

SCOPE: Stratum-Coordinated Optimization Pipeline for Efficient Agent Context

Pushpendu Ghosh

Amazon
gpuspendu@amazon.com

Aatir Zaki

Amazon
aatzaki@amazon.com

Rudraraju Sacin Varma

Amazon
rudrarav@amazon.com

Stanley Kanagaraj

Amazon
kstanly@amazon.com

Anish Nediyanath

Amazon
anishned@amazon.com

Abstract

LLM agents routinely spend orders of magnitude more tokens on context than on generation, re-sending interaction history, tool schemas, and system prompts at every turn. We introduce **SCOPE**, a unified framework for *context compilation* that compresses these three strata jointly under a fixed token budget while preserving agent behaviour. SCOPE combines (i) a learned *minimal sufficient context* reranker for memory, trained via leave-one-out supervision, (ii) a plan-conditioned tool compressor that speculatively loads likely tools while retaining lightweight signatures for the rest, and (iii) a constraint-aware prompt compressor that preserves instructions. A lightweight planner coordinates these components via a contextual bandit, allocating budget across memory, tools, and prompt to maximize fidelity to full-context execution. In offline replay of 3142 production chat trajectories, SCOPE achieves **96.8%** decision fidelity at **4.46×** compression (60K tokens/turn), matching full-context task success within 0.9% while using only 22% of the tokens. Across operating regimes, SCOPE Pareto-dominates strong single-stratum baselines and naive compositions, with gains exceeding **11%** at matched cost. Our results show that optimizing context in isolation is insufficient: coordinated, budget-aware compression is key to scaling LLM agents without sacrificing reliability.

1 Introduction

LLM agents are moving from single-turn assistants to long-horizon systems that run over hundreds of steps. Here the bottleneck is no longer generation but *context*: outputs stay short, yet each turn reprocesses an input that can exceed 10^5 tokens. In real deployment traces, over 80% of these tokens are never referenced by the agent’s next decision – but pruning them naïvely degrades behaviour sharply and unpredictably, including lost-in-the-middle effects (Liu et al., 2023). The difficulty is structural:

agent context spans three qualitatively different strata – interaction *memory*, a tool *registry*, and a system *prompt* – each with distinct roles, redundancy patterns, and failure modes under compression.

Existing work treats these strata in isolation. Memory methods optimise proxy signals such as recency, similarity, or predicted future-use (Packer et al., 2024; Hu et al., 2024; Zhang et al., 2026); tool-selection methods predict likely next tools but offer no fallback on misprediction (Jia and Li, 2025; Franko, 2025); prompt compressors prune tokens with no awareness of memory or tools (Pan et al., 2024; Xu et al., 2025). No system coordinates all three under a shared token budget, and no memory method supervises what actually matters: preserving the agent’s decisions. The result at deployment is a jagged Pareto frontier where per-stratum tuning either degrades fidelity non-linearly or leaves 3–5× of cost on the table.

We argue context compression should instead be a *joint, budget-constrained, output-preservation* problem – a shift from heuristic pruning to *context compilation*. Central to this view is the *Minimal Sufficient Context* (MSC), $M_t^* = \arg \min_{S \subseteq M_t} |S|$ s.t. $A(S, q_t) = A(M_t, q_t)$: the smallest history that preserves the agent’s output. We instantiate it in **SCOPE**, which factors the compiler into three stratum-specific modules – memory, tools, and prompt – coordinated by a lightweight planner that allocates the budget $B = B_M + B_T + B_P$ online.

Our contributions are: (C1) a unified formulation of agent context compression as joint, budget-constrained output preservation; (C2) the MSC principle, with a supervised Leave-One-Out pipeline that learns M_t^* -membership directly from the agent’s own decisions rather than proxy signals; (C3) a minified-fallback schema hierarchy that decouples tool awareness from executability via hook-based bounded rollback; and (C4) a contextual-bandit planner that learns per-stratum thresholds

and the budget split from offline trajectory sweeps. In offline replay of 3,142 long-horizon production trajectories, SCOPE achieves $4.46\times$ context reduction at 96.8% decision fidelity (3.2% below full context), outperforming per-stratum baselines and naive compositions by 7–21% fidelity at matched budgets, with ~ 600 ms compile overhead and no extra generative-LLM calls in the common case.

2 Related Work

Memory management for long-horizon agents.

The dominant line of work treats memory as a hierarchical store with heuristic read/write policies, paging between tiers (Packer et al., 2024), chunking history by subgoals (Hu et al., 2024), or curating reflection trees (Nguyen et al., 2026). A second line replaces heuristics with learned policies that jointly train memory and reasoning (Zhou et al., 2025), expose memory operations as RL actions (Yu et al., 2026), or add admission control over future-utility (Zhang et al., 2026; Jiang et al., 2026). Across both lines, supervision targets a *proxy* signal – recency, similarity, or predicted reuse – rather than output preservation. SCOPE-Mem is the first method to directly supervise M_t^* -membership via Leave-One-Out labels on the agent’s own decisions, yielding a supervised, reproducible alternative to RL-driven curation.

