TEMPA: TEMPORALLY EXTENDED MEMORY FOR POLICY ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents must learn from online experience to improve their behavior, even though their parameters remain frozen after deployment. Reusing complete episode trajectories as prompt context can incur additional token costs, while estimating individual action values from experience is challenging under sparse and delayed rewards. These challenges motivate policy adaptation through planning over action segments at an intermediate temporal granularity. We introduce **TEMPA (Temporally Extended Memory for Policy Adaptation)**, a gradient-free framework that adapts an agent's policy over actor actions and executable memory segments. It partitions interaction at reward events and stores the resulting **reward-to-reward segments** as executable options. These segments are selected using empirical advantages. A router uses evidence from exploratory trials to choose between executing the selected segment and invoking the LLM actor. TEMPA reuses multi-step behaviors without separate credit estimates for constituent actions and reduces repeated actor inference during memory execution. We formulate memory exploitation as finite-horizon semi-Markov planning and establish a conditional sublinear bound on exploration loss relative to the incumbent policy. Experiments on Jericho and ScienceWorld show that TEMPA achieves higher average scores and stronger late-episode performance. Its total inference token consumption is roughly 56% lower than the mean across the seven baselines. Ablations support the contributions of reward-based segmentation, continuation values, adaptive routing, and persistent execution.

1 INTRODUCTION

Large language model (LLM) agents have shown promising capabilities in interactive tasks by interleaving reasoning and action (Yao et al., 2023; Lin et al., 2023; Zhou et al., 2024). However, these agents still struggle to reliably complete tasks that require many steps in realistic environments such as WebArena (Zhou et al., 2023). Agents that repeatedly interact with the environment should retain useful behaviors and improve their decisions through experience. The challenge is to achieve this improvement without updating the underlying LLM.

Existing approaches incorporate experience through parameter updates or external memory. Reinforcement learning can improve agent policies, as demonstrated by WebRL (Qi et al., 2024), but requires parameter access and a training pipeline. Training-free alternatives retain interaction traces, verbal reflections (Shinn et al., 2023), or reusable workflows (Wang et al., 2024) to guide subsequent action generation. JitRL (Li et al., 2026a) instead retrieves step-level experience to estimate action advantages and directly adjust the LLM's action distribution. These training-free approaches use experience through two main interfaces: prompt context and direct action correction.

The **temporal granularity** of experience reuse also affects inference costs and credit assignment. Repeatedly incorporating complete trajectories into prompts can increase context overhead, while action-level adaptation depends on reliable estimates of individual action values. Such estimation is difficult when rewards arrive only after extended sequences of preparatory actions. JitRL (Li et al., 2026a) addresses this challenge through auxiliary LLM-based step-wise reward evaluation. Even with reliable value estimates, step-level correction requires a new action-selection decision at every step, including within previously observed successful sequences. These limitations motivate an intermediate unit of experience between individual actions and complete trajectories. Such a

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.

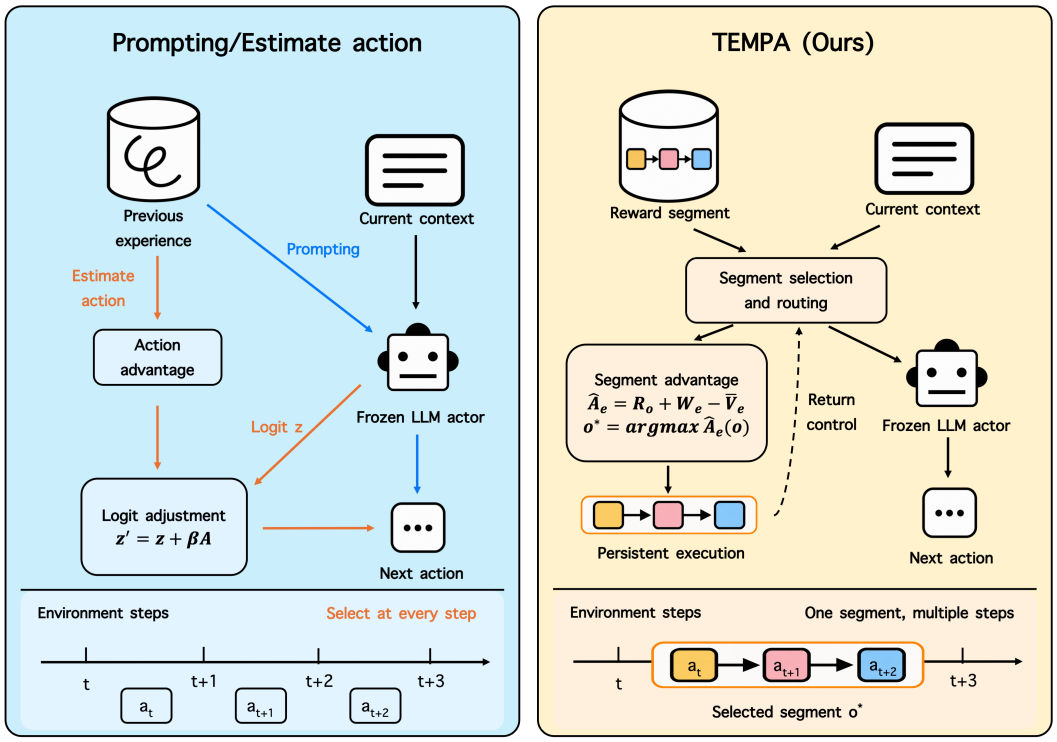


Figure 1: While prompting and action-level correction use past experience to guide individual action decisions (left), TEMPA stores experience as executable reward-to-reward segments (right). Segment advantages guide selection, and the router allocates control between memory and the frozen LLM actor. A selected segment can execute across multiple environment steps before returning control. See Section 3.2 for the detailed computation of segment values and advantages.

unit captures reusable knowledge for carrying out part of a task (Wang et al., 2023; Mi et al., 2026; Li et al., 2026b). Our focus is on using these units consistently across memory storage, value evaluation, and persistent execution.

We introduce **TEMPA—Temporally Extended Memory for Policy Adaptation**. Unlike methods that use memory as prompt context or to adjust action preferences, TEMPA stores experience as executable segments, as illustrated in Figure 1. The framework follows two connected design principles. First, TEMPA uses reward events as observable boundaries for grouping actions with their outcomes. The resulting **reward-to-reward segments** are stored as executable options. This preserves the observed behavior leading to a reward without requiring separate credit estimates for its constituent actions. Second, selecting known behavior and deciding whether to explore require different evidence. An empirical semi-Markov planner (Sutton et al., 1999) evaluates each segment using its own return and the value of its known continuations. The resulting segment advantages guide memory selection. However, these values do not establish whether further exploration is worthwhile. The router therefore tracks how often exploration adds new reusable experience and uses this feedback to choose between memory and the frozen LLM actor. Once selected, a segment retains control until completion, interruption, or a designated handoff. The LLM remains frozen, but the agent's policy changes as its executable memory and decision statistics are updated.

We formalize the method through a segment-level Bellman formulation and analyze exploration loss relative to a reliable incumbent under fixed-context, no-discovery assumptions. Experiments on Jericho (Hausknecht et al., 2020) and ScienceWorld (Wang et al., 2022) show improved adaptation performance and reduced inference-token consumption. Controlled ablations support the complementary roles of experience-dependent routing and persistent execution.

Our contributions are threefold:

- We introduce reward-to-reward segments as executable memory units at an intermediate temporal granularity. This representation aligns experience storage, value evaluation, and execution with observed reward events, preserving multi-step behavior without requiring separate credit estimates for constituent actions.
- We present TEMPA, a non-parametric policy adaptation algorithm for agents with frozen LLMs. Empirical continuation values and segment advantages guide option selection, while exploration statistics adapt control allocation between memory and the actor.
- We evaluate TEMPA on Jericho games and ScienceWorld settings. Results demonstrate improved adaptation performance and reduced token consumption during inference compared with training-free baselines. Controlled ablations examine the roles of adaptive routing, temporal persistence, and option value estimation.

2 Related work

Adaptation from experience. Experience can guide action generation through verbal reflections and reusable insights (Shinn et al., 2023; Zhao et al., 2024), or contextual knowledge of environment dynamics (Liu et al., 2025; Chen et al., 2026). Other approaches revise prompts, memory, and tool-use routines using episode feedback (He et al., 2026), or adjust action distributions through estimated advantages (Li et al., 2026a). However, contextual reuse can increase token costs, while estimating individual action values remains challenging under sparse and delayed rewards.

