chen, Marie Siew, Tania Lorido-Botran, and Carlee Joe-Wong. Llm-powered decentralized generative agents with adaptive hierarchical knowledge graph for cooperative planning, 2025a. URL https://arxiv.org/abs/2502.05453.

Jingbo Yang, Kwei-Herng Lai, Xiaowen Wang, Shiyu Chang, Yaar Harari, and Evgeniy Gabrilovich. Groupmembench: Benchmarking llm agent memory in multi-party conversations, 2026. URL https://arxiv.org/abs/2605.14498.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems, 2025b. URL https://arxiv.org/abs/2504.00587.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=yGOytgjurF.

Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. Exploring collaboration mechanisms for LLM agents: A social psychology view. 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. 14544–14607, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.782. URL https://aclanthology.org/2024.acl-long.782/.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents, 2024b. URL https://arxiv.org/abs/2404.13501.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024. URL https://arxiv.org/abs/2308.10144.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. Multiagentbench: Evaluating the collaboration and competition of llm agents, 2025. URL https://arxiv.org/abs/2503.01935.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=syG9I9ofd8.

A LIMITATIONS

Our evaluation uses one ordered stream of 100 tasks per environment, so small differences between methods should not be interpreted as evidence of stability across random seeds or task orders. We evaluate one adapted provenance-aware item-routing baseline, which does not cover the broader range of item-ranking methods; the reported aggregate results do not identify individual negative-transfer events. The router observes a team-level outcome rather than an isolated reward for each agent's memory decision, making per-agent credit assignment approximate. In Research, semantic anchors are constructed from agent profiles across the evaluation configurations before the task stream begins. This uses no outcome labels, but assumes those profiles are available in advance. The experiments span three environments and two backbones and do not establish generalization to other team structures or longer streams.

B REPRODUCIBILITY AND IMPLEMENTATION DETAILS

This appendix collects the implementation details, fixed hyperparameters, and protocol definitions summarized in §5.1. The conceptual description of each mechanism remains in §4; what follows records the concrete settings used to run it.

B.1 HYPERPARAMETERS

Table 4 gives the fixed LoRAM configuration. The values reported here are held constant across the evaluated environments and backbones.

Table 4: Fixed LoRAM hyperparameters (from the canonical implementation spec), held constant across environments and backbones.

Component	Parameter	Value
Role gate	threshold θ_{role}	0.55
	compat. prior strength c	4.0
Retrieval / budget	top-K per source	5
	shared token budget B	1000
Online router	UCB coefficient λ_{UCB}	0.25
	trust prior	$\text{Beta}(1, 1)$
Consolidation	interval N	2 tasks
	min. cluster support	2 episodes
	cluster cosine threshold	0.8

Baseline settings. A-MEM uses `all-MiniLM-L6-v2` (Reimers & Gurevych, 2019) for embeddings, with retrieval $k = 5$ and an evolution threshold of 100. The memory consolidation interval is set to $N = 2$ tasks for the LLMA-Mem and A-MEM baselines, matching LoRAM's default so that the three memory-enabled configurations share a consolidation schedule.

Adapted item-routing baseline. This Research comparison uses the same role-compatibility gate as LoRAM: peer source a_j is eligible when $\text{UCB}[\text{Compat}(r_j, r_i)] \geq 0.55$. The target agent pools its local episodes and procedures with eligible peer procedures,

$$\mathcal{M}_{\text{pool}} = (E_i \cup P_i) \cup \left(\bigcup_{j \in \mathcal{E}} P_j \right),$$

retaining ownership and role metadata. Peer episodes remain private. Items are ranked by

$$\begin{aligned} S(m, q_t) = {} & 0.45\, \text{Sim}(m, q_t) + 0.25\, \hat{\rho}_m \\ & + 0.20\, \hat{\mu}_{\text{Compat}}(r_{\text{src}}, r_i) + 0.10\, \mathbf{1}[m \in M_i], \end{aligned}$$

where Sim is cosine similarity from `all-MiniLM-L6-v2`, $\hat{\rho}_m$ is Laplace-smoothed reliability, $\hat{\mu}_{\text{Compat}}$ is the role-compatibility posterior mean, and the indicator gives local items a fixed preference. Items are greedily packed in score order under the same $B = 1000$ token cap. The coefficients are fixed, whereas LoRAM uses an outcome-adaptive policy over memory access modes and sources. This is an adapted item-routing comparison, not a claimed reproduction of RCR-Router. Because role gating is shared, the comparison cannot attribute a performance difference to the presence of the gate. It also does not isolate the effects of learned scoring, source selection, or candidate-pool construction.

