REUSE, REFRESH, OR REPLAN? SELECTIVE DECISION UPDATING FOR REAL-TIME LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-time large language models (LLMs) agents must decide whether to reuse a previous decision or update it as the environment evolves. Reuse saves computational cost when the decision remains valid, but an inherited action can mislead the agent once the state changes. Controlled intervention experiments show that prior-action carryover benefits agent performance under safe conditions, yet induces decision anchoring once the action becomes unsafe. Explicit action-outcome information improves the selection of alternative actions. Motivated by these observations, we study selective decision updating supervised via simulator replay. A low-cost model predicts the action for the current state alongside an action-treatment label, while a controller estimates whether further reasoning could correct the action before fallback. The controller triggers AgileThinker if the candidate is unusable or its estimated risk exceeds a threshold selected through validation. On held-out seeds and states, selective fallback trails full AgileThinker by 0.6 percentage points in one-step reference-set accuracy, while reducing mean completion tokens by nearly 60% and invoking the thinking model on about 44% of requests. Collectively, these results show how the expected benefit of further reasoning can guide computational resource allocation for agents updating decisions in dynamic environments, within the evaluated offline, one-step task.

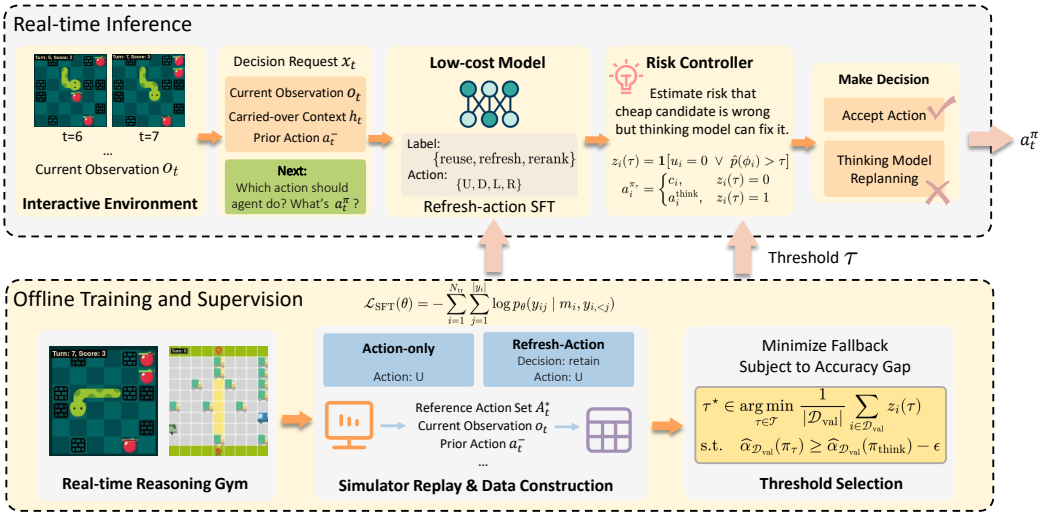


Figure 1: Selective refresh. Offline supervision trains the low-cost action interface, and validation selects the fallback threshold. The controller accepts low-risk candidates or invokes thinking-model fallback.

1 INTRODUCTION

In real-time games, smart home systems, autonomous driving and robotics, agents act as observations change (Gonzalez-Pumariega et al., 2025; Seo et al., 2026; Cao et al., 2026b). If prior decisions

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.

remain valid, agents can reuse them without replanning at every timestep. Environmental changes can also invalidate an inherited decision. Blind reuse risks outdated actions, whereas unconditional replanning wastes computational resources on decisions that are still valid. The agent must choose when to reuse, update cheaply, or replan (Snell et al., 2025; Kang et al., 2025). How should computation depend on whether an inherited decision remains valid?

AgileThinker coordinates extended planning with rapid responses to new observations (Wen et al., 2025). Other methods reuse reasoning templates or route queries to balance quality and cost (Yang et al., 2024; Ong et al., 2025). These mechanisms still require deciding when state changes invalidate an inherited decision. We study this relationship and develop a control interface for reuse, refresh, and replan. The interface must distinguish valid reuse from necessary revisions and identify errors that further planning can correct. We first test when state changes invalidate prior decisions and how carryover affects current decisions.

Controlled diagnostics show a larger effect of crossing an action-validity boundary than of decision age under the tested protocol. Carryover helps when the prior action remains safe yet harms safety when the action becomes unsafe, even when the agent receives updated observation. Oracle validity information improves crossed-state decisions, but self-generated validity judgments do not reliably match it. Simulator-derived outcomes improve action selection where raw descriptions of successor states are inadequate. These findings motivate our framework: predicting actions for the current state with selectively activated supplementary planning.

Selective refresh mechanism coordinates reuse, low-cost updates, and replanning for inherited decisions. A supervised low-cost path uses the current observation and prior decision to produce an action with action-treatment label, supervised by simulator-generated actions and protocol annotations. The controller scores recoverable errors: incorrect low-cost actions for which the thinking model produces a viable alternative. A validation-selected threshold accepts low-risk usable candidates and otherwise invokes the thinking model.

We evaluate offline game decisions, including fresh seeds and states disjoint from risk controller's training and validation. Selective refresh invokes thinking on 43.75% of requests and reduces mean completion tokens by 57.33%. It reaches 96.04% accuracy versus 96.65% for the reference (-0.61 points; 95% state-cluster interval $[-2.90, 1.22]$), without establishing non-inferiority. These results support selective refresh, with a small aggregate drop in observed accuracy under the tested setting.

Our contributions are threefold:

- **Diagnosing decision reuse under state change.** We construct paired interventions that separate the age of an inherited decision from changes in its action validity, and identify how prior actions, carried-over facts, and action–outcome information affect current-state decisions.
- **A low-cost decision-update interface.** We use simulator replay to supervise current-state action prediction together with a structured label for treating an inherited action, making decision updating an explicit output of the agent.
- **Fallback targeted at recoverable errors.** Under runtime-observable signals, we estimate when a low-cost action is likely to fail while additional thinking would recover an acceptable action, and use this estimate to allocate fallback computation on states.

2 Related Work

Reasoning and reuse in evolving environments. Prior work coordinates reasoning, revises hierarchical plans, and manages evolving context (Zhang et al., 2025b; Wan et al., 2026; Cao et al., 2026a; Cao & Yuan, 2025). AgileThinker couples ongoing planning with a reactive thread using current observations and partial plans (Wen et al., 2025). The Buffer of Thoughts retrieves high-level reasoning templates, while Agentic Plan Caching adapts cached plan templates to new task contexts (Yang et al., 2024; Zhang et al., 2025a). These methods reuse planning traces or templates; memory-based agents reuse subgoals or trajectories (Hu et al., 2025; Zheng et al., 2024). Our work examines the continued use of a concrete action inherited from an earlier state. Controlled state interventions test action carryover, while supervised updating enables selective AgileThinker fallback. The selective refresh thereby provides a decision-update and computation-allocation interface.

Cost-aware routing and cascading. Routing and cascading allocate costly inference using predicted quality. RouteLLM learns from pairwise preference data to predict the relative advantage of a stronger model and selects a model before generating its response (Ong et al., 2025). Online cascade learning trains small models with a deferral policy from streamed LLM demonstrations (Nie et al., 2024). Unified routing integrates model selection and sequential escalation to achieve a favourable cost-quality trade-off (Dekoninck et al., 2025). In contrast, our controller observes the cheap candidate action before deciding whether to escalate. Its target is reference-relative recoverable error: the candidate fails the simulator reference but the fallback passes. We apply selective computation to inherited decisions under state change, supported by paired diagnostics and offline action-level evaluation.