Experience granularity and executable memory. Reusable workflows and skills provide intermediate behavioral units (Wang et al., 2024; 2023; Mi et al., 2026), while milestone segments support training-time credit assignment (Wang et al., 2026). Executable graphs built through offline exploration and action-group mining use trained value models for path search (Qin et al., 2026). Cached plan templates are adapted across requests to reduce repeated planning (Zhang et al., 2025). TEMPA instead forms executable segments at observed reward boundaries and uses their recorded outcomes and durations for empirical planning.

Policy reuse and temporal control. Temporal abstraction and policy reuse formalize multi-step control and the reuse of learned policies (Sutton et al., 1999; Fernández & Veloso, 2006; Bacon et al., 2016). Milestone selection for LLM agents can also incorporate dependency constraints and bandit statistics (Li et al., 2026b). In TEMPA, segment values guide memory selection, while exploratory trial outcomes guide actor–memory routing. Selected segments retain control across multiple actions without updating the LLM or training a separate value model.

3 Method

TEMPA adapts the agent's policy over actions from the LLM actor π_θ and executable memory options, while keeping θ fixed (Figure 2). Experience updates the available options, their empirical values, and the statistics used to allocate control. Algorithm 1 in Appendix A summarizes the core control flow.

We consider K episodes indexed by e, each with $T_e \leq H$ primitive steps, and aim to maximize average expected undiscounted return. Within an episode, action a_t yields reward r_t; we omit e when clear. The pre-action context x_t contains interaction history, remaining budget $h = H - t$, and controller state. For memory matching, $b_t = \phi(x_t)$ is extracted from observable information; it is not assumed to be sufficient for the hidden environment state.

3.1 Executable Reward-Segment Memory

Segment construction. Let $\mathcal{M}_e$ be the persistent option memory available at the start of episode e. We define reward events as transitions with nonzero environment reward. Boundaries occur at the episode start and immediately after these events (Figure 2(a)):

$$\mathcal{B}_e = \{0\} \cup \{t+1 : 0 \leq t < T_e,\ r_{e,t} \neq 0\}.$$

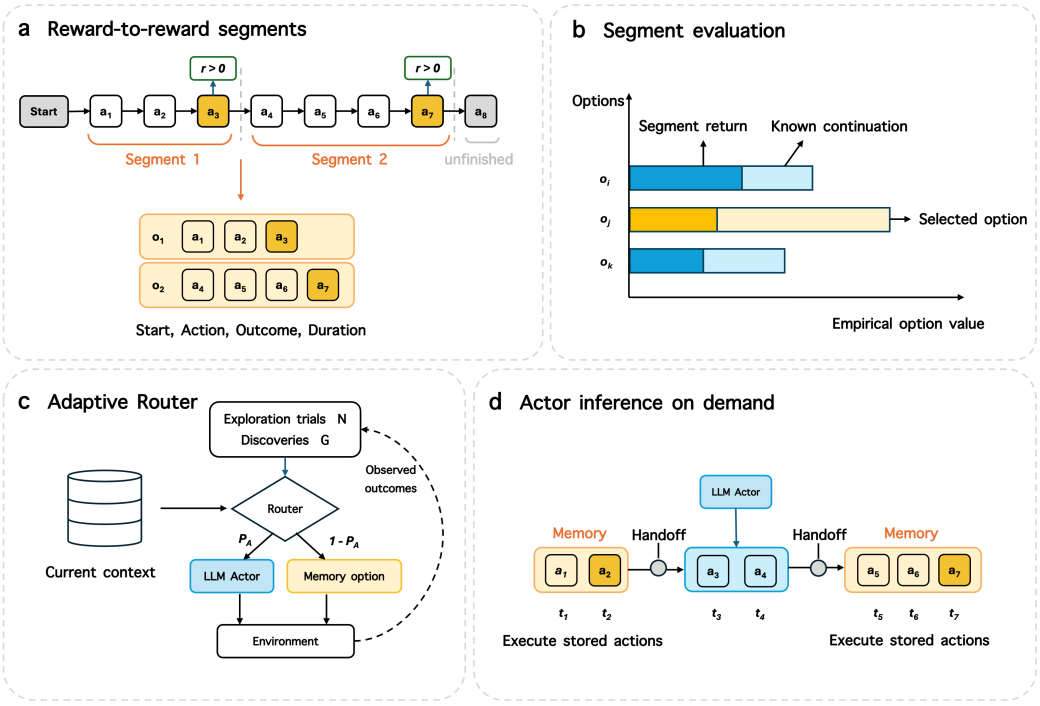


Figure 2: Overview of TEMPA. **(a)** Interaction is partitioned at reward events into executable segments, retaining their initiation context, actions, outcome, and duration. **(b)** Segment values combine observed returns with empirical continuation values; advantages relative to a common baseline guide selection among eligible options. **(c)** At free decision points, the router uses evidence from qualified exploratory trials to allocate control between the LLM actor and memory. **(d)** Selected memory segments execute persistently, while the frozen LLM generates actions on demand when assigned control. The trajectories and value bars are schematic.

For consecutive boundaries $u < v$ in $\mathcal{B}_e$, the completed segment records $\mathbf{a}_o = (a_u, \ldots, a_{v-1})$, its initiation descriptor $b_o^- = b_u$, its endpoint $b_o^+ = b_v$, and the intermediate descriptors. Its recorded length and net return are

$$\ell_o = v - u, \qquad R_o = \sum_{t=u}^{v-1} r_t.$$

Each segment contains zero rewards internally and ends with a reward event. Its return includes the final reward-producing action but excludes the preceding segment's reward. A nonempty suffix after the last boundary remains incomplete and provides exploration evidence rather than an executable option. Memory admission is specified in Appendix B.1.

The record induces an option $o = (I_o, \kappa_o, \zeta_o)$ (Sutton et al., 1999): initiation matching defines the eligible set I_o, the recorded program and cursor define the internal policy κ_o, and Section 3.3 specifies the termination rule ζ_o. The actor is a one-step option. Intermediate descriptors support execution alignment. Actual duration τ_o may differ from recorded length ℓ_o because of alignment, early termination, or the deadline. At episode end, admitted completed segments, including those generated through mixed Actor–Memory execution, enter $\mathcal{M}_{e+1}$.

Eligibility. At inference time, TEMPA retrieves options whose initiation descriptors match the current descriptor and whose recorded lengths fit the remaining budget h. Endpoint representatives reduce redundant alternatives; the canonicalization and tie-breaking rules are specified in Appendices B.1 and B.2. When an exploration target is active, options with known paths to it receive priority. We denote the resulting selection pool by $\mathcal{C}_e(b, h)$. Inclusion in memory records an ob-

served completed segment, not a guarantee of applicability at every future invocation; execution checks are applied as described in Section 3.3.

3.2 Empirical Option Values and Advantages

Stored segments form an empirical option graph: descriptors are vertices, and each option is an edge with a recorded endpoint, length, and net return. The planner evaluates known continuations through these observed transitions without fitting a parametric critic.

Let $\mathcal{O}_e(b, h)$ contain the representative outgoing options with $\ell_o \leq h$. An option's undiscounted value combines its net return with the best known continuation (Figure 2(b)):

$$\begin{aligned} \widehat{Q}_e(b, o, h) &= R_o + W_e(b_o^+, h - \ell_o), \\ W_e(b, h) &= \max\left\{0, \max_{o \in \mathcal{O}_e(b,h)} \widehat{Q}_e(b, o, h)\right\}. \end{aligned} \tag{1}$$

We set $W_e(b, 0) = 0$ and $W_e(b, h) = 0$ when no feasible outgoing option exists. The zero alternative stops expansion of the remembered plan; it does not assign zero value to subsequent actor behavior. Every edge consumes at least one step, so the recursion is finite even for cyclic graphs.

The recursive planner uses $\mathcal{O}_e$, whereas the current selection pool $\mathcal{C}_e(b, h) \subseteq \mathcal{O}_e(b, h)$, defined in Section 3.1, may be further restricted by the active control objective. For a nonempty pool, we define a common reference baseline and advantage:

$$\overline{V}_e(b, h) = \frac{1}{|\mathcal{C}_e(b, h)|} \sum_{o \in \mathcal{C}_e(b,h)} \widehat{Q}_e(b, o, h), \qquad \widehat{A}_e(b, o, h) = \widehat{Q}_e(b, o, h) - \overline{V}_e(b, h). \tag{2}$$

The reference averages eligible first-option choices under the same empirical continuation planner. Memory selects $o^\star \in \arg\max_{o \in \mathcal{C}_e(b,h)} \widehat{A}_e(b, o, h)$, breaking value ties by total planned execution cost and the remaining deterministic criteria. These quantities describe relative value within the empirical model, not an actor value estimate. As memory expands, updated continuation values can change which option is selected at the same context and remaining budget.

3.3 Temporal Control

At each step, the controller first updates execution status and scheduled handoffs. It then continues an active option or honors an existing reservation for transport or actor continuation. The actor–memory router applies Equation (3) only when neither determines control.