B.2 Memory Schema and On-Disk Layout

Episodic memory E_i stores task-level experience $e = \langle q, r_i, \tau, o, \text{TS}, \text{CS}, l, t\rangle$ with an LLM-extracted lesson l; its router importance is assigned as 0.75 for successful tasks and 0.25 otherwise, with an evidence support of 1. *Procedural memory* P_i stores consolidated reusable strategies $p = \langle id, \mathit{owner}, r_{\text{src}}, \mathit{title}, \mathit{content}, \tau, s, f\rangle$ with success/failure counts (s, f). The router quantifies its importance via Laplace-smoothed reliability $\hat{\rho} = (s + \alpha)/(s + f + 2\alpha)$ with $\alpha = 1$ (avoiding feature saturation on small sample sizes), and its evidence support is $s + f$. Consolidation (§4.4) distills repeated successful episodes into procedures. *Role profile memory* R_i tracks the agent's per-type proficiency $\Gamma_i(\tau)$ as a Beta posterior over task successes in the directory.

Each agent a_i maintains a native on-disk store serialized as a JSON document per role (e.g., `systems_language_agents_store.json`). Unlike unified architectures, episodic and procedural memories are instantiated via distinct, specialized schemas (`marble/memory/llma_mem/schemas.py`):

- **Episodic Records** (E_i): Store raw historical trajectories and are *strictly local* (never shared across agents). Each record contains: `agent_id` (string, bound to semantic role), `episode_id` (UUID), `timestamp` (float), `task_description` (string), `team_composition` (list), `actions_taken` (list), `outcome` (string), `outcome_success` (bool), `outcome_metrics` (dict), `outcome_error_info` (optional), `context_state` (dict carrying `task_type` and `semantic_role`), `lessons_learned` (string), `related_procedures` (list of IDs), `collaboration_quality` (float), and `embedding` (vector).
- **Procedural Records** (P_i): Store consolidated strategies eligible for *cross-agent transfer*. Each record contains: `agent_id` (string), `procedure_id` (UUID), `timestamps` (list of floats), `timestamp_created` (float), `timestamp_updated` (float), `title` (string), `type` (string), `knowledge_content` (string), `prerequisites` (list), `success_count` (s), `failure_count` (f), `success_rate` ($s/(s+f)$), `avg_performance_gain` (float), `confidence_score` ($\hat{\rho}$), `source_episodes` (list of contributing UUIDs), `last_used` (float), `last_used_episode` (string), `viewed_times` (int), and `embedding` (vector).

Agent proficiencies R_i reside within the Transactive Directory graph state (`GraphState.proficiency`) rather than local agent files, recording Beta distributions over task types.

B.3 Task Typing and Anchor Discovery

To provide meaningful sub-task granularity without manual annotation or extra LLM calls, LoRAM maps each task description to a shared semantic anchor space via deterministic embedding lookup (`all-MiniLM-L6-v2`). For Research, anchors are data-driven: we pool agent profiles across all 100 task configurations and discover $K = 10$ cluster centroids labeled by their top keywords (e.g., `systems_applications_privacy`, `gnns_graphs_tasks`). Because personas and tasks share this semantic vocabulary, task typing τ_t and role assignments operate on the same space. In Database and Coding, where task-specific anchor sets are not pre-indexed, typing naturally falls back to matching task embeddings against the active team's agent profiles, mapping τ_t to the most relevant role slot. No ground-truth outcome or diagnosis label is ever exposed to the router.