3 Problem Formulation

We study reuse, refresh, and replan as observations change. At decision step t, the agent receives

$$x_t = (o_t, h_t, a_t^-, \mathcal{L}_t), \tag{1}$$

Here o_t is the observation, h_t the inherited context, a_t^- the prior action, and $\mathcal{L}_t$ the legal-action set. When no prior action is available, we write $a_t^- = \perp$.

The suitability of a prior action depends on its relationship to the current state, as changing reachability constraints also illustrate (Wang et al., 2025). Let $\mathcal{A}_t^\star \subseteq \mathcal{L}_t$ denote the reference action set obtained from the simulator for the current request. For a request with an available prior action, we define its validity as

$$v_t = \mathbf{1}\{a_t^- \in \mathcal{A}_t^\star\}. \tag{2}$$

Thus, $v_t = 1$ denotes a reference-acceptable prior action and $v_t = 0$ an unacceptable one. Observations may change without invalidating the prior action.

We measure decision quality by whether the final action belongs to the current reference set. Let $a_t^\pi \in \mathcal{L}_t \cup \{\perp\}$ be the final action, where $\perp$ means no usable action. On an evaluation set $\mathcal{D}$ containing N requests, synchronization accuracy is

$$\widehat{\alpha}_{\mathcal{D}}(\pi) = \frac{1}{N} \sum_{t \in \mathcal{D}} \mathbf{1}\{a_t^\pi \in \mathcal{A}_t^\star\}. \tag{3}$$

Only reference-acceptable actions receive credit. When several actions are acceptable, any of them receives credit. The metric applies to every route.

At the decision level, reuse, $\mathrm{refresh}$, and replan govern inherited actions and context. Reuse returns a_t^-; refresh predicts a_t^{ref} cheaply from the current state; replan invokes thinking for a_t^{think}. We denote these computational routes by $r_t^\pi \in \{\mathrm{reuse}, \mathrm{refresh}, \mathrm{replan}\}$. The final action is therefore

$$a_t^\pi = \begin{cases} a_t^-, & r_t^\pi = \mathrm{reuse}, \\ a_t^{\mathrm{ref}}, & r_t^\pi = \mathrm{refresh}, \\ a_t^{\mathrm{think}}, & r_t^\pi = \mathrm{replan}. \end{cases} \tag{4}$$

The reuse route requires an available prior action. These decision-level routes differ from the low-cost model's $\mathrm{retain}/\mathrm{refresh}/\mathrm{rerank}$ labels for the inherited action. Let C_t^π denote the inference cost incurred in selecting the route and producing the final action, and let π_{think} denote the policy that invokes the thinking model for every request. We formulate the computation-allocation goal as

$$\min_{\pi \in \Pi} \frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_{t \in \mathcal{D}_{\mathrm{val}}} C_t^\pi \quad \text{s.t.} \quad \widehat{\alpha}_{\mathcal{D}_{\mathrm{val}}}(\pi) \geq \widehat{\alpha}_{\mathcal{D}_{\mathrm{val}}}(\pi_{\mathrm{think}}) - \epsilon, \tag{5}$$

Here Π is the policy class, $\mathcal{D}_{\mathrm{val}}$ the validation set, and $\epsilon \geq 0$ the allowed accuracy loss.

4 Diagnosis of Stale-Action Failures

Decision updating requires evaluating the validity of inherited actions and selecting appropriate alternatives. Three diagnostics examine both requirements. First, we separate action validity from decision age. Second, we separate prior actions from facts and compare self-generated labels with oracle validity labels. Third, we test alternative actions using successor states and consequences.

4.1 When Does Action Carryover Become Harmful?

We separate action validity from decision age. For each action and age, a common parent yields preserved and crossed successors where that action is safe or legal but unsafe. Both states contain safe alternatives and unsafe actions. We test each state under two conditions: with and without carryover of prior safety judgments, the inherited action recommendation and the reassessment prompt.

For regime $g \in \{\text{preserved}, \text{crossed}\}$ and age k, define the paired effect as

$$\widehat{\Delta}_{g,k} = \frac{1}{n_{g,k}} \sum_{i=1}^{n_{g,k}} \left(Y_{i,g,k}^{\text{carry}} - Y_{i,g,k}^{\text{none}} \right), \tag{6}$$

Here Y indicates a parseable, immediately safe action and $n_{g,k}$ counts paired evaluations. Thus, $\widehat{\Delta}_{g,k}$ measures the change in safety. We also average paired effects across formal cells: $\widehat{\Delta}_g = \widehat{\mathbb{E}}_{c \in \mathcal{C}_g}[\widehat{\Delta}_{g,c}]$. We compare

$$\widehat{B} = \widehat{\Delta}_{\text{preserved}} - \widehat{\Delta}_{\text{crossed}}, \qquad \widehat{A} = \max_g \left| \widehat{\Delta}_{g,8} - \widehat{\Delta}_{g,1} \right|. \tag{7}$$

Here $\widehat{B}$ contrasts validity and $\widehat{A}$ compares ages 1 and 8.

Carryover improves safety by 23.75 percentage points for preserved states and reduces it by 48.75 percentage points for crossed states. Table 1 reports the 72.50-point boundary contrast. Their difference has a confidence interval entirely above zero. Current validity matters more than elapsed steps here. Figure 2 shows that carryover helps preserved states and harms crossed states at every age. Stale prior information harms crossed states at age 1 but helps preserved states at age 8. Neither sequence exhibits a monotonic deterioration with age.

Table 1: Boundary and age contrasts (pp), with 95% bootstrap intervals.

Contrast	Estimate	95% CI
Boundary $\widehat{B}$	72.50	[57.62, 87.73]
Age $\widehat{A}$	20.83	[6.71, 54.80]
$\widehat{B} - \widehat{A}$	51.67	[13.73, 73.33]

We further identify actions behind carryover failures. Table 2 shows more prior-action selections with carryover in crossed states. An unsafe flip switches a safe choice to an unsafe one under carryover. Most flips follow the inherited recommendation, linking it to the penalty. This concentration suggests anchoring on the inherited action even after it becomes unsafe. Memory-augmented agents show a related tendency to follow past experience (Xiong et al., 2026). We next separate the effects of old facts and the explicit action recommendation.

4.2 What Drives Anchoring, and Can Validity Judgments Correct It?

Carryover integrates an inherited recommendation with factual judgments. We separate them across 120 paired bundles with a frozen model. Conditions include four settings: no, action-only, facts-only, and combined carryover. Facts-only lists earlier safe and unsafe actions without a recommendation.

Table 2: Prior-action following and carryover-induced unsafe flips.

Crossed-state prior-action following	
Without carryover	29.17%
With carryover	87.50%
Carryover-induced unsafe flips	
Total flips	133
Returning to prior action	118
Share returning	88.72%

Figure 3 shows that the explicit action yields the largest preserved benefit and crossed penalty. Old facts also harm crossed-state safety, though less. Combining both remains harmful despite a reassessment instruction. The action recommendation produces stronger anchoring than old facts alone. Deleting history forfeits useful information in valid states.