Adaptive routing. At an initiation descriptor b, a trial spans an attempted reward segment, ending at the next reward event or episode end. A trial is qualified if it contains at least one LLM-generated action. We count a qualified trial as a *discovery* if it completes an admitted segment with a previously unobserved endpoint identity and an outcome not dominated by earlier segments from the same initiation context. Dominance compares score, duration, and endpoint capabilities; the precise endpoint identity and dominance criteria are specified in Appendix B.1.

We maintain N_b, the number of qualified trials, and G_b, the number of those trials counted as discoveries. Discovery is defined relative to the current memory and does not necessarily imply a higher task score.

When no active execution or transport plan already determines control, the LLM actor probability (Figure 2(c)) is

$$p_{\mathrm{A}}(b) = \begin{cases} 1, & \mathcal{C}_e(b, h) = \emptyset \quad \text{or} \quad N_b < 2, \\ \dfrac{1 + G_b}{2 + N_b}, & \text{otherwise.} \end{cases} \tag{3}$$

The second branch uses the posterior mean under a Bernoulli discovery model with a $\mathrm{Beta}(1,1)$ prior as a probability-matching rule. Trials without discoveries increase N_b but not G_b; pure Memory executions do not advance this exploration evidence clock. Updating N_b and G_b adapts the local routing policy; the Memory branch selects an option as described in Section 3.2.

Persistent execution and handoff. A selected option retains control until completion, a reward event, execution failure, a scheduled handoff, or episode termination. Without an active option, initiation matching resumes after each primitive step, allowing actor behavior to reconnect with Memory. Handoffs along remembered plans use exposure statistics and the remaining budget (Appendix B.3).

Actor inference on demand. TEMPA defers actor inference when a remembered program can supply the next action (Figure 2(d)). If execution is rejected, it obtains a real actor action without resampling ownership. The checks require no learned predictor or simulated transitions; alignment, recovery, and inference order are detailed in Appendix B.4.

3.4 Regret Analysis

We analyze the local router's exploration cost relative to a fixed incumbent Memory policy. Consider T repetitions of the same boundary decision problem with identical remaining budget. Assume that the statistical key remains unchanged and each exploratory trial updates its count before the next repetition; pure Memory executions leave the counts unchanged. After warm-up, let $N_0 \geq 2$ and $0 \leq G_0 \leq N_0$ be the initial counts. After n additional exploratory trials, Equation (3) becomes

$$p_n = \frac{a}{n+c}, \qquad a = 1 + G_0, \quad c = 2 + N_0. \tag{4}$$

Let Y_j be the return of repetition j. Conditional on prior history, selecting Memory yields expected return v_{M}, while selecting exploration yields expected return at least $v_{\mathrm{M}} - B$, for $B \geq 0$. Here T counts local repetitions, not primitive steps. Expected regret relative to the incumbent is

$$\mathcal{R}_{\mathrm{inc}}(T) = \sum_{j=1}^{T} \left(v_{\mathrm{M}} - \mathbb{E}[Y_j] \right). \tag{5}$$

Proposition 1 (Sublinear exploration cost) *Under the assumptions stated in Appendix C.1, the local router satisfies*

$$\boxed{\mathcal{R}_{\mathrm{inc}}(T) \leq B \left[\sqrt{\left(c - \tfrac{1}{2}\right)^2 + 2aT} - \left(c - \tfrac{1}{2}\right) \right] = O(B\sqrt{aT}).} \tag{6}$$

The bound follows by controlling the expected number of exploratory trials under Equation (4); the complete proof is provided in Appendix C. For fixed initial counts, the bound on average additional loss decreases as $O(T^{-1/2})$: exploration declines with actual trial evidence rather than a predetermined episode schedule. This is a guarantee for the local router in the stated regime, not for the complete controller or discovery of an optimal policy.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate two text-based environments. **Jericho** (Hausknecht et al., 2020) contains interactive fiction games requiring navigation, object manipulation, and multi-step puzzle solving under sparse rewards. We evaluate the eight games in Table 1 for 50 episodes of at most 60 steps each. Within each run, a different environment seed is used for every episode. **ScienceWorld** (Wang et al., 2022) tests scientific reasoning through experiments in a simulated environment. We evaluate the five settings in Table 2 for 10 episodes, with at most 100 environment steps per episode. Exact

Table 1: Performance on eight Jericho games.

Method	Zork1		Zork3		Library		Detective	
	Avg	Final-5	Avg	Final-5	Avg	Final-5	Avg	Final-5
Static	13.45	12.33	1.39	1.73	24.19	25.47	46.80	44.00
Memory	25.22	32.67	1.95	2.07	20.53	19.60	184.00	236.00
Reflexion	33.21	36.87	1.97	2.20	21.69	18.53	194.13	187.33
AWM	22.63	24.67	1.62	1.80	13.24	13.80	122.87	226.67
EvoTest	32.48	32.00	2.63	2.93	9.10	9.33	236.27	299.33
APEX	29.01	33.80	1.93	2.53	26.42	28.07	71.73	116.00
JitRL	29.27	33.33	1.69	2.07	25.49	28.80	235.33	276.67
TEMPA	**43.19**	**52.33**	**2.74**	**3.80**	**26.59**	**29.33**	**277.87**	**320.00**

Method	Temple		Pentari		Ludicorp		ZTUU	
	Avg	Final-5	Avg	Final-5	Avg	Final-5	Avg	Final-5
Static	6.18	6.00	6.73	7.33	5.63	5.87	1.47	2.80
Memory	7.94	8.47	9.97	8.00	6.41	5.93	7.23	6.80
Reflexion	7.03	7.00	17.60	16.00	12.03	15.47	5.22	1.73
AWM	4.97	5.00	8.40	8.00	7.45	7.00	2.43	1.67
EvoTest	4.97	5.00	11.97	12.67	10.47	14.53	7.95	11.67
APEX	7.87	8.00	18.87	22.33	8.42	10.40	2.64	5.33
JitRL	6.61	6.93	12.47	18.33	11.99	14.73	4.11	3.40
TEMPA	**8.27**	**11.47**	**22.70**	**28.67**	**15.05**	**17.47**	**12.52**	**19.13**

task names are listed in Appendix B.5. Each experiment is repeated with three random seeds. We compute Avg and Final-K within each run and report their means across seeds.

Metrics. **Avg** averages all episode scores; **Final-**K averages the last K episodes, with $K = 5$ for Jericho and $K = 3$ for ScienceWorld. Jericho uses terminal scores. We report results per game or setting to avoid mixing score scales. Bold indicates the best result in each column.

Baselines. We compare against seven training-free baselines. **Static** uses no cross-episode memory, while **Memory** adds previous episode transcripts to the context through a FIFO buffer. **Reflexion** (Shinn et al., 2023) generates verbal reflections to guide subsequent attempts. **AWM** (Wang et al., 2024) extracts reusable workflows. **EvoTest** (He et al., 2026) iteratively revises agent components using experience. **APEX** (Li et al., 2026b) selects semantic milestones using dependency constraints and bandit statistics. **JitRL** (Li et al., 2026a) adjusts action preferences through memory-derived advantages. All methods in the main experiments use Qwen3.6-27B, with matched interaction budgets and no preloaded experience. TEMPA's base prompt templates are provided in Appendix D.

4.2 Main Results

Jericho. Table 1 shows that TEMPA achieves the highest Avg and Final-5 on all eight games. Gains vary by environment: Final-5 exceeds the strongest baseline by 41.9% on Zork1 and 63.9% on ZTUU, whereas Library remains closely contested. On Zork3 and Temple, TEMPA's Avg is close to the strongest baseline, but its Final-5 lead is larger. Its Final-5 also exceeds its own Avg in every game, reflecting stronger performance in later episodes.

ScienceWorld. Table 2 shows that TEMPA achieves the highest Avg in four of the five settings and the best or tied-best Final-3 in all five. Final-3 exceeds the strongest baseline by 38.5% on chemistry mixing and 38.4% on plant growth. TEMPA reaches the score ceiling of 100 in four settings. The largest late-episode gap occurs on melting point, where the strongest baseline reaches only 6.44. Electrical conductivity remains more competitive: AWM and Memory achieve higher

Table 2: Performance on five ScienceWorld settings.