Discovered Anchors in Research. Rather than imposing subjective taxonomies, `scripts/lifelong/build_role_anchors.py` pools all agent persona descriptions across the 100 Research benchmark tasks and applies K-means clustering ($K = 10$) over their normalized embeddings. Each centroid is automatically named by its highest-frequency non-stopword tokens, generating 10 canonical expertise anchors:

1. `systems_understanding_theoretical`
2. `gnns_graphs_tasks`
3. `applications_performance_image`
4. `systems_accuracy_xai`
5. `systems_applications_privacy`
6. `systems_language_agents`
7. `language_performance_recent`
8. `systems_environments_robots`
9. `medical_imaging_performance`
10. `graph_networks_applications`

In Research, agent personas rotate per task. Both role assignment (`assign_role`) and task typing (`type_task`) share this identical anchor embedding space, ensuring that τ_t and r_i map to unified semantic coordinates.

Fallback in Database and Coding. In Database and Coding, tasks are executed by a fixed set of persistent agent slots (e.g., Coordinator, Executor). With no dedicated anchor dictionary specified, `TransactiveGraph.type_task` falls back to computing cosine similarities directly against the active team's persona profile embeddings, producing slot-level assignments (`agent1` through `agent5`). This guarantees that the system executes deterministically across all environments without leaking benchmark ground-truth labels.

B.4 Routing Feature Parameterization

Candidate actions $a \in \mathcal{A}_{i,t}$ are encoded into normalized vectors $x_{i,t,a}$ using the router's exact feature pipeline (`router.py`):

- **Semantic Relevance**: Mean cosine similarity between the retrieved candidate embeddings and the task prompt q_t, scaled to $[0, 1]$.
- **Memory Importance**: For episodic records, importance is set to 0.75 if `outcome_success` is true and 0.25 otherwise. For procedural records, importance is the Laplace-smoothed confidence score $\hat{\rho} = (s + 1)/(s + f + 2)$, dampening initial variance.
- **Source Expertise**: The posterior mean $\alpha/(\alpha + \beta)$ of the source agent's proficiency $\Gamma_o(\tau_t)$ on task type τ_t.
- **Directed Transfer History**: The posterior mean of past transfer success from source j to target i, initialized with a uninformative $\mathrm{Beta}(1, 1)$ prior.
- **Evidence Support**: Sample count n, scaled via log-compression $\min(1.0, \log(1 + n)/\log(10))$. Here $n = 1$ for episodic records and $n = s + f$ for procedural records.
- **Retrieval Cost Penalty**: Ratio of total candidate tokens to shared memory budget B, negated.
- **Peer Novelty**: Minimum cosine distance between peer procedure embeddings and the receiver's local memory store.
- **Action Type Indicator**: One-hot encoding spanning bypass, local, cross, and hybrid.

For the bypass action, all candidate-dependent features are set to 0, cost penalty is 0, and only the bypass indicator is active. Online LinUCB policies initialize with covariance $A_{i,0} = I_d$ and bias vector $b_{i,0} = \mathbf{0}$.

B.5 MEMORY CONSOLIDATION

Every N tasks, each agent's successful episodes are grouped by (task type, role), lessons are extracted with the same LLM backbone as the agents, embedded with `all-MiniLM-L6-v2` (Reimers & Gurevych, 2019), and greedily clustered at cosine threshold 0.8. Clusters with at least two successful episodes are distilled into a role-tagged procedure, linked into G_T with its `owns` and `supports` edges. This same pipeline runs identically for every agent, so consolidation grounds a transferable procedure in repeated, role-specific success before it can be routed across agents. To keep the stores bounded, LoRAM applies confidence-based pruning, duplicate merging, and per-key capacity bounds. These are bounded-maintenance mechanisms, not a learned temporal-decay or long-term retention policy, as discussed under Limitations.

The consolidation interval is $N = 2$ tasks; other fixed settings appear in Table 4.