Let $\widehat{\Delta}_g^m$ be the paired safety effect of carryover m versus none. The action–facts contrast is $\widehat{G} = \widehat{\Delta}_{\text{crossed}}^{\text{action}} - \widehat{\Delta}_{\text{crossed}}^{\text{facts}}$. The contrast is -36.67 points (95% cluster-bootstrap interval $[-45.00, -26.67]$, 10,000 resamples). Old facts may impair decisions, but explicit old-action recommendations correspond to larger crossed-state losses.

Can validity judgments preserve useful reuse while avoiding harmful reuse? We compare no prior action, direct carryover, self-gating, and an oracle validity label. Self-gating predicts action validity and the candidate action. Oracle gating supplies simulator-derived validity before action selection.

Providing the oracle validity label substantially changes how the model uses the inherited action. As shown in Figure 4 (left), oracle gating improves crossed-state safety by 83.75 percentage points compared with direct prior-action input while preserving its benefit in preserved states. Correct validity information improves the selection of safe alternatives. Self-gating fails to reproduce this selectivity. It recovers safety for crossed states but falls below no-prior safety in preserved states. Abandoning the inherited action under the right conditions remains difficult.

Recognition is asymmetric across regimes. Table 3 shows that the model frequently identifies an invalid prior action but rarely recognizes a valid one. Mean valid/invalid recognition is near 50%. The bias toward invalidity aligns with lost useful reuse under self-gating. Oracle validity and self-judgment confer different benefits. Revision also requires choosing a suitable alternative.

4.3 Can the Model Evaluate Alternatives Once the Prior Action Becomes Invalid?

Detecting an invalid action does not identify its replacement (Xia et al., 2025). Other agents evaluate candidate actions using goal-conditioned values or aligned world models (Hong et al., 2025; Zhou et al., 2025). We test two approaches: candidate-outcome extraction and correct action attribution. The first diagnostic varies information about two candidate actions. The current-state condition provides the observation, inherited prior action, candidates, and task objective. Successor-state adds each candidate's next state without safety or progress labels. The outcome-vector condition supplies simulator-derived safety and progress values. Safety is simulated failure-avoidance probability over the specified horizon. Progress is positive rewards in Snake or the farthest Freeway lane.

Table 3: Validity recognition across the formal diagnostic cases.

Correct judgment	**Count**	**Rate** (%)
Valid in preserved	29/240	12.08
Invalid in crossed	201/240	83.75
Balanced accuracy (%)		47.92
95% CI (%)		[43.64, 51.29]

We separate outcome comparison from action selection. Comparison determines which candidate action yields greater safety or higher progress. Action selection identifies the action to execute under a safety or progress objective. Let $e \in \{\text{current}, \text{successor}, \text{vector}\}$ denote input type and $\mathcal{Q}_{\text{cmp}}, \mathcal{Q}_{\text{dec}}$ the question groups. For q, the higher token score after `Action:` selects the candidate. Writing this choice as $\widehat{a}^{\,e}(q)$ and the simulator-defined answer as $a^\star(q)$, we measure

$$\widehat{\alpha}^{\,e}_{\mathcal{Q}} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbf{1}\left\{\widehat{a}^{\,e}(q) = a^\star(q)\right\}, \mathcal{Q} \in \{\mathcal{Q}_{\text{cmp}}, \mathcal{Q}_{\text{dec}}\}. \tag{8}$$

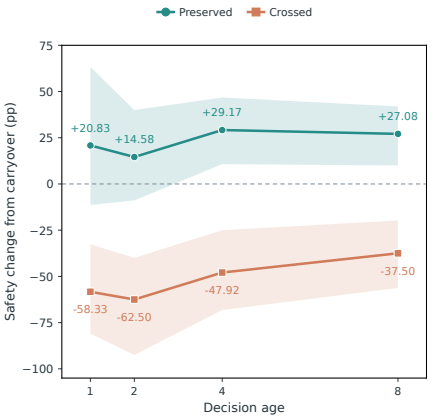


Figure 2: Carryover effects on immediate safety (pp). Shading denotes 95% bootstrap intervals.

Figure 4 (right) shows that successor-state descriptions help, but action-selection accuracy remains below 60%. Explicit outcome values yield nearly perfect comparisons and action selection under both objectives. Directly provided safety and progress values work better than extracting them from these descriptions. This difficulty in extracting decision values from successor states echoes reports of inconsistent use of predicted futures (Qian et al., 2026).

The next diagnostic tests action–consequence correspondence. Four variants share the same decision problem. The baseline provides the observation and legal actions. State-change variant adds no action consequences. Correct-consequence variant lists each action's successor observation and termination or reset status. Mismatched-consequences variant assigns these descriptions to different actions. For example, we swap consequences for actions a and b. This tests whether choices depend on correct action–outcome pairing.

We evaluate action safety and prior-action repetition. Let $\widehat{a}^{\,v}$ be the selected action, $\mathcal{S}(s)$ the immediately safe set, and a^{-} the prior action. For regime $g \in \{\text{preserved}, \text{crossed}\}$, we define

$$\widehat{S}_g^{\,v} = \widehat{\mathbb{E}}_g[\mathbf{1}\{\widehat{a}^{\,v} \in \mathcal{S}(s)\}], \qquad \widehat{R}_g^{\,v} = \widehat{\mathbb{E}}_g[\mathbf{1}\{\widehat{a}^{\,v} = a^{-}\}]. \tag{9}$$

Here, $\widehat{\mathbb{E}}_g$ averages over evaluations in regime g. $\widehat{S}_g^{\,v}$ measures immediate safety; $\widehat{R}_g^{\,v}$ measures prior-action repetition.

Figure 5(left) shows that the description of the state change alone does not improve crossed-state safety over the current-state baseline. Correct consequences improve safety and reduce unsafe-action repetition (Figure 5(right)). Misassigned consequences reduce safety below the baseline and increase repetition. The action–consequence mapping therefore matters beyond simply providing access to consequences.

The crossed-state safety contrast is $\widehat{\Gamma}_{\text{corr}} = \widehat{S}^{\text{correct}}_{\text{crossed}} - \widehat{S}^{\text{mismatched}}_{\text{crossed}}$. The contrast is 69.79 percentage points (95% CI, [59.90, 79.18]), while correct consequences improve safety over the current-state baseline by 39.06 percentage points (95% CI, [29.69, 48.44]). Correct consequences also improve preserved-state safety.

These diagnostics separate access to updated states from action evaluation. Explicit values and correct action–outcome assignments help where state descriptions alone are insufficient.

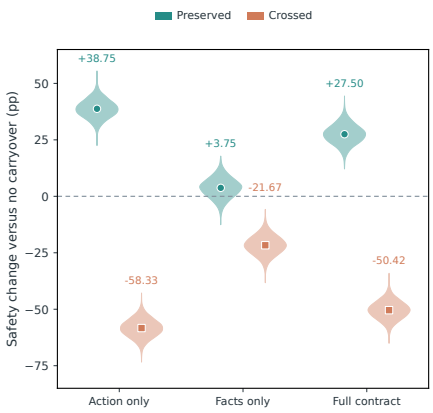


Figure 3: Carryover effects relative to no carryover (pp).

5 Selective Refresh with Runtime Risk Control

Decision updating requires treating the inherited decision and selecting a current action. Figure 1 shows training and runtime flow. The low-cost path predicts an action and $\text{retain}/\text{refresh}/\text{rerank}$ label from current and inherited context. Simulator replay supplies action references while training records supply action-treatment labels. The label informs risk scoring but does not certify old-action invalidity. The risk controller estimates whether thinking could correct the cheap candidate. It accepts usable low-risk candidates below a validation-selected threshold and otherwise invokes AgileThinker.