Method	Chemistry mixing		Inclined plane angle		Plant growth		Electrical conductivity		Melting point	
	Avg	Final-3	Avg	Final-3	Avg	Final-3	Avg	Final-3	Avg	Final-3
Static	-31.13	-18.56	21.33	15.56	26.80	39.22	13.70	35.33	-26.67	-33.33
Memory	-91.13	-85.22	14.17	25.56	20.60	19.00	70.10	**100.00**	-40.00	-33.33
Reflexion	44.17	11.11	28.70	76.67	25.40	35.00	-10.83	-5.78	-83.10	-77.78
AWM	13.33	55.56	43.00	83.00	45.87	41.33	**72.07**	74.67	-40.00	-33.33
EvoTest	-40.00	33.33	53.33	60.00	37.03	42.00	17.23	77.78	-14.57	-9.56
APEX	44.47	72.22	20.83	10.00	14.00	15.11	26.40	30.67	-4.93	6.44
JitRL	-26.67	-33.33	46.67	55.56	45.70	42.00	6.57	-11.11	-48.40	-80.67
TEMPA	**66.67**	**100.00**	**60.00**	**100.00**	**54.17**	**58.11**	63.07	**100.00**	**60.00**	**100.00**

Table 3: Ablations of TEMPA on three Jericho games.

Variant	Zork1		Pentari		ZTUU	
	Avg	Final-5	Avg	Final-5	Avg	Final-5
Fixed-Segment	18.67	19.33	18.67	16.00	7.71	8.73
Adaptive-Step	9.77	14.67	13.20	16.00	3.95	5.00
Fixed-Step	7.10	17.00	11.87	15.00	2.49	4.13
Myopic segment selection	30.39	36.13	15.20	20.00	7.95	10.33
Length-5	31.00	33.00	13.40	17.67	4.60	6.33
Length-10	30.44	43.27	12.23	16.33	8.19	13.27
Best-prefix replay	10.00	16.33	8.67	10.00	6.69	8.33
TEMPA	**43.19**	**52.33**	**22.70**	**28.67**	**12.52**	**19.13**

Avg, and Memory matches TEMPA's Final-3. Plant growth remains below the score ceiling despite TEMPA's lead.

4.3 Ablations

Adaptive routing and persistent execution. We cross adaptive or fixed routing with persistent or stepwise execution. Fixed-Segment and Fixed-Step use a 50%–50% actor–memory split; Adaptive-Step and TEMPA use adaptive routing. The Step variants repeat actor–memory arbitration before every action within a segment. Fixed-Segment and TEMPA retain control until completion or interruption. Full TEMPA results are taken from Table 1.

Table 3 shows that persistent execution improves both metrics under fixed routing. Adaptive routing improves Avg under stepwise execution, but not consistently Final-5. On Zork1, Fixed-Segment and Adaptive-Step have 63.1% and 72.0% lower Final-5 than TEMPA, respectively. These results highlight the complementary roles of adaptive routing and persistent execution.

Myopic segment selection. The myopic variant removes W_e from Equation (1) and computes advantages using only recorded segment returns R_o. This reduces Avg by 29.6–36.5% and Final-5 by 30.2–46.0% relative to TEMPA, with the largest decline on ZTUU. The gap supports evaluating remembered continuations rather than immediate segment returns alone.

Reward-boundary segmentation. Length-5 and Length-10 cut eligible spans every five or ten actions counted from the episode start, independently of intermediate rewards. Short residual blocks at span endpoints are retained without padding. The remaining suffix after the last accepted reward endpoint contributes to exploration exposure but is excluded from executable memory.

Both variants trail TEMPA in Avg and Final-5. TEMPA's Final-5 gains over the stronger fixed-length variant range from 20.9% to 62.3%. Length-10 outperforms Length-5 in Final-5 on Zork1 and ZTUU but not Pentari, favoring reward boundaries over uniformly longer segments.

Best-prefix replay. This variant replays the best successful prefix collected so far, then lets the LLM continue exploring. It reduces Avg by 46.6–76.8% and Final-5 by 56.5–68.8% relative to TEMPA (Table 3). Replaying the best known prefix alone does not reproduce TEMPA's performance.

Additional results with Gemma-4-31B-it are reported in Appendix E.1.

4.4 Adaptation Dynamics

Figure 3 shows that TEMPA achieves higher late-episode scores than JitRL and Reflexion on all three games. The advantage develops differently across environments. On Zork1 and Pentari, TEMPA establishes a lead relatively early. On ZTUU, it initially trails Reflexion before separating from both baselines later in the run. These trajectories complement the aggregate results by showing how TEMPA's performance advantage develops through repeated interaction.

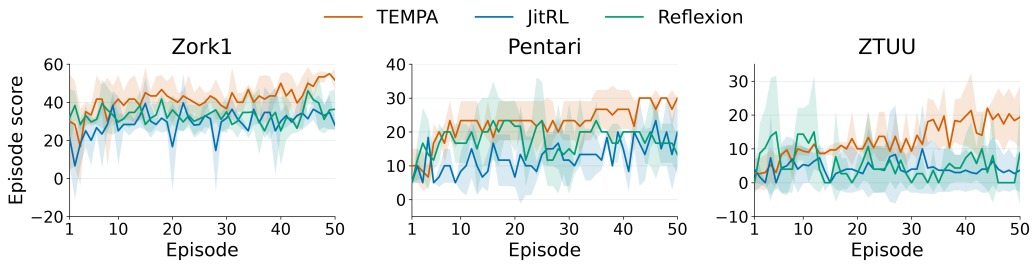


Figure 3: Mean episode scores across three seeds; shaded bands indicate standard deviation.

4.5 Efficiency

Table 4 shows that TEMPA has the lowest aggregate input and output token consumption across the two benchmarks. Consumption is 56.1% below JitRL and 80.8% below Memory, while remaining 14.0% below Static.

At memory-controlled steps, stored programs supply executable actions without requiring the LLM to generate them again. Segment selection and temporal control determine which behavior to reuse and when to return control to the actor. On-demand inference implements this allocation without generating actor outputs that memory would override (Appendix B.4). The resulting savings accompany the performance gains in Section 4.2. Appendix E.2 reports the execution shares.

Table 4: Comparison of total token consumption.

Method	Static	Memory	Reflexion	AWM	EvoTest	APEX	JitRL	TEMPA
Tokens	164.49M	735.96M	235.81M	207.18M	359.49M	240.51M	321.99M	**141.51M**

5 Conclusion

We introduced TEMPA, a gradient-free framework that adapts LLM agents through executable reward-to-reward segments. Segment values guide memory selection, while exploration evidence governs control allocation between memory and the frozen actor. Experiments on Jericho and ScienceWorld show improved performance with lower aggregate token consumption. Future work can explore how executable memory can better guide the discovery of new behaviors.

AI USE STATEMENT

We used large language models to assist with language editing, LaTeX formatting, manuscript organization, and literature search, including the identification of potentially relevant related work. Language editing included sentence rephrasing, grammar correction, and improvements to readability and textual flow.

LLMs were not used to develop research ideas, formulate or prove theoretical claims, implement methods, design experiments, or analyze or interpret results. The authors independently verified the cited sources and assessed their relevance, and reviewed all AI-assisted revisions to ensure that they preserved the intended scientific meaning. The authors take full responsibility for the final manuscript.

ETHICS STATEMENT

We evaluate TEMPA in the simulated environments of Jericho and ScienceWorld. Our experiments focus on task performance and inference efficiency rather than real-world deployment. Because TEMPA stores and executes past experience, its memory may preserve errors or inappropriate behaviors. In real-world applications, stored experience may also contain sensitive information. Such deployments should therefore incorporate privacy protection, restricted action permissions, and human oversight for consequential actions.

REPRODUCIBILITY STATEMENT

To support independent reimplementation, we provide the algorithm in Appendix A, implementation details in Appendix B, and base prompt templates in Appendix D. Section 4.1 specifies the benchmarks, backbone model, interaction budgets, evaluation metrics, and aggregation across random seeds. Appendix C provides the proof of our theoretical result and clarifies its assumptions. Additional experimental results are reported in Appendix E.

REFERENCES

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture, 2016. URL https://arxiv.org/abs/1609.05140.

Arthur Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Heinecke, Silvio Savarese, Victor Zhong, and Caiming Xiong. Test-time adaptation for llm agents via environment interaction, 2026. URL https://arxiv.org/abs/2511.04847.

Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pp. 720–727, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933034. doi: 10.1145/1160633.1160762. URL https://doi.org/10.1145/1160633.1160762.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7903–7910. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6297. URL https://doi.org/10.1609/aaai.v34i05.6297.

Yufei He, Juncheng Liu, Yue Liu, Yibo Li, Tri Cao, Zhiyuan Hu, Xinxing Xu, and Bryan Hooi. Evotest: Evolutionary test-time learning for self-improving agentic systems, 2026. URL https://arxiv.org/abs/2510.13220.

Yibo Li, Zijie Lin, Ailin Deng, Xuan Zhang, Yufei He, Shuo Ji, Tri Cao, and Bryan Hooi. Just-in-time reinforcement learning: Continual learning in llm agents without gradient updates, 2026a. URL https://arxiv.org/abs/2601.18510.