To maintain bounded storage: (1) procedures whose reliability drops below $\rho_{\min} = 0.35$ over at least 3 evaluations are pruned; (2) candidate procedures exceeding a cosine similarity of 0.85 with an existing entry are merged via support-weighted averaging; (3) episodic stores are bounded to 50 items and procedural stores to 20 items per agent using least-recently-used (LRU) eviction on reliability ties.

B.6 PROMPT AND CONTEXT CONSTRUCTION

Selected memories are rendered into compact cards that label their owner and role, preserving provenance into the prompt. Every memory-access mode receives the same total memory-token budget B; a hybrid action splits that budget between local and peer cards, so modes differ in which source supplies context rather than in how much context they are allowed.

Retrieved memories are serialized into compact text blocks: `[Memory Card | Source:` Agent j `(`r_j`) | Type:` τ `| Reliability:` ρ`] Strategy: <content>`. These cards are inserted into the system prompt within an `<experience>` block preceding the current task description. If the total length of selected cards exceeds token budget $B = 1000$, cards are greedily truncated in reverse order of relevance.

B.7 FAILURE HANDLING

A run that fails to complete is scored with a pre-specified value rather than being dropped, so that the cumulative moving average AP_t and the scalar AIP are computed over the full task stream (§5.1). The same failure-handling rule and the same common-task set are applied to every method so that the reported task means and cumulative metrics use consistent denominators. An uncompleted task receives score 0 rather than being omitted from the sequence.

B.8 ENVIRONMENT-SPECIFIC SETUP

Backbone serving. Qwen2.5-7B-Instruct is served locally with vLLM 0.6.6 (Kwon et al., 2023) in `bfloat16`, tensor-parallel-size 2 on 2× RTX A5000, with a 32K context window and the Hermes tool-call parser. GPT-4o-mini (OpenAI et al., 2024) is accessed through the Azure API. Both backbones receive identical prompts and identical retrieval depth, consolidation interval, and evolution settings.

Research. Research uses 100 collaborative research-idea-generation tasks, five role-differentiated agents, a five-question ("5q") output format, at most three coordination iterations under a graph topology, and a 30-minute wall-clock timeout.

Database and Coding. In the *Database* setting, 5 agents (Coordinator, Query Optimizer, Index Specialist, Concurrency Controller, Hardware Tuner) collaborate over 100 diagnosis cases with a 15-minute timeout, scored on root-cause identification accuracy. In the *Coding* setting, 4 agents (Architect, Programmer, Code Reviewer, Tester) collaborate across 100 algorithmic and system-design problems with a 20-minute timeout, evaluated by the four-dimension code-quality rubric in Appendix E.

B.9 End-to-End Execution Loop

Algorithm 1 Role-Aware Transactive Memory Routing

Require: Tasks $Q = \{q_1, \ldots, q_T\}$, agents A, native stores $\{M_i = \langle E_i, P_i \rangle\}$, directory G_T, per-agent policy states $\{(A_{i,0}, b_{i,0})\}$, budget B, top-K, gate θ_{role}, interval N

for each task $q_t \in Q$ **do**
  $\tau_t \leftarrow \textsc{TaskTyper}(q_t)$ {deterministic embedding lookup; no LLM call}
  for each agent $a_i \in A$ **do**
    $C_{\text{local}} \leftarrow \textsc{TopK}(M_i, q_t)$ {retrieval from own episodes and procedures}
    $\mathcal{E} \leftarrow \{a_j \neq a_i : \text{UCB}[\text{Compat}(r_j, r_i)] \geq \theta_{\text{role}}\}$ {role gate}
    $C_{\text{cross}}(j) \leftarrow \textsc{TopK}(P_j, q_t)$ for each $a_j \in \mathcal{E}$ {procedures only; source-preserving}
    $\mathcal{A}_{i,t} \leftarrow \{\text{bypass}, \text{local}\} \cup \{\text{cross}(j), \text{hybrid}(i,j) : a_j \in \mathcal{E}\}$
    $x_{i,t,a} \leftarrow$ normalized action features; compute $U_{i,t}(a)$ with Eq. 4
    $a_i^\star \leftarrow \arg\max_{a \in \mathcal{A}_{i,t}} U_{i,t}(a)$ {one budget B; hybrid splits B}
  end for
  Render provenance-labeled cards for each $a_i^\star$; build prompts; execute task
  Observe TS_t, CS_t, cost; form normalized reward r_t
  for each agent a_i **do**
    Update $(A_{i,t}, b_{i,t})$ using Eq. 5 {selected action only}
    Update selected-source proficiency and directed transfer history
  end for
  if $t \bmod N = 0$ **then**
    Consolidate each agent's successful episodes into procedures; prune/merge/bound; update G_T
  end if
end for