5.1 Supervised Decision Updating

Simulator replay provides supervision for decision updating. Cloned simulator replays identify legal actions avoiding reset and unsafe termination. They form $\mathcal{A}_t^{\star}$: training targets one stored action, while evaluation accepts any reference action.

Action-only predicts an action. Refresh-action predicts a treatment label followed by that action. These labels concern the prior action, whereas the routes in Equation 4 concern the inherited decision and its computation. Appendix K details label construction and nominally invalid but reference-acceptable actions.

Both formats fine-tune the same underlying backbone separately. Let m_i contain the observation, legal actions, objective, inherited prior action, and decision age, and let y_i be the target output completion. We optimize the autoregressive completion loss:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{i=1}^{N_{\text{tr}}} \sum_{j=1}^{|y_i|} \log p_\theta(y_{ij} \mid m_i, y_{i,<j}) \,. \tag{10}$$

Here θ is trainable, N_{tr} counts examples, and only completion tokens contribute to loss. For Refresh-action, one completion supervises both the label and action. At inference time, the controller accepts the candidate or invokes thinking using the output label and action.

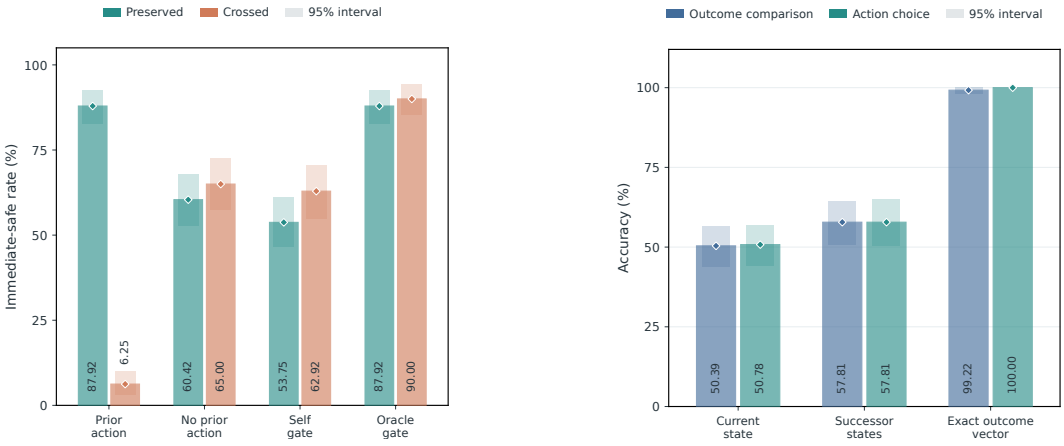


Figure 4: Validity interventions and outcome information. Panel (left) shows immediate safety under validity interventions, while panel (right) shows outcome comparison and action choice under three information conditions.

5.2 Runtime Estimation of Recoverable Errors

The controller estimates whether thinking can correct a low cost candidate. Training pairs cheap and thinking outputs on requests with identical observations, inherited decisions, and reference sets. For pairs in which both outputs are usable, we define a recoverable error as

$$e_i^{\text{rec}} = \mathbf{1}\left[c_i \notin \mathcal{A}_i^\star \ \wedge \ a_i^{\text{think}} \in \mathcal{A}_i^\star\right], \tag{11}$$

Here c_i is the cheap candidate and a_i^{think} the thinking action. The target excludes errors shared by both paths.

Features ϕ_i are available after cheap generation and before thinking. They encode the label, output usability and legality, prior-action agreement, age, and generation length.

The scorer buckets usable pairs by exact feature matches; $b(\phi_i)$ indexes each bucket. With n_b usable pairs and m_b recoverable errors, Laplace smoothing gives

$$\widehat{p}_b = \frac{m_b + 1}{n_b + 2}. \tag{12}$$

The same smoothing gives the global estimate:

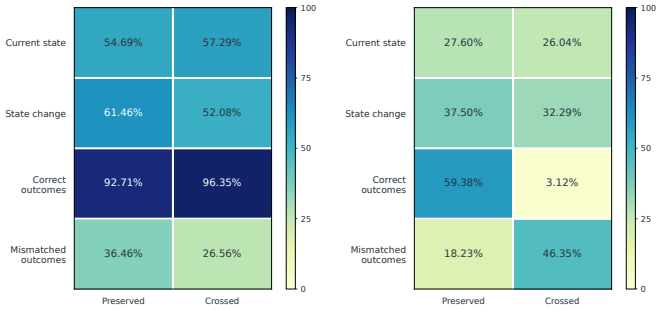


Figure 5: Action–consequence correspondence under matched diagnostic conditions. Panel (left) reports immediate safety, and panel (right) reports prior-action repetition.

$$\widehat{p}_{\text{global}} = \frac{\sum_{b \in \mathcal{B}_{\text{tr}}} m_b + 1}{\sum_{b \in \mathcal{B}_{\text{tr}}} n_b + 2}, \tag{13}$$

where $\mathcal{B}_{\text{tr}}$ is the set of feature groups observed during training. Risk uses the matching bucket, or the global estimate for unseen buckets:

$$\widehat{p}(\phi_i) = \begin{cases} \widehat{p}_{b(\phi_i)}, & b(\phi_i) \in \mathcal{B}_{\text{tr}}, \\ \widehat{p}_{\text{global}}, & \text{otherwise.} \end{cases} \tag{14}$$

This score controls thinking-model fallback.

5.3 Selective Fallback and Threshold Selection

We obtain c_i, its usability u_i, and the risk score in Equation 14. For legal actions $\mathcal{L}_i$, usability is $u_i = \mathbf{1}\{c_i \in \mathcal{L}_i\}$. Parse failures and illegal outputs have $u_i = 0$.

At threshold τ, the policy accepts usable low-risk candidates and otherwise invokes thinking:

$$z_i(\tau) = \mathbf{1}[u_i = 0 \ \vee \ \widehat{p}(\phi_i) > \tau], \qquad a_i^{\pi_\tau} = \begin{cases} c_i, & z_i(\tau) = 0, \\ a_i^{\text{think}}, & z_i(\tau) = 1, \end{cases} \tag{15}$$

Here $z_i(\tau) = 1$ invokes thinking. Validation selects the lowest call rate under an allowed accuracy gap ϵ (Ding et al., 2025). Let $\mathcal{T}$ be the validation thresholds. Then

$$\tau^\star = \arg\min_{\tau \in \mathcal{T}} \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{i \in \mathcal{D}_{\text{val}}} z_i(\tau) \quad \text{s.t.} \quad \widehat{\alpha}_{\mathcal{D}_{\text{val}}}(\pi_\tau) \geq \widehat{\alpha}_{\mathcal{D}_{\text{val}}}(\pi_{\text{think}}) - \epsilon. \tag{16}$$

We freeze $\tau^\star$ for testing. Held-out evaluation keeps that threshold fixed. Appendix A derives paired accuracy and token costs.

6 Experiment

6.1 Experimental Setup

We evaluate offline Freeway, Snake and Overcooked requests from RealtimeGym (Wen et al., 2025). Experiments test data selection, output format, fixed-budget routing and fresh-seed fallback.