Yibo Li, Jiashuo Yang, Zhi Zheng, Zhiyuan Hu, Yuan Sui, Shizun Wang, Yufei He, and Bryan Hooi. Apex: Autonomous policy exploration for self-evolving llm agents, 2026b. URL https://arxiv.org/abs/2605.21240.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 23813–23825. Curran Associates, Inc., 2023. doi: 10.52202/075280-1034. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4b0eea69deea512c9e2c469187643dc2-Paper-Conference.pdf.

Yitao Liu, Chenglei Si, Karthik R Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. 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. 14179–14198, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.694. URL https://aclanthology.org/2025.acl-long.694/.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents, 2026. URL https://arxiv.org/abs/2602.01869.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Xinyue Yang, Jiadai Sun, Yu Yang, Shuntian Yao, Tianjie Zhang, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

Zerui Qin, Sheng Yue, Xingyuan Hua, Yongjian Fu, and Ju Ren. Executable agentic memory for gui agent, 2026. URL https://arxiv.org/abs/2605.12294.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

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, 2023. URL https://arxiv.org/abs/2305.16291.

Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.775. URL https://doi.org/10.18653/v1/2022.emnlp-main.775.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents, 2026. URL https://arxiv.org/abs/2605.06078.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024. URL https://arxiv.org/abs/2409.07429.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Qizheng Zhang, Michael Wornow, and Kunle Olukotun. Agentic plan caching: Test-time memory for fast and cost-efficient llm agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*,

volume 38, Main Conference, pp. 103270–103296. Curran Associates, Inc., 2025. doi: 10.52202/085713-3451. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/9549f7d06700f0966d5f938f1d11022a-Paper-Conference.pdf.

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.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A ALGORITHM OVERVIEW

Algorithm 1 presents the main interaction and adaptation loop. It separates empirical option selection, allocation of control, and memory construction from executed experience. Blue step comments identify the main stages; indentation indicates which stages repeat within an episode.

Algorithm 1 TEMPA: policy adaptation with executable segment memory

Input: Frozen Actor π_θ, descriptor map ϕ, episodes K, step budget H.
Initialize: $\mathcal{M}_1 \leftarrow \emptyset$, $\mathcal{D}_1 \leftarrow 0$.

for episode $e = 1, \ldots, K$ **do**
  // Step 1: Initialize episode
  Reset; observe x_0; $t \leftarrow 0,\ z \leftarrow \emptyset$.
  $\tau_e, U, \mathcal{P}_e \leftarrow \emptyset$.
  $c \leftarrow \mathrm{InitController}(\mathcal{M}_e, \mathcal{D}_e, x_0, H)$.
  while $t < H$ and not terminal **do**
    // Step 2: Check execution
    $b \leftarrow \phi(x_t), \quad h \leftarrow H - t$.
    Update (z, c): alignment, termination, reservations.
    if z retains control **then** $u \leftarrow \mathrm{Memory}$.
    else
      // Step 3: Evaluate segments
      Retrieve $\mathcal{C}_e(b, h)$; if nonempty, evaluate Eqs. (1)–(2).
      // Step 4: Route control
      if c has a reservation **then** apply it to (u, z).
      else
        Set p_{A} by Eq. (3); $q \sim \mathrm{Uniform}(0, 1)$.
        $u \leftarrow \mathrm{Actor}$ if $q < p_{\mathrm{A}}$, else Memory.
      end if
      if $u = \mathrm{Memory}$ and $z = \emptyset$ **then**
        If $\mathcal{C}_e(b, h) = \emptyset$, $u \leftarrow \mathrm{Actor}$; otherwise choose
        $o^\star \in \arg\max_{o \in \mathcal{C}_e(b,h)} \widehat{A}_e(b, o, h)$; $z \leftarrow o^\star$.
      end if
    end if
    // Step 5: Actor inference on demand
    $a_t \leftarrow \mathrm{CheckedAction}(z, c, x_t)$ if $u = \mathrm{Memory}$, else $\perp$.
    if $a_t = \perp$ **then**
      Release rejected program; $a_t \leftarrow \mathrm{ActorAction}(\pi_\theta, x_t)$.
    end if
    // Step 6: Execute and segment
    Execute a_t; observe r_t, x_{t+1}; log transition and action source in τ_e, U.
    if $r_t \neq 0$ **then**
      $\mathcal{P}_e \leftarrow \mathcal{P}_e \cup \mathrm{StageSegment}(U), \quad U \leftarrow \emptyset$.
    end if
    Update (z, c); $t \leftarrow t + 1$.
  end while
  // Step 7: Update experience
  $\mathcal{D}_{e+1} \leftarrow \mathrm{UpdateEvidence}(\mathcal{D}_e, \tau_e, \mathcal{P}_e, U)$.
  $\mathcal{M}_{e+1} \leftarrow \mathcal{M}_e \cup \mathcal{P}_e$.
end for

Controller state and subroutines. Here $\mathcal{D}_e$ contains exploration statistics, τ_e the episode trace, U the segment buffer, and $\mathcal{P}_e$ pending completed segments. InitController initializes reservations in c (Appendix B.3); z holds an active program and cursor. Persistent execution and existing reservations precede the local router. Retrieval priorities, fallback tiers, and tie-breaking follow Appendix B.2.

CheckedAction returns an executable program command or $\perp$. ActorAction generates or reuses real candidates following Appendix B.4, without resampling ownership after a rejected Memory

command. The conservative preliminary check may already have called the actor, so not every Memory action avoids inference. $\mathrm{StageSegment}(U)$ constructs the record in Section 3.1 and returns the set admitted by Appendix B.1, or the empty set. Failure transitions remain in τ_e for evidence updates even when they are not staged for memory. $\mathrm{UpdateEvidence}$ distinguishes completed Actor-containing trials, pure Memory records, failure spans, and unfinished suffixes (Appendices B.1 and B.3).

B Implementation Details

This appendix describes empirical memory, planning, and the execution interface. Appendices B.1–B.4 describe the Jericho reference implementation; Appendix B.5 summarizes the ScienceWorld adaptation. The option values, local discovery counts, and exposure statistics have different meanings and are maintained separately; none is an estimate of the Actor's action logits or of memory reliability.

B.1 Observable Descriptors and Memory Construction

Initiation matching. The Jericho descriptor is

$$b = (\text{location}, \text{inventory}, \text{score}, \text{room description}).$$

Text fields are lowercased and whitespace is normalized; inventory entries are normalized and sorted. Matching uses equality of the resulting descriptors. Commands are compared using the implementation's action normalization. The descriptor uses information exposed by the environment adapter, not the hidden simulator state. In particular, equality does not certify equality of unobserved object states or future transitions. The current action list is checked separately and is not part of the memory key.

Stored records. In addition to the quantities in Section 3.1, each segment retains the descriptors before and after each action, its source episode and step indices, and the number of actions supplied by Actor and Memory. The writer follows the executed trajectory, including changes of controller ownership. Consequently, a completed segment can contain both Actor and Memory actions. New records are staged during an episode and incorporated into the persistent archive at its end. An unfinished suffix provides exploration evidence, but is not an edge with an observed reward endpoint in the option planner.

In the evaluated configuration, only segments ending with a positive reward are admitted to executable memory; negative-reward spans remain exploration evidence.

Let $\rho_o = r_{v-1} \neq 0$ denote a segment's endpoint reward. For reward-to-reward segments, $R_o = \rho_o$, since internal rewards are zero. For score-delta rewards, $R_o = \mathrm{score}(b_o^+) - \mathrm{score}(b_o^-)$. The implementation uses endpoint scores for planning and retains ρ_o for endpoint identity and execution diagnostics.

Endpoint representatives. Within one initiation descriptor, records are grouped by

$$\big(b_o^+,\ \rho_o,\ \mathrm{normalize}(a_{v-1})\big).$$

Each group contributes one representative to planning. Selection first prefers records without a forward cursor synchronization during their collection, then smaller length, earlier source episode, earlier completion step, and smaller record identifier, in that order. Thus length minimization is subject to a provenance preference rather than unconditional. Other records remain available for support counts; representative selection does not erase repeated observations.

Local discovery criterion. For records with the same initiation descriptor, a previous segment o' dominates a candidate o for discovery counting if all of the following hold:

$$\begin{aligned} \mathrm{location}(b_{o'}^+) &= \mathrm{location}(b_o^+), \\ \mathrm{score}(b_{o'}^+) &\geq \mathrm{score}(b_o^+), \\ \mathrm{inventory}(b_{o'}^+) &\supseteq \mathrm{inventory}(b_o^+), \qquad \ell_{o'} \leq \ell_o. \end{aligned}$$

Table 5: Statistics used by the temporal controller. Exposure counts primitive environment steps, not LLM calls.