C Additional Backbone Analysis

Backbone sensitivity. Memory gains are smaller with Qwen2.5-7B than with GPT-4o-mini in our setting. For example, LLMA-Mem changes the Research score from 82.47 to 82.70 on Qwen2.5-7B, whereas the corresponding gain is larger on GPT-4o-mini. LoRAM also achieves only a modest improvement over LLMA-Mem on Qwen2.5-7B (82.90 vs. 82.70). With only two backbones, we treat this as a robustness observation rather than evidence of a monotonic relationship between model capability and memory benefit.

D Extended Related Work

Memory and experiential learning in LLM agents. A large body of work equips a single LLM agent with persistent experience so that it improves across episodes rather than within one context window. Early designs store verbal self-criticism or skills induced from environment interaction (Shinn et al., 2023; Wang et al., 2024), and later work abstracts trajectories into reusable insights and learned experience summaries (Zhao et al., 2024; Majumder et al., 2024) or into explicitly reusable procedural routines (Wang et al., 2025). A parallel thread treats memory as a resource to be managed: paging between context tiers (Packer et al., 2024), hierarchical working-memory compression over long horizons (Hu et al., 2025), agentic linking and evolution of stored notes (Xu et al., 2025), learned admission and distillation of what is worth keeping (Ma et al., 2026), and reinforcement learning over memory-management operations (Yan et al., 2026b). A third thread explores structured memory and learned retrieval over relational representations, including multi-graph organizations and reinforcement-learned traversal policies (Jiang et al., 2026a;b). Benchmarks, surveys, and empirical analyses have consolidated this space around representation, storage, retrieval, update, and evaluation (Wu et al., 2025; Zhang et al., 2024b; Luo et al., 2026; Jiang et al., 2026c), while GroupMemBench extends memory evaluation to multi-party conversational settings (Yang et al., 2026). What these systems learn is increasingly sophisticated—what to store, how to structure it, and which memories to retrieve or traverse—but the learned decision is still primarily over items within an already available memory scope. When reusable experience is instead distributed across

role-specialized agents, an additional question appears: *whose* experience should be retrieved for a given target?

Multi-agent memory and distributed experience. Work on multi-agent memory makes persistent experience available beyond the agent that produced it. LLMA-Mem (Wu et al., 2026) is the closest architectural reference point for our setting: it studies lifelong multi-agent learning under local, shared, and hybrid memory configurations and reports a tension between fragmenting knowledge and exposing agents to cross-role interference. Other systems organize team memory through structured or governed shared state, including typed memory hierarchies coordinated by dedicated memory agents (Wang & Chen, 2025), provenance- and permission-aware shared graphs (Wang et al., 2026), explicit governance of shared-memory scope (Margalit et al., 2026), and decentralized per-agent knowledge graphs exchanged through structured protocols (Yang et al., 2025a). Related work on decentralized coordination lets agents specialize and evolve their connectivity or policies from interaction (Yang et al., 2025b; Xue et al., 2026), although the adapted object there is coordination structure or agent capability rather than persistent memory-source selection. MultiAgentBench and its MARBLE framework (Zhu et al., 2025) provide the role-structured environments and milestone-based collaboration metrics used in our evaluation; they are a benchmark and execution framework rather than a memory architecture. Evidence that cross-agent propagation can be harmful also comes from studies of collective error: multi-agent systems can amplify false details beyond what any single model holds (Xu et al., 2026), while injected misinformation can disrupt collective fact recovery once it enters communication (Yan et al., 2026a). These mechanisms differ from the role-incompatible transfer studied here, but they reinforce the broader point that unrestricted peer exposure is not always beneficial.