Table 4: Synchronization accuracy on the same held-out requests. Δ_{base} denotes the gain over unadapted reactive, and Δ_{sel} denotes Diagnostic minus Random within each output format.

Method	**Selection**	$\widehat{\alpha}_{\mathcal{D}}$ (%)	Δ_{base}	Δ_{sel}
Unadapted reactive	–	86.18	0.00	–
Action-only	Random	88.13	+1.95	–
	Diagnostic	90.48	+4.30	+2.34
Refresh-action	Random	88.67	+2.49	–
	Diagnostic	90.28	+4.10	+1.61

Matched training compares diagnostic and random selection. Paired requests compare Action-only and Refresh-action variants. Fixed-budget routing compares buckets, random, fixed-rule, single-feature, and logistic baselines under equal call rates. Fresh-seed routing freezes a validation-selected threshold before test-time AgileThinker fallback. Metrics are synchronization accuracy, thinking calls, completion tokens, and recorded-path time estimates. Intervals use state-cluster bootstraps. For random routing, intervals instead vary routing seeds.

Appendices B and C specify evaluation sets and model settings. Appendix D specifies the routing configurations.

6.2 Diagnostic-Guided Data Selection

We test whether diagnostically selected training examples improve action prediction. Action-only and Refresh-action each compare Diagnostic with length-stratified Random selection against unadapted reactive. Within each format, strategies match record and token counts and training protocols, excluding test data from selection. Appendices B and I detail sampling and reactive inference.

Table 4 reports 90.48% and 90.28% for diagnostic Action-only and Refresh-action, 4.30 and 4.10 points above reactive. Diagnostic exceeds Random by 2.34 points for Action-only and 1.61 for Refresh-action. Both positive differences support diagnostic selection under this protocol. Figure 6(left) shows training trajectories. We next compare output formats and fallback.

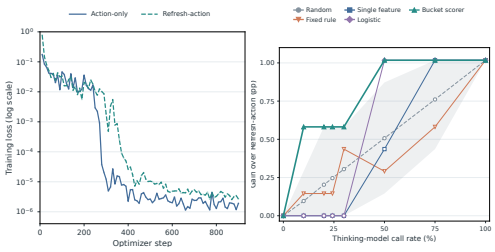


Figure 6: Panel (left) reports training loss for Action-only and Refresh-action on diagnostic-selected data. Panel (right) reports synchronization gains over standalone Refresh-action at a shared 24.27% thinking-model call rate.

6.3 Action-Only versus Structured Decision Updating

We compare decision reuse, Action-only, and Refresh-action. Table 5 uses identical requests.

Action-only and Refresh-action reach accuracy of 97.67% and 98.98%. Refresh-action is always usable. Action-only has nine unavailable outputs and 98.97% accuracy when usable. The structured format exposes the label and uses 12.48 versus 6.00 mean completion tokens.

Both fallback variants reach 99.13% accuracy versus 100.00% for full AgileThinker at roughly one quarter of its completion-token cost. Action-only requires more calls for unavailable outputs. Updating the inherited action improves synchronization over reuse; selective fallback further improves accuracy at extra cost. Appendix F reports time estimates. Next, we test routing at a fixed call budget.

6.4 Risk Routing under a Fixed Planner-Call Budget

We evaluate whether recoverable-error scores allocate thinking calls effectively. On fixed Refresh-action outputs, we compare buckets with random, fixed-rule, selected-feature, and logistic routing. Equal call counts isolate request selection as the source of accuracy differences. Appendix D details baselines.

Figure 6(right) compares routing methods at a fixed call rate. At the shared budget, bucket routing exceeds random by 0.335 points with fewer mean tokens. It recovers four of seven recoverable errors, versus 1.69 for random and one for fixed-rule routing. Single-feature and logistic recover none at this budget but use fewer tokens. The descriptive advantage and recovery counts appear in Appendix G. We next test a validation-selected threshold on fresh seeds.

Table 5: Quality and cost on the common paired test set.

Method	$\widehat{\alpha}_{\mathcal{D}}$ (%)	**Fallback calls** (%)	**Tokens**
Action-only	97.67	0.00	6.00
Refresh-action	98.98	0.00	12.48
Action-only + fallback	99.13	25.73	516.14
Refresh-action + fallback	99.13	25.00	514.18
Full AgileThinker	100.00	100.00	2,098.74

6.5 Selective Fallback on Fresh Seeds

We finally evaluate validation-selected fallback on held-out fresh-seed states. Unlike fixed-budget routing, the validation threshold remains fixed across all test games. Refresh-action, selective fallback, and full AgileThinker share requests and reference sets. Appendix B gives splits.

Table 6: Quality and cost on the fresh-seed test set.

Policy	$\widehat{\alpha}_{\mathcal{D}}$ (%)	**Fallback calls** (%)	**Tokens**
Refresh-action	90.55	0.00	12.53
Full AgileThinker	96.65	100.00	1,876.73
Selective fallback	96.04	43.75	800.73

Table 6 shows accuracy rising from 90.55% to 96.04% (+5.49 points). Fallback is invoked on 43.75% of requests, cutting mean tokens by 57.33% versus full AgileThinker. Recorded-path runtime drops by an estimated 56.41% (Appendix H). Accuracy trails full AgileThinker by 0.61 points. State-cluster analysis does not establish non-inferiority.

7 Discussion and Conclusion

Decision reuse requires checking inherited action applicability and potential benefits of further reasoning. Paired diagnostics expose anchoring and limits in reading action consequences. Selective refresh combines supervised action prediction with bucket-based fallback. Candidate-level risk routing reduces cost, but Freeway shows that fallback can also harm accuracy.

Long-horizon task quality and large-scale online latency remain unverified. In audited fresh-seed Overcooked instances, every legal action passes the reference, so accuracy cannot measure collaboration or order completion. Future work should test episode returns and task-specific objectives (Ma et al., 2024; Paglieri et al., 2025), and more seeds and environments.

AI USE STATEMENT

Generative AI tools were utilized to assist with the implementation of partial methodological code, polishing manuscript language, organizing the structure and presentation of the paper and identifying and surveying relevant literature. All AI-assisted code was reviewed and checked by the authors. The authors also critically reviewed AI-assisted writing and literature suggestions and take full responsibility for the final manuscript, including its claims, analyses, code, figures and references.

REPRODUCIBILITY STATEMENT

Section 6.1 describes the evaluation protocol and comparisons. Appendix B records the seed groups and data construction, Appendix C gives model and training configurations, and Appendix D specifies the controller features and routing baselines. Appendix E records software settings. The source code is available at `https://anonymous.4open.science/r/ICLR37753-6BE9/`

REFERENCES

Shijie Cao and Yuan Yuan. Reflecsched: Solving dynamic flexible job-shop scheduling via llm-powered hierarchical reflection. *CoRR*, abs/2508.01724, 2025. doi: 10.48550/ARXIV.2508.01724. URL https://doi.org/10.48550/arXiv.2508.01724.

Shijie Cao, Yuan Yuan, and Jing Liu. Harmonizing real-time constraints and long-horizon reasoning: An asynchronous agentic framework for dynamic scheduling. In *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence and the 29th European Conference on Artificial Intelligence, IJCAI/ECAI 2026, Bremen, Germany, 15-21 August 2026*, pp. 6146–6155. ijcai.org, 2026a. doi: 10.24963/IJCAI.2026/684. URL https://doi.org/10.24963/ijcai.2026/684.