Decision and key	Event	Trial or exposure
Local router, b	New nondominated endpoint identity	One attempted segment containing an Actor action, including an unfinished suffix.
Boundary targeting, b	Improvement of the local empirical option plan	Length of each Actor-containing segment or unfinished suffix.
Internal handoff, (b, a)	First score above the designated incumbent	Steps after an executed Actor release, up to that event or episode end.

Equality is sufficient to suppress duplicate discovery credit. Room text participates in exact endpoint identity but not in this dominance predicate. A discovery counted by G_b must have both a previously unseen endpoint identity and no dominating prior record. A shorter record with an already observed endpoint identity does not increment G_b in this configuration.

Each admitted segment containing at least one Actor action increments N_b; it increments G_b only if the discovery criterion holds. An Actor-containing unfinished suffix or negative-reward span increments N_b, but not G_b. Pure Memory executions can add records without increasing either count. Updates are made at episode end, with earlier pending segments included when testing later segments for endpoint novelty and dominance. Several decisions within an episode can therefore use the same previous counts, a distinction relevant to the assumptions of Appendix C.

B.2 Empirical Planning and Option Selection

Finite-horizon computation. Equation (1) is evaluated by recursion with memoization on (b, h). The score-delta implementation stores the highest reachable final score and, among equal-score plans, the smallest total recorded length. Subtracting the current score gives the corresponding additional return. The option graph contains only observed completed segments; an empty continuation means that no further remembered edge is selected, not that the Actor cannot obtain additional reward. The planner does not simulate the environment or estimate the probability that a composed plan will replay successfully.

Selection pool and priorities. Only representative options whose first segment fits the current budget are eligible. If a transport target is active and some eligible first options have a graph path from their recorded endpoints to that target, those options form the preferred tier. Otherwise, all eligible first options form the tier. This tier is the pool $\mathcal{C}_e(b, h)$ used for the reference baseline in Equation (2); downstream value recursion still uses $\mathcal{O}_e$. The target test is graph reachability, not an additional reward or a certificate that the entire transport path fits the residual horizon. Recorded lengths constrain the value recursion and the separate transport-cost calculation.

After selection by empirical advantage within this tier, ties are resolved by smaller total planned length, larger final reward label ρ_o, larger endpoint score, earlier source episode, earlier completion step, and finally larger record identifier. No Actor probability enters these option values. The resulting plan is an empirical preference over remembered behavior, not a calibrated estimate of the true option value under a reference policy in the partially observed environment.

B.3 Discovery Statistics and Scheduled Handoffs

The local router uses Equation (3) when no active program or existing control reservation determines ownership. The full controller additionally makes an episode-initialized decision about where to hand control to the Actor. Table 5 distinguishes its three statistical ledgers. Sharing a word such as "discovery" does not make their events interchangeable.

An exposure-based scheduling score. Let D_j count events and E_j accumulated exposure for a scheduling key j. A Gamma shape–rate working model gives

$$\lambda_j \mid D_j, E_j \sim \mathrm{Gamma}(\alpha_0 + D_j, \beta_0 + E_j), \qquad p_j(L) = 1 - \left(\frac{\beta_0 + E_j}{\beta_0 + E_j + L}\right)^{\alpha_0 + D_j}. \quad \text{(B.1)}$$

Here L is the number of steps left after transport; $p_j(L) = 0$ for $L \leq 0$. The formula integrates $1 - \exp(-\lambda_j L)$ under the Gamma distribution. The Jericho configuration uses $\alpha_0 = 1$ and $\beta_0 = 60$, equal to one episode's primitive-step budget. Events and exposure are never added to option returns or Actor logits.

Handoff inside an established plan. At episode initialization, the controller constructs an incumbent plan using empirical final score and planned length. Its last option is eligible for internal handoff scheduling when the same start and endpoint identity have been observed in at least two distinct source episodes and a complete trace is available. For every pre-action cursor $k \in \{0, \ldots, \ell_o - 1\}$ of this last option, let b_k be the recorded descriptor and a_k its known next action. If preceding options cost c_{pre} steps, the residual budget is

$$L_k = H - c_{\mathrm{pre}} - k.$$

The controller uses the ledger keyed by $(b_k, \mathrm{normalize}(a_k))$, chooses the cursor with largest $p_j(L_k)$, and breaks ties in favor of the later cursor. The post-option endpoint is not another candidate cursor in this procedure.

With the selected probability, the controller schedules transport and an Actor handoff at that cursor; otherwise it gives priority to Memory exploitation. Transport executes real actions. A nonempty internal prefix is replayed with a strict descriptor contract, whereas a cursor at the option start requires no internal prefix. A planned handoff is not counted as an exploratory trial unless it is reached and an Actor action is actually executed. After a zero-reward Actor action at the selected target, an Actor continuation reservation remains active until a nonzero reward, termination, or the deadline. This reservation takes precedence over the router.

For an executed internal handoff, the incumbent threshold is the recorded endpoint score of the last option in the chosen plan. The event is the first subsequent observed score strictly above that threshold. Exposure includes all primitive steps from the release through that event, or through episode end if no event occurs, including any later Memory execution. These statistics are not success counts for reproducing the incumbent itself.

Fallback to reachable reward boundaries. If no established terminal option qualifies, the controller considers other descriptors reachable through completed options. Shortest recorded transport costs must leave at least one step for exploration. It ranks candidate boundaries by Equation (B.1), then by higher score, lower transport cost, and deterministic descriptor order. It samples between transporting to the selected boundary for exploration and prioritizing Memory exploitation. Without a transported candidate, control falls back to the ordinary local procedure.

This boundary ledger uses a different event: a new Actor-containing segment must improve the best local plan lexicographically in final score and negative total length. The comparison uses the configured planning horizon and the existing archive, with the local best updated across pending segments. Segment length contributes exposure whether or not the plan improves; an unfinished tail contributes exposure without an event. Thus a shorter equivalent plan can improve this ledger without qualifying as a new endpoint for G_b.

Interpretation and limits. Equation (B.1) is a scheduling model, not a theorem of optimal stopping. A literal time-to-event posterior requires a suitable hazard model and noninformative censoring. Changing policies, changing incumbent thresholds, terminal failures, and non-Markov descriptors can violate those assumptions. Moreover, boundary statistics aggregate segment outcomes rather than first-event times, whereas the internal-handoff ledger stops at the first threshold crossing. We therefore use the scores operationally without claiming calibrated discovery probabilities or guaranteed transport.

Local routing only. We compare the full controller with a variant that retains empirical option selection and persistent segment execution but uses only the local router in Equation (3) at free decision points. It removes the episode-initialized boundary targeting and internal handoff scheduling described above. Full TEMPA results are taken from Table 1.
Table 6 shows that full TEMPA improves Avg by 10.7–18.0% and Final-5 by 17.1–20.5% over local routing alone. The larger Final-5 gains indicate that the benefit is more pronounced in later episodes, with the largest gain on ZTUU. These results support complementing local routing with scheduled control.

Table 6: Local routing alone versus the full TEMPA controller.

	Zork1		Pentari		ZTUU	
Variant	Avg	Final-5	Avg	Final-5	Avg	Final-5
Local routing only	39.01	44.67	19.23	24.00	11.24	15.87
TEMPA	**43.19**	**52.33**	**22.70**	**28.67**	**12.52**	**19.13**

B.4 Execution, Recovery, and Actor Inference on Demand

Alignment is not a universal termination test. For an ordinary active reward segment, the controller first compares the current descriptor with the recorded descriptor at its cursor. If these differ but a later pre-action descriptor in the same segment matches, the cursor moves to the nearest such position. If no later match exists, the reference implementation retains the cursor and checks whether the next command is available; descriptor inequality alone does not always terminate ordinary replay. No other option is spliced into the active segment by this alignment operation. Transport to an internal handoff instead enforces strict descriptor checks and is rejected on mismatch.

Command checks and termination. A remembered command can be matched to an already generated Actor candidate or inserted from the environment's current action list. The adapter also permits the historical observation command `look`, which the action enumerator can omit. In the candidate-matching branch, membership in the environment action list is recorded but not an additional rejection condition. These are execution eligibility checks, not semantic safety guarantees. If the command cannot be supplied, control returns to a real Actor action. An ordinary active segment is cleared at a reward event, exhaustion of its program, environment termination, or the interaction deadline; scheduled handoffs can release it earlier. An unexpected positive endpoint is recorded as such.