Role-aware context routing and source selection. Closest to our formulation is work that varies what a given agent receives rather than only where memory is stored. RCR-Router (Liu et al., 2025) routes context by scoring items in a centralized shared memory store against the target agent's role and current task stage, then greedily filling a token budget; agent outputs are incorporated back into the shared store. This routing is genuinely dynamic and role-conditioned. The distinction we draw is in the unit of the decision: RCR-Router performs *item-level* routing within a common memory pool, whereas LoRAM performs *source-level* routing across independently owned persistent memories. For each agent task pair, LoRAM decides whether long-term memory should be used at all and, if so, whether the source should be local memory, a particular compatible peer, or a budgeted combination of the two. Peers are not merged into a common pool before source selection, and source utility is updated online from downstream task outcomes. Item routing and source routing are therefore complementary: one determines which entries to expose within a memory scope, while the other determines which ownership source should supply the final memory context.

Dynamic information routing has also been studied outside persistent memory. Conversation and workflow frameworks structure who communicates with whom (Wu et al., 2024; Hong et al., 2024), analyses of collaboration study how roles and interaction structures shape team outcomes (Zhang et al., 2024a), and other systems transfer agent state at the representation or KV-cache level (Zou et al., 2026; Ye et al., 2025). These methods operate primarily on messages, activations, or transient state within a task, whereas LoRAM targets ownership and reuse of consolidated experience across tasks.

Transactive memory and online decision making. The notion that a group encodes "who knows what" as shared meta-memory, and retrieves through the member who owns the relevant knowledge, originates in organizational psychology (Wegner, 1987). We use this concept as design vocabulary rather than as a novelty claim: memory content remains with its owner, while a lightweight directory maintains the metadata needed for routing, including ownership, per-type expertise, role compatibility, and directed transfer history. Selecting among sources of uncertain utility, with feedback observed only for the source actually used, naturally connects this problem to contextual-bandit methods (Li et al., 2010; Auer et al., 2002). LoRAM therefore treats access-mode and source selection as an online contextual decision with optimism-based exploration, updating the policy from observed feature–reward pairs rather than assigning counterfactual rewards to unobserved alternatives.

E Evaluation Protocol and Rubrics

E.1 Judge Configuration and Provenance

LLM-based rubric evaluation uses GPT-4o-mini through Azure OpenAI, with deployment name `gpt-4o-mini` and API version `2024-12-01-preview`. These are the values of `AZURE_OPENAI_DEPLOYMENT_NAME` and `AZURE_OPENAI_API_VERSION`, respectively. The API version specifies the service interface, not the underlying model snapshot; the deployment name alone does not identify an immutable model version. All judge calls use temperature 0.0, leave `top_p` unset, and set `max_tokens` to 512, except for the code-quality rubric, which uses 4096. Prompts require JSON-only output, with no chain-of-thought and no few-shot examples. The judge model is recorded separately from the policy model that executes the task. Each run writes `eval_provenance.json`, recording the judge, policy model, temperature, and scoring scale.

E.2 Rubric Dimensions and Aggregation

Every rubric dimension is rated on a 1–5 Likert scale. A rubric score is the arithmetic mean of its dimensions. The four rubrics are:

- **Research task quality**: Innovation, Safety, and Feasibility.
- **Coding task quality**: Instruction-Following, Executability, Consistency, and Quality.
- **Planning**: Role Clarity, Task Alignment, and Autonomy.
- **Communication**: Information Exchange, Clarity, Task Assistance, and Efficiency.

Let p_t and c_t denote the Planning and Communication means, averaged over available iteration-level evaluations for task t. The raw coordination score is $(p_t + c_t)/2$: the two rubrics receive equal weight, despite having different numbers of dimensions. Research task quality is the mean of its three dimensions, and Coding task quality is the mean of its four dimensions. Research reports the joint score $J_t = (\mathrm{TS}_t + \mathrm{CS}_t)/2$; the Coding column reports task quality. Database accuracy is a separate diagnosis metric and is not a Likert score. The iteration-level sentinel -1 denotes unavailable rubric feedback and is omitted from the corresponding mean; it is not a valid rating of 1. The task-failure convention is specified in Appendix B.7.