Shijie Cao, Yuan Yuan, and Jing Liu. Dynaschedbench: Calibrated dynamic scheduling benchmarks and observability paradox in llm-based scheduling agents. *CoRR*, abs/2605.27566, 2026b. doi: 10.48550/ARXIV.2605.27566. URL https://doi.org/10.48550/arXiv.2605.27566.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 12987–13010. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/dekoninck25a.html.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. BEST-route: Adaptive LLM routing with test-time optimal compute. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 13870–13884. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/ding25d.html.

Gonzalo Gonzalez-Pumariega, Leong Yean, Neha Sunkara, and Sanjiban Choudhury. Robotouille: An asynchronous planning benchmark for llm agents. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 83757–83792, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d033407a1a4b7a75cd5f9e4575ad9fb5-Paper-Conference.pdf.

Joey Hong, Anca Dragan, and Sergey Levine. Planning without search: Refining frontier llms with offline goal-conditioned rl. 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. 10032–10056. Curran Associates, Inc., 2025. doi: 10.52202/085713-0338. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0e87f947391341256517bdc5d206e539-Paper-Conference.pdf.

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/.

Hao Kang, Qingru Zhang, Han Cai, Weiyuan Xu, Tushar Krishna, Yilun Du, and Tsachy Weissman. Win fast or lose slow: Balancing speed and accuracy in latency-sensitive decisions of llms. 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. 150862–150884. Curran Associates, Inc., 2025. doi: 10.52202/085713-5044. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ddaec864ba433e8889ab08dcf5c26e55-Paper-Conference.pdf.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 74325–74362. Curran Associates, Inc., 2024. doi: 10.52202/079017-2365. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/877b40688e330a0e2a3fc24084208dfa-Paper-Datasets_and_Benchmarks_Track.pdf.

Lunyiu Nie, Zhimin Ding, Erdong Hu, Christopher Jermaine, and Swarat Chaudhuri. Online cascade learning for efficient inference over streams. 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. 38071–38090. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/nie24a.html.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5503a7c69d48a2f86fc00b3dc09de686-Paper-Conference.pdf.

Davide Paglieri, Bartł omiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Woł czyk, Akbir Khan, Eduardo Pignatelli, Ł ukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Foerster, Jack Parker-Holder, and Tim Rocktaeschel. Balrog: Benchmarking agentic llm and vlm reasoning on games. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 96666–96702, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f0b1515be276f6ba82b4f2b25e50bef0-Paper-Conference.pdf.

Cheng Qian, Emre Can Acikgoz, Bingxuan Li, Xiusi Chen, Yuji Zhang, Bingxiang He, Qinyu Luo, Gokhan Tur, Dilek Hakkani-Tür, Yunzhu Li, and Heng Ji. Current agents fail to leverage world model as tool for foresight. 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. 13686–13723, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.623. URL https://aclanthology.org/2026.acl-long.623/.

Gyuhyeon Seo, Jungwoo Yang, Junseong Pyo, Nalim Kim, Jonggeun Lee, and Yohan Jo. Simuhome: A temporal- and environment-aware benchmark for smart home llm agents. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 40070–40097, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/43456e781fac3a034842f0807860474a-Paper-Conference.pdf.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b623663fd9b874366f3ce019fdfdd44-Paper-Conference.pdf.

Guangya Wan, Mingyang Ling, Xiaoqi Ren, Rujun Han, Sheng Li, and Zizhao Zhang. COMPASS: Enhancing agent long-horizon reasoning with evolving context. 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. 3360–3380, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.152. URL https://aclanthology.org/2026.acl-long.152/.

Dong Wang, Xinghang Li, Zhengshen Zhang, Jirong Liu, Xiao Ma, Hanbo Zhang, Tao Kong, and Huaping Liu. ProcWorld: Benchmarking large model planning in reachability-constrained environments. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12564–12594, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.635. URL https://aclanthology.org/2025.emnlp-main.635/.

Yule Wen, Yixin Ye, Yanzhe Zhang, Diyi Yang, and Hao Zhu. Real-time reasoning agents in evolving environments. *CoRR*, abs/2511.04898, 2025. doi: 10.48550/ARXIV.2511.04898. URL https://doi.org/10.48550/arXiv.2511.04898.

Yu Xia, Yiran Jenny Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, Lina Yao, and Julian McAuley. SAND: Boosting LLM agents with self-taught action deliberation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3062–3077, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.152. URL https://aclanthology.org/2025.emnlp-main.152/.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts LLM agents: An empirical study of experience-following behavior. 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. 623–645, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.27. URL https://aclanthology.org/2026.acl-long.27/.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 113519–113544. Curran Associates, Inc., 2024. doi: 10.52202/079017-3607. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/cde328b7bf6358f5ebb91fe9c539745e-Paper-Conference.pdf.

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., 2025a. doi: 10.52202/085713-3451. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/9549f7d06700f0966d5f938f1d11022a-Paper-Conference.pdf.

Zhenyu Zhang, Tianyi Chen, Weiran Xu, Alex Pentland, and Jiaxin Pei. Recap: Recursive context-aware reasoning and planning for large language model 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.

95158–95186. Curran Associates, Inc., 2025b. doi: 10.52202/085713-3181. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/88af3540325dd0b70617a9ab605f294d-Paper-Conference.pdf.

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

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. Wall-e: World alignment by neurosymbolic learning improves world model-based 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. 65352–65391. Curran Associates, Inc., 2025. doi: 10.52202/085713-2192. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/5e772a13ccba5255331240dcd99aa38b-Paper-Conference.pdf.

A Accuracy and Cost Accounting

For usable paired requests, fallback uses the recorded thinking-model action and acceptance uses the low-cost candidate. The following identity decomposes the policy's accuracy gap into accepted candidates that correct a thinking-model error and accepted candidates whose errors the thinking model would correct. On the common set $\mathcal{P}$ of requests for which both paths produce usable actions, the accuracy difference between the policy and the thinking model arises only from accepted low-cost candidates. Defining $g_i = \mathbf{1}[c_i \in \mathcal{A}_i^\star \wedge a_i^{\text{think}} \notin \mathcal{A}_i^\star]$ as a low-cost correction of a thinking-model error, we obtain

$$\widehat{\alpha}_{\mathcal{P}}(\pi_\tau) - \widehat{\alpha}_{\mathcal{P}}(\pi_{\text{think}}) = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \big(1 - z_i(\tau)\big)\big(g_i - e_i^{\text{rec}}\big). \tag{17}$$

where e_i^{rec} is the recoverable-error indicator defined in Equation 11. The risk estimator targets the loss term e_i^{rec}, while the resulting accuracy difference reflects both corrections gained and recoverable errors accepted.

Every request incurs the cost of low-cost generation, and a fallback request additionally incurs the cost of the thinking model. Writing T_i^{cheap} and T_i^{think} for the completion-token counts of the two paths, the policy cost is

$$T_i^{\pi_\tau} = T_i^{\text{cheap}} + z_i(\tau) T_i^{\text{think}}. \tag{18}$$

For an evaluation request set $\mathcal{D}$, the corresponding mean completion-token costs are

$$\overline{T}^{\pi_\tau} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} T_i^{\pi_\tau}, \qquad \overline{T}^{\text{think}} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} T_i^{\text{think}}. \tag{19}$$

We measure token reduction relative to invoking the thinking model on every request using the ratio of mean costs over the same evaluation requests:

$$R_{\text{token}} = 1 - \frac{\overline{T}^{\pi_\tau}}{\overline{T}^{\text{think}}}. \tag{20}$$

The identity applies on the paired usable subset $\mathcal{P}$. System-level test accuracy also includes requests with unavailable low-cost outputs, which trigger fallback. For recorded path times E_i^{cheap} and E_i^{think}, the analogous offline time estimate is

$$\widehat{E}_i^{\pi_\tau} = E_i^{\text{cheap}} + z_i(\tau) E_i^{\text{think}}. \tag{21}$$

Table 7: Held-out evaluation sets. Output-format and fixed-budget routing comparisons share the same requests; the other evaluations use separate sets.