Order of operations. The main execution loop proceeds as follows:

1. Update initiation matches and execution alignment, honor existing transport or Actor continuation reservations, and process scheduled handoffs. These precede a new router decision.
2. Check whether an active or selectable remembered program can supply the action. If an Actor action is already necessary, generate candidates; otherwise defer generation.
3. Finalize ownership once. Continue an eligible program, select an option under the active plan, or invoke the router when neither determines control. If a deferred call becomes necessary because the final decision cannot execute Memory, obtain real Actor candidates and complete that decision without drawing the ownership choice again.
4. Execute exactly one actual command, record its observation, reward, descriptor, and source of control, and update execution state. When no program or reservation remains active, the next step can again match an option initiation. At episode end, finalize the statistical ledgers and merge completed segments into the archive.

Deferral is a controller check, not a trained predictor, and no placeholder action is sent to the environment. Every executed command counts toward the primitive-step budget, including transport and recovery; there is no free state restoration. The preliminary check is conservative, so not every step ultimately owned by Memory necessarily saves an Actor call. Token accounting must use all actual requests, including fallback generation, history planning, and candidate-validation retries,

rather than infer cost from the fraction of Memory actions. Avoiding unused calls does not itself guarantee identical trajectories to eager inference, because the sequence of model requests can affect subsequent outputs.

B.5 ScienceWorld Adaptation

The settings in Table 2 correspond to the following ScienceWorld task names:

- Chemistry mixing: `chemistry-mix`.
- Inclined plane angle: `inclined-plane-determine-angle`.
- Plant growth: `grow-plant`.
- Electrical conductivity: `test-conductivity-of-unknown-substances`.
- Melting point: `measure-melting-point-unknown-substance`.

State representation and memory. The actor receives the task description, textual observations, inventory, and the environment's valid-action list. Memory matching uses normalized location, inventory, score, and room description, as in Appendix B.1. In our experiments, the environment seed determines the task variation. Resetting the environment between episodes preserves this variation within each run. Memory persists across episodes within a task-variation run but is not shared across runs. Gold action sequences and internal goal-progress annotations are not used.

Rewards and execution. Reward events delimit segments, and observed score changes determine their net returns without an LLM reward evaluator. We report terminal environment scores, retaining negative scores as failures rather than replacing them with the best intermediate score. Negative scores terminate the episode. Memory execution retains observable-state checks and on-demand actor fallback; in-episode `reset task` commands are excluded.

Control adaptation. Empirical option planning and the local router are retained. Exploration scheduling at reachable reward boundaries additionally accounts for terminal failures alongside discovery and exposure statistics.

C Regret Analysis

C.1 Proof of Proposition 1

We retain the assumptions in Section 3.4. Let $\mathcal{F}_{j-1}$ denote the history available before repetition j, and let I_j indicate whether that repetition selects exploration. Define $Z_0 = 0$ and $Z_j = \sum_{i=1}^{j} I_i$. With no further discovery events and each trial's count incorporated before the next repetition,

$$\Pr(I_j = 1 \mid \mathcal{F}_{j-1}) = \frac{a}{Z_{j-1} + c}, \qquad a = 1 + G_0, \quad c = 2 + N_0.$$

Since $N_0 \geq 2$ and $0 \leq G_0 \leq N_0$, we have $a > 0$, $c \geq 4$, and $a < c$; thus the displayed quantity is a valid probability. No independence of returns across repetitions is required beyond the stated conditional expectation assumptions.

Bounding the number of exploratory trials. Consider the potential

$$F(n) = n^2 + (2c - 1)n.$$

Its increment is $F(n+1) - F(n) = 2(n + c)$. Since the count increases only when exploration is selected,

$$\begin{aligned}\mathbb{E}[F(Z_j) - F(Z_{j-1}) \mid \mathcal{F}_{j-1}] &= \frac{a}{Z_{j-1} + c}\, 2(Z_{j-1} + c) \\ &= 2a.\end{aligned}$$

Summing over $j = 1, \ldots, T$ and using $F(Z_0) = 0$ yields

$$\mathbb{E}[Z_T^2] + (2c - 1)\mathbb{E}[Z_T] = 2aT.$$

Let $m_T = \mathbb{E}[Z_T]$. Jensen's inequality gives $m_T^2 + (2c-1)m_T \le 2aT$. Solving this quadratic inequality for $m_T \ge 0$ gives

$$\mathbb{E}[Z_T] \le \sqrt{\left(c - \tfrac{1}{2}\right)^2 + 2aT} - \left(c - \tfrac{1}{2}\right).$$

From exploratory trials to expected loss. By assumption, Memory has conditional expected return v_{M} whenever it is selected, and an exploratory trial has conditional expected return at least $v_{\mathrm{M}} - B$. Writing $p_j = a/(Z_{j-1} + c)$, the conditional mean return therefore satisfies

$$\mathbb{E}[Y_j \mid \mathcal{F}_{j-1}] \ge (1 - p_j)v_{\mathrm{M}} + p_j(v_{\mathrm{M}} - B) = v_{\mathrm{M}} - Bp_j.$$

Taking expectations and summing,

$$\begin{aligned}\mathcal{R}_{\mathrm{inc}}(T) &\le B\sum_{j=1}^{T}\mathbb{E}[p_j] = B\,\mathbb{E}[Z_T] \\ &\le B\left[\sqrt{\left(c - \tfrac{1}{2}\right)^2 + 2aT} - \left(c - \tfrac{1}{2}\right)\right].\end{aligned}$$

The bracket is at most $\sqrt{2aT}$, establishing the stated $O(B\sqrt{aT})$ upper bound.

Scope. The result applies to the repeated local decision problem specified in Section 3.4. In particular, the no-discovery condition defines the analyzed regime; it is not obtained by retrospectively conditioning a general adaptive run on the absence of discoveries, which could change the law of its routing decisions. Counts must be updated before the next repetition, so the proof does not automatically cover multiple decisions made using stale statistics within an episode. The result also does not cover changing initiation contexts, newly discovered options, or the episode-initialized handoff controller. Finally, it compares against a fixed Memory policy with the specified expected return, not an oracle that selects the best action or knows when memory will fail.

D Prompt Templates

We provide TEMPA's base actor, history-summary, and recovery templates. The actor follows the JitRL-style candidate interface (Li et al., 2026a). Braced fields are populated at runtime; line wrapping is adjusted for readability. The displayed action template uses three candidates, reduced when fewer legal actions are available. TEMPA obtains rewards and segment values from environment feedback rather than an LLM evaluator. These templates describe TEMPA, not a claim that every baseline uses identical prompts.

D.1 Jericho Action Generation

The actor receives the current state, a planning summary, and up to 20 recent raw transitions from the current episode. At the first step, the history field is `No game history.` The reference-action list contains the exact command strings presented to the actor. Candidate preferences are read from index-token log probabilities, so the JSON requests no verbalized confidence fields. A JSON schema also enforces the legal-action constraint.

Actor: system prompt

```
You are an expert player aiming to complete a text-based
adventure game. Points are given for making progress in the
game. Select promising actions based on the game state and
memory of past interactions.

**EXPLORATION PRIORITY**: When you arrive at a NEW location
you haven't fully explored before, you MUST thoroughly
explore it FIRST before leaving.

Follow this guide: Explore the environment and try to
maximize your score.
```

```
**CRITICAL CONSTRAINT**: When REFERENCE ACTIONS are provided,
you MUST ONLY choose actions from that list. Any action not
in the REFERENCE ACTIONS list is INVALID and will fail. Do
NOT create custom actions. The list is unordered - position
doesn't indicate quality.
```

Actor: user prompt

```
GAME HISTORY:
{history_summary}

CURRENT STATE: {state_text}

TASK:
1. Analyze your progress: What have you achieved? What's your
   next objective?
2. MANDATORY: Check the REFERENCE ACTIONS list below and only
   select from it
3. Propose 3 different actions with reasoning

RESPONSE FORMAT (JSON):
{
    "progress_analysis":
        "What you've achieved and current challenges",
    "next_objective": "Your next goal",
    "reasoning1": "Why this action makes sense",
    "option1": "action command",
    "reasoning2": "Why this action makes sense",
    "option2": "action command",
    "reasoning3": "Why this action makes sense",
    "option3": "action command",
    "best_action": 1
}

KEY RULES:
- ALL actions MUST be selected EXACTLY from the REFERENCE
  ACTIONS list
- Do not create, modify, or combine actions
- Pay attention to game hints and clues in state descriptions
- Do not repeat failed actions or create loops

REFERENCE ACTIONS (ONLY VALID ACTIONS):
{valid_actions}
```

D.2 History Summary

When episode history is available, a separate call summarizes the observed trajectory for the next actor decision. Each historical step is serialized with its state, executed action, reward, and resulting score; the current state is appended as the final entry. The resulting summary is combined with recent raw transitions to populate `history_summary` above. This is planning context, not a learned state descriptor or a source of reward labels. The same summary template is retained for ScienceWorld.