E.3 Score Scales and Comparability

For a valid rubric mean $x \in [1, 5]$, the MARBLE convention used for the reported rubric scores is

$$S_{20} = 20x \in [20, 100].$$

The attainable lower bound is 20, not zero. For comparison, the alternative affine mapping to the full $[0, 100]$ interval is

$$S_{\mathrm{aff}} = 25(x - 1) = 1.25S_{20} - 25 \in [0, 100].$$

This relationship also holds for equal-weight averages of valid scaled rubric scores, including Research J. The affine mapping is a reporting conversion; it does not specify a change to the router's training reward. For the same paired observations transformed uniformly, score rankings are unchanged and each paired difference is multiplied by 1.25. Consequently, paired t-tests and signed-rank tests have unchanged p-values under this positive affine transformation, before rounding. This is an algebraic property, not an additional significance result.

Failure scores require separate treatment. A zero assigned to a failed task is a penalty outside the valid rubric range $[20, 100]$. If failures remain zero under both reporting conventions while valid scores are transformed, the overall transformation is not globally affine; aggregate rankings and paired-test results are not guaranteed to be identical. In that case, alternative-scale means and tests must be recomputed from the task-level scores and failure indicators. No such invariance is asserted for the failure-penalized task streams. Database accuracy is unchanged by rubric rescaling.

F ROUTING-DECISION LOGGING PROTOCOL

The routing-action analysis covers the 100-task Research stream with five agents per task and GPT-4o-mini through Azure OpenAI (deployment `gpt-4o-mini`, API version `2024-12-01-preview`). Policy and judge calls in this analysis use temperature 0.0. Before each task executes, the contextual policy selects one action for each agent, yielding 500 decisions across the stream. Each non-overlapping 25-task window therefore contains 125 decisions. Tasks that subsequently fail or time out remain in the window: their five pre-task routing decisions are included, while their task scores follow the zero-penalty convention in Appendix B.7.

For action $a \in \{\text{bypass}, \text{local}, \text{cross}, \text{hybrid}\}$, let $n_{w,a}$ be its integer count in window w. Its percentage is

$$p_{w,a} = 100\frac{n_{w,a}}{125}, \qquad \sum_a n_{w,a} = 125.$$

Thus the percentages are multiples of 0.8 percentage points. Table 5 reports the integer counts and corresponding percentages.

Table 5: Routing-action counts and percentages on Research with GPT-4o-mini. Each 25-task window contains 125 pre-execution agent decisions; the full stream contains 500. Decisions from tasks that subsequently fail remain included.

Task window	Bypass	Local	Cross	Hybrid	Total
1–25	23 (18.4%)	77 (61.6%)	10 (8.0%)	15 (12.0%)	125
26–50	15 (12.0%)	58 (46.4%)	20 (16.0%)	32 (25.6%)	125
51–75	10 (8.0%)	47 (37.6%)	28 (22.4%)	40 (32.0%)	125
76–100	6 (4.8%)	40 (32.0%)	31 (24.8%)	48 (38.4%)	125
All tasks	54 (10.8%)	222 (44.4%)	89 (17.8%)	135 (27.0%)	500

In Tasks 1–25, local access accounts for 61.6% of decisions and bypass for 18.4%, while cross and hybrid together account for 20.0%. By Tasks 76–100, cross and hybrid account for 24.8% and 38.4%, respectively, and bypass declines to 4.8%. The observed distribution therefore shifts toward peer access over the stream. This is consistent with increasing use of procedures as memory evolves, but the action counts alone do not measure procedure quality or the causes of bypass decisions. Evolving candidate availability and task composition can change frequencies even without policy learning; attributing the shift to LinUCB updates requires a matched comparison that controls these factors.