Evaluation	Requests	States
Data selection	2,048	41
Output format	688	43
Fixed-budget routing	688	43
Fresh-seed evaluation	656	41

Table 8: Seed-separated data sources. The output-format/routing splits support earlier paired comparisons; the new-seed splits fit and evaluate the risk controller.

Dataset	Split	Requests	States	Seeds
SFT source	Training	–	–	0–3
	Validation	–	–	4–5
	Test	–	–	6–7
Output-format/routing	Training	1,456	91	8–11
	Validation	688	43	12–13
	Test	688	43	14–15
New-seed controller	Training	1,296	81	24–27
	Validation	752	47	28–29
	Test	656	41	30–31

B Evaluation Sets and Data Construction

Table 7 reports the evaluation sizes. Table 8 separates the existing SFT data (seeds 0–7), the paired output-format/routing study (seeds 8–15), and the new-seed risk-controller evaluation (seeds 24–31). Requests derived from the same simulator state are treated as a cluster for uncertainty estimation.

State overlap across splits is zero within each controller dataset. The SFT seed ranges describe the data used to train existing checkpoints. The new-seed training and validation rows fit the risk controller and select its threshold. They do not train a new SFT model.

All actions that satisfy the no-reset and terminal-safety constraints constitute the reference set. If no valid action satisfies these predicates, the first legal action is retained as a fallback reference and the request is preserved. Reference membership therefore measures one-step acceptability under this labeling procedure, instead of long-horizon return or an unconditional safety guarantee.

C Training and Inference Configuration

Figure 6(left) shows decreasing logged training loss for both output formats over the recorded optimization steps. These trajectories describe optimization on the selected training data. Table 9 records the verified training settings for the data-selection experiment and the checkpoints used in fresh-seed evaluation.

If truncation leaves no supervised assistant token, the preprocessing raises an error instead of silently retaining the sample. Table 10 separates the matched attribution runs from the previously trained checkpoints reused on fresh seeds.

The low-cost checkpoints use deterministic generation, with the requested token budget as the maximum number of new tokens. The verified prompt path applies the tokenizer chat template and attempts to disable thinking. Action-only produces `\boxed{ACTION}`, while Refresh-action produces a decision label followed by the boxed action. Parsing prioritizes boxed actions and checks action legality.

Thinking-model fallback is executed through AgileThinker using both reactive and planning configurations. Table 11 gives the verified fallback settings for fixed-budget routing and fresh-seed evaluation.

Table 9: Verified supervised fine-tuning configuration. These settings apply to the data-selection experiment and the checkpoints used for fresh-seed evaluation.

Parameter	Setting
Backbone	Qwen3.5-27B
Adaptation	LoRA
LoRA rank / alpha / dropout	$16/32/0.05$
Target modules	`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`
Precision	bfloat16
Optimizer	`AdamW`
Epochs / learning rate	$1/2 \times 10^{-4}$
Per-device train / evaluation batch size	$1/1$
Gradient accumulation steps	8
Maximum sequence length	4,096 tokens
Truncation	Retain the sequence suffix
Training objective	Assistant-completion tokens only, with prompt labels masked
Checkpoint selection	Final saved checkpoint
Training seed	9101

Table 10: Supervised fine-tuning data counts. Training tokens are matched between Random and Diagnostic within each output format, not across formats.

Quantity	Random	Diagnostic	Existing checkpoints
Training requests	7,248	7,248	15,360
Validation requests	8,448	8,448	8,448
Action-only training tokens	1,090,922	1,090,922	-
Refresh-action training tokens	1,135,851	1,135,851	-

D Risk Controller and Routing Baselines

The controller groups requests by exact matches of the complete transformed runtime-feature vector. Table 12 specifies the 17 features and their numerical transformations.

Risk estimates use the Laplace-smoothed bucket and global formulas in Equations 12–14. The fresh-seed risk model is fitted to 1,287 usable training pairs, containing 57 recoverable errors across 48 buckets. Validation considers zero and the distinct observed risk scores as candidate thresholds, maximizes accepted cheap candidates subject to the accuracy constraint, and breaks ties toward the smaller threshold. The selected threshold is 0.0588235294, accepting 474 validation candidates. Table 13 records the fixed-budget comparison settings.

E Software and Execution Environment

Table 14 records the verified software environment and serving settings.

F Paired Output-Format Evaluation

The evaluation pairs Action-only, Refresh-action, and AgileThinker outputs. All five policies use the same 688 requests from 43 state clusters.

Table 15 reports recorded elapsed-time estimates. Hybrid estimates add the cheap-path time to the recorded reference-path time on requests selected for fallback. The full-reference policy uses the same recorded AgileThinker outputs as hybrid fallback, without incurring cheap-path cost. Refresh-action has higher recorded time than Action-only, both independently and with fallback.

Table 11: AgileThinker fallback configuration.

Parameter	Reactive	Planning
Temperature	0.7	1.0
Top-p	0.8	0.95
Top-k	20	20
Presence penalty	1.5	1.5
Configured maximum output tokens	32,768	32,768
Thinking enabled	No	Yes

Table 12: Runtime features used before thinking-model fallback. Feature names follow the implementation.

Group	Features
Decision indicators	`decision_retain`, `decision_refresh`, `decision_rerank`, `decision_missing`
Presence and legality	`prior_action_present`, `gate_action_present`, `gate_action_legal`, `refresh_action_present`, `refresh_action_legal`
Parsing and agreement	`gate_parser_ok`, `gate_timeout`, `prior_gate_action_match`
Request and output	`legal_action_count`, `prior_decision_age`, `gate_completion_tokens`, `gate_output_length`
Candidate usability	`candidate_usable`
Transformations	Booleans map to 0/1. legal-action count is rounded and capped at 16. age, completion tokens, and output length are capped at 32, 256, and 512, respectively

G Fixed-Budget Routing Results

The bucket scorer reaches its full observed gain by the 50% call budget, whereas the single-feature baseline reaches that level at 75%. Logistic regression also reaches that level at 50%, despite recovering no errors at the main 24.27% budget. Table 16 shows the absolute synchronization accuracies and mean token costs at the main budget.

Table 17 reports how effectively each routing method selects recoverable errors. Recovery recall is the fraction of recoverable errors selected for fallback. The single-feature and logistic baselines do not recover any errors at the reported budget. This observation does not establish that their rankings are uninformative at other budgets.

H Fresh-Seed Results by Game

Table 18 reports the complete per-game comparison under the single validation-selected threshold. The slices use the same frozen test policy without retraining or threshold adjustment.