History summary: system prompt

```
You are an expert at analyzing text-adventure trajectories and
creating concise, distinctive planning summaries.

Return exactly these sections:
[SUMMARY] A 2-4 sentence summary of the inferred objective,
concrete progress, current situation, and important failed
```

```
or repeated actions.
[PROGRESS] Only milestones supported by a score increase,
formatted as 'M#: action -> result'; write 'No Progress' if
none.
[LOCATION] A concise meaningful location path with
unproductive backtracking removed.
[NEXT_OBJECTIVE] One concrete, non-walkthrough objective
justified by the observed trajectory.
[AVOID] Actions that were ineffective, redundant, or already
completed.

Use specific objects and state changes. Do not invent
accomplishments, inventory, rewards, or inaccessible
walkthrough knowledge.
```

History summary: user prompt

```
Analyze the trajectory below for the next action decision.

{trajectory}
```

D.3 ScienceWorld Actor Adaptation

The `memory_scientist` profile replaces the adventure-game role with the following system prompt while retaining the candidate JSON format. The user template in Appendix D.1 is reused, except for the two rules about game hints and repeated failed actions. The state field additionally includes the public task description and task-instance identifier. History summarization is not rewritten by this role adapter.

ScienceWorld actor: system prompt

```
You are an expert scientist acting in a text-based science
environment. Maximize task score using the current public
state, current-episode history, and fallible cross-episode
experience.

**CRITICAL CONSTRAINT**: When REFERENCE ACTIONS are provided,
you MUST ONLY choose actions from that list. Any action not
in the REFERENCE ACTIONS list is INVALID and will fail. Do
NOT create custom actions. The list is unordered - position
doesn't indicate quality.
```

D.4 Validation and Recovery

A candidate-validation error appends the following feedback to a retry of the actor user prompt. The error text is supplied by response validation, not an environment-reward evaluator.

Validation feedback: appended to a retry

```
VALIDATION FEEDBACK FROM THE PREVIOUS ATTEMPT:
{validation_error}. Generate a fresh response that fixes
this exact problem and uses only the reference actions.
```

ScienceWorld additionally permits a constrained single-action request after an overlong candidate response cannot be recovered through the normal retries. The history block below is omitted when no planning context is available. This extra recovery request is not part of the Jericho reference's source-style fallback.

ScienceWorld recovery: system prompt

```
You are an expert scientist acting in a text-based science
environment. Maximize task score using the current public
```

```
state, current-episode history, and fallible cross-episode
experience. Output one concise executable command. Return
only the required JSON.
```

ScienceWorld recovery: user prompt

```
GAME HISTORY:
{planning_context}

CURRENT STATE:
{state_text}

REFERENCE ACTIONS (choose exactly one exact string):
{valid_actions}

Return {"action": <one reference action>}.
```

E Additional Experiments

E.1 Backbone Comparison

Table 7 shows that TEMPA achieves the highest Avg and Final-5 on all three games with Gemma-4-31B-it. Gains vary by game: Final-5 exceeds the strongest baseline by 38.6% on Zork1 and 26.6% on ZTUU, while Pentari shows the largest Avg gain at 100.4%. TEMPA's Final-5 also exceeds its own Avg in every game, reflecting stronger performance in later episodes. Together with the Qwen results, these findings show that TEMPA improves adaptation across different LLM backbones.

Table 7: Performance with Gemma-4-31B-it on three Jericho games.

	Zork1		Pentari		ZTUU	
Method	Avg	Final-5	Avg	Final-5	Avg	Final-5
Static	13.93	18.00	7.10	7.00	0.81	0.67
Memory	18.36	21.67	8.23	10.00	5.65	4.20
Reflexion	36.26	33.20	14.50	25.00	8.46	10.27
AWM	19.17	20.00	4.40	5.00	0.00	0.00
EvoTest	34.45	34.00	8.10	8.67	2.77	5.00
APEX	36.45	41.67	13.23	17.33	6.62	10.00
JitRL	38.92	42.00	14.57	13.33	1.03	1.07
TEMPA	**47.66**	**58.20**	**29.20**	**34.33**	**9.75**	**13.00**

E.2 Execution Sources

Figure 4 shows that memory accounts for 43.5–86.2% of executed steps in Jericho, with the largest shares on Detective and Ludicorp. The balance also varies across ScienceWorld settings: memory supplies 72.7% of steps on chemistry mixing and 68.7% on melting point, compared with 21.2% on plant growth. Plant growth relies most heavily on the actor and also has the lowest Final-3 score among these settings. These execution patterns show that stored experience supplies a substantial share of environment actions. The relative contributions of actor generation and memory execution vary across environments.

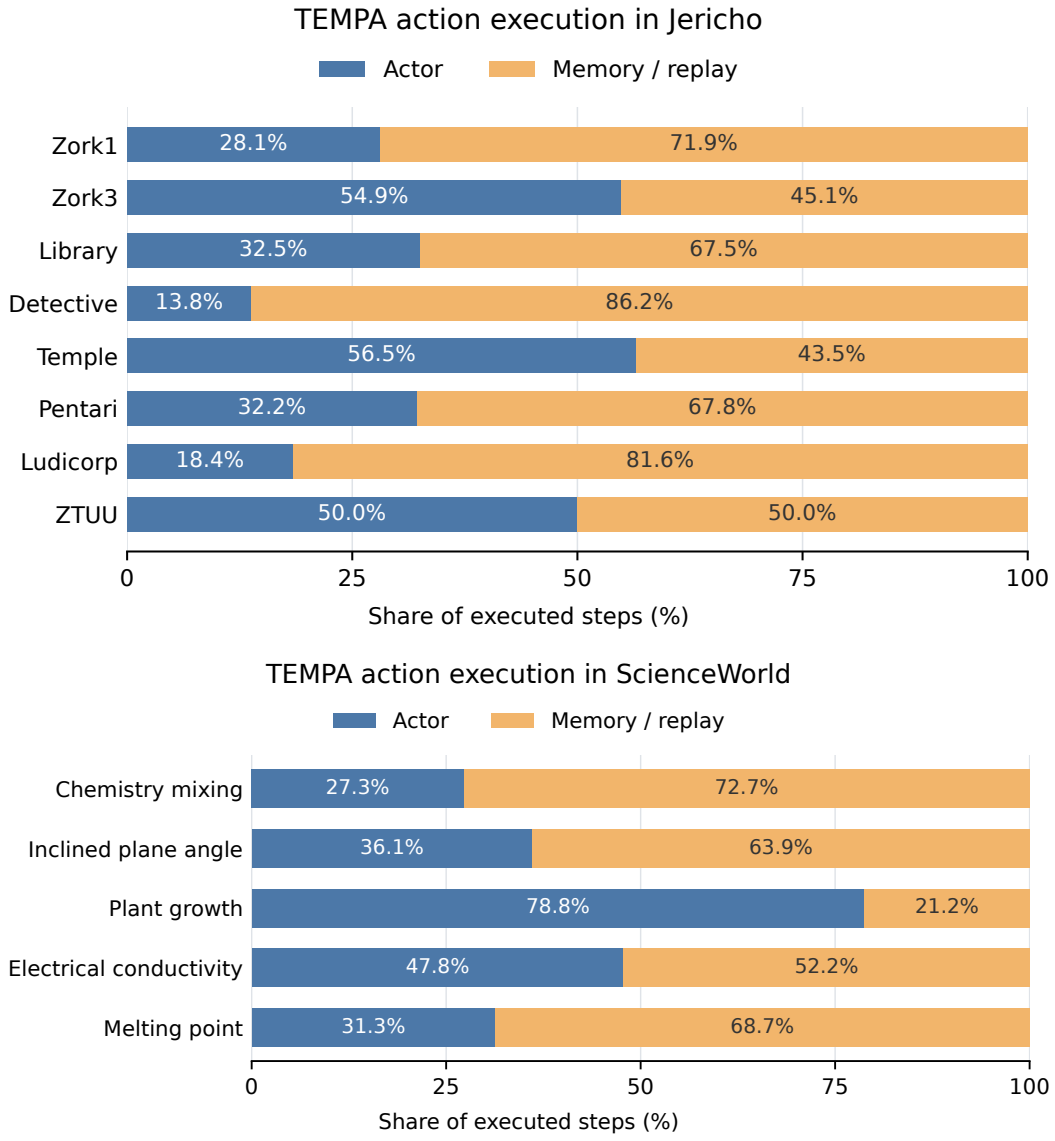


Figure 4: Execution sources in Jericho (top) and ScienceWorld (bottom). Within each game or setting, shares pool actual executed steps across all seeds and episodes. Memory includes reward-segment execution and replay of stored action programs. Runs are weighted by their executed step counts.