Relative to Refresh-action, selective fallback yields 44 additional correct decisions on Snake and eight fewer on Freeway, with no change on Overcooked. These are net changes in correctness, not counts of all individual corrections or regressions. The Overcooked slice is not informative about cooperation quality under this reference criterion. In the audited Overcooked fresh-seed source rows, all six legal actions belong to the reference set for each of 256 requests from 16 states, and all recorded one-step progress values are zero. The criterion accepts even repeated waiting, so the reported 100% measures membership in a nondiscriminating set, not order completion. A post-hoc sensitivity calculation excluding Overcooked gives 338/400 (84.50%), 374/400 (93.50%), and 378/400 (94.50%) for standalone Refresh-action, selective fallback, and full AgileThinker, respectively. The original all-game results remain reported above.

Standalone Refresh-action produces 652 usable outputs, with four unavailable outputs retained as errors in its 656-request accuracy denominator. Selective fallback makes 283 risk-triggered calls and four calls for unavailable cheap outputs. The full reference uses the same recorded AgileThinker outputs as the fallback policy.

Table 13: Fixed-budget routing configurations. Risk models are fitted on training data, the single feature is selected on validation data, and test data are used only for evaluation.

Setting	Configuration
Invocation budgets	$0\%, 10\%, 20\%, 24.27\%, 30\%, 50\%, 75\%, 100\%$, the primary budget uses 167 of 688 requests
Budget conversion	$k = \mathrm{round}(\rho N)$ for invocation fraction ρ and evaluation size N
Score ordering	Descending score with stable-hash tie-breaking
Fixed route rule	Prioritize outputs whose decision label is not `retain`
Single-feature baseline	Validation-selected feature: `legal_action_count` at the primary budget
Logistic regression	Training-set mean/std standardization with scale floored at 1. zero initialization. 300 gradient-descent iterations. step size $0.2/n$ for n training examples. no explicit L1/L2 penalty or hyperparameter search
Random routing	Exactly k selected requests in each of 1,000 repeats

Table 14: Verified software and execution settings. vLLM settings apply to the fallback path in fixed-budget routing and fresh-seed evaluation.

Component	Recorded configuration
PyTorch / Transformers	2.11.0 / 5.10.2
Accelerate / PEFT	1.14.0 / 0.20.0
vLLM	0.22.1
Maximum model length	65,536 tokens
Maximum sequences	128
Maximum batched tokens	32,768
Chunked prefill / eager execution	Enabled / enabled
GPU model	NVIDIA H100 80GB HBM3
Number of GPUs	4

I Unadapted Reactive Reference

The reference uses an unadapted Qwen3.5-27B model through the vLLM ReactiveAgent path. Table 19 reports the resulting accuracy.

J Deterministic Prior-Action Reuse

The reuse policy returns the runtime-visible prior action without consulting reference labels. Every request has a legal prior action, so no missing-action or illegal-action fallback is required. Table 20 reports the resulting accuracy.

K Audit of Structured Supervision Labels

The action target is the first action in the stored reference list. The decision target uses the stored when available; otherwise applies the condition-to-label mapping described in Section 5.1. The generator attempts to select a prior action outside the safe set for an `invalid` condition but uses the last legal action if no such action exists, retaining the original condition label. Table 21 shows that protocol-defined invalid conditions frequently contain reference-acceptable prior actions.

For example, an `invalid_unchanged` training record has prior action `S` and reference set $\{\mathtt{U}, \mathtt{D}, \mathtt{L}, \mathtt{R}, \mathtt{I}, \mathtt{S}\}$, but its target is `refresh` with action `U`. The prior action is acceptable, yet the protocol annotation requests refresh. **Decision-label agreement therefore cannot by itself establish recognition of strict prior-action invalidity.** Analyses of strictly unacceptable prior actions must use $a_i^- \notin \mathcal{A}_i^\star$. This reference-membership criterion is distinct from action legality.

Table 15: Mean recorded elapsed-time estimates for the output-format comparison.

Method	Recorded time (s/request)
Action-only	1.272
Refresh-action	6.534
Action-only + fallback	9.418
Refresh-action + fallback	14.551
Full AgileThinker reference	33.592

Table 16: Routing comparison at a shared 24.27% thinking-model invocation rate. Tokens are mean completion tokens per request.

Routing method	$\widehat{\alpha}_{\mathcal{D}}$ (%)	**Tokens**
Random	99.23	520.66
Fixed route rule	99.13	547.28
Single feature	98.98	453.55
Logistic regression	98.98	457.54
Bucket scorer	99.56	491.32

Table 17: Error recovery and recorded elapsed-time estimates at the fixed invocation budget. The test set contains 7 recoverable errors.

Routing method	Recovered	Recall (%)	Time (s/request)
Single feature	0	0.00	13.16
Logistic regression	0	0.00	13.27
Fixed route rule	1	14.29	14.99
Random	1.69	24.19	14.67
Bucket scorer	4	57.14	13.85

Table 18: Fresh-seed results by game. Correct gives the number of reference-acceptable final actions over all requests in the slice. Calls denotes thinking-model invocation rate. Tokens and recorded time are means per request.

Game	Policy	Correct	$\widehat{\alpha}_{\mathcal{D}}$ (%)	Fallback calls (%)	Tokens	Time (s)
Freeway	Refresh-action	139/144	96.53	0.00	12.597	2.573
	Selective fallback	131/144	90.97	56.94	1,430.854	121.301
	Full AgileThinker	127/144	88.19	100.00	2,531.653	213.375
Snake	Refresh-action	199/256	77.73	0.00	12.449	2.298
	Selective fallback	243/256	94.92	80.08	1,234.438	105.411
	Full AgileThinker	251/256	98.05	100.00	1,592.754	134.363
Overcooked	Refresh-action	256/256	100.00	0.00	12.566	2.365
	Selective fallback	256/256	100.00	0.00	12.566	2.365
	Full AgileThinker	256/256	100.00	100.00	1,792.320	149.425
Overall	Refresh-action	594/656	90.55	0.00	12.527	2.385
	Selective fallback	630/656	96.04	43.75	800.726	68.686
	Full AgileThinker	634/656	96.65	100.00	1,876.733	157.585

Table 19: Unadapted reactive performance on the matched data-selection subset. These requests are distinct from the historical output-format test and the final fresh-seed test.

Game	Correct / requests	$\widehat{\alpha}_{\mathcal{D}}$ (%)
Freeway	564/621	90.82
Overcooked	719/719	100.00
Snake	482/708	68.08
Overall	1,765/2,048	86.18

Table 20: Deterministic prior-action reuse on the historical paired test. Accuracy measures membership in the request's reference action set.

Game	Correct / requests	$\widehat{\alpha}_{\mathcal{D}}$ (%)
Freeway	201/208	96.63
Overcooked	256/256	100.00
Snake	186/224	83.04
Overall	643/688	93.46

Table 21: Structured supervision audit. Accepted prior counts the records whose prior action belongs to the reference set. Every record within a listed row receives the indicated decision label, including those with an accepted prior action.

Training data	Condition	Records	Accepted prior	Decision label
Diagnostic	`invalid_changed`	1,883	1,050	`rerank`
	`invalid_unchanged`	1,733	965	`refresh`
Random	`invalid_changed`	1,840	1,059	`rerank`
	`invalid_unchanged`	1,761	978	`refresh`
Existing SFT	`invalid_changed`	3,840	2,400	`rerank`
	`invalid_unchanged`	3,840	2,400	`refresh`
	`valid_changed`	3,840	3,840	`rerank`
	`valid_unchanged`	3,840	3,840	`retain`