Tool selection and schema exposure. Scaling tool registries beyond a handful of APIs forces sparsification. Embedding retrievers select top- k schemas by similarity (Qin et al., 2023; Patil et al., 2023; Sengupta et al., 2026); AutoTool (Jia and Li, 2025) exploits sequential tool-usage inertia via a transition graph; ITR (Franko, 2025) jointly retrieves tools and instruction fragments; a complementary thread finetunes the agent for large toolspaces directly (Gupta et al., 2026). All share a binary choice: a tool schema is either fully loaded or entirely omitted, leaving the agent unaware of unloaded tools with no correction path on misprediction. SCOPE-Tools introduces an intermediate *minified signature* tier and a pre-execution hook that guarantees reachability of any tool within one retry – decoupling awareness from executability rather than collapsing them.

Prompt and token-level compression. Prompt compression has matured from perplexity-based pruning (Li et al., 2024) to distilled token classifiers (LLMLingua-2 (Pan et al., 2024)), attribution-

based methods (ProCut (Xu et al., 2025)), and architecture-aware approaches (Fei et al., 2025). These methods deliver strong per-prompt compression in isolation but treat the prompt as a closed string, ignoring cross-stratum dependencies – e.g., dropping a tool-name reference in P while T still advertises that tool. SCOPE-Prompt uses LLMLingua-2 as its backbone but imposes protected-clause constraints tied to the tool module’s outputs, preserving these dependencies.

Joint context optimisation. Hierarchical multi-agent frameworks (Liu et al., 2025; Kawabe and Takano, 2026) decompose tasks across specialised agents but do not compress the three strata of any single agent’s input under a shared budget; attention-pattern studies (Liu et al., 2023) motivate compression globally without an algorithmic recipe. To our knowledge, SCOPE is the first system to (i) formalise agent context compression as a joint, budget-constrained output-preservation problem over (M_t, T, P) , (ii) supervise memory on M_t^* -membership rather than proxies, and (iii) coordinate per-stratum thresholds and the budget split through a single learned planner.

3 Methodology

3.1 Preliminaries and Problem Formulation

At turn t , an agent consumes a typed context $C_t = (M_t, T, P)$, where $M_t = \{m_1, \dots, m_t\}$ denotes interaction memory, $T = \{\tau_1, \dots, \tau_n\}$ the tool registry (each schema τ_j has size s_j tokens), and P the system prompt. Since outputs are short (< 200 tokens), inference cost is dominated by the input, $\text{Cost}(C_t) \approx \alpha|C_t|$. In long-horizon settings ($t > 100$, $n > 50$), $|C_t|$ often exceeds 10^5 tokens.

Intuitively, we seek the smallest context that preserves the agent’s behaviour. We formalise this as a *context compilation* problem: a function $\phi : (C_t, B) \mapsto \tilde{C}_t$ that produces a compressed context under a token budget B ,

$$\begin{aligned} \max_{\phi} \mathbb{E}_{(C_t, q_t) \sim \mathcal{D}} \left[\mathcal{U}(A(\tilde{C}_t, q_t), A(C_t, q_t)) \right], \\ \text{s.t. } |\tilde{C}_t| \leq B, \end{aligned} \quad (1)$$

where $A(\cdot)$ denotes agent execution and $\mathcal{U}$ measures fidelity to full-context behaviour (instantiated as the decision-fidelity metric $\mathcal{F}$ used in evaluation, Sec. 4; exact match for tool calls and token-level F_1 for text). The global budget decomposes per-stratum as $B = B_M + B_T + B_P$, with the split itself learned jointly with compression parameters by

the planner (Sec. 3.5). While the global objective targets end-to-end decision fidelity, individual modules are trained on tractable surrogates (leave-one-out importance for memory, transition likelihoods for tools, pretrained token-retention for prompt); the planner is the single component trained directly on $\mathcal{U}$ and is responsible for aligning module-level decisions with the end objective.

3.2 SCOPE-Mem: Minimal Sufficient Context via Learned Reranking

MSC principle. Given query q_t and full-context answer $a_t = A(M_t, q_t)$, the *minimal sufficient context* is

$$M_t^* = \arg \min_{S \subseteq M_t} |S| \quad \text{s.t.} \quad A(S, q_t) = a_t. \quad (2)$$

Exact computation is combinatorial; we approximate membership with a learned scorer $f_\theta(q_t, m_i) \in [0, 1]$ estimating $\Pr(m_i \in M_t^*)$. This approximation treats message importance as conditionally independent given q_t , ignoring higher-order interactions (e.g., multi-hop co-dependence between messages); we find this assumption empirically benign in our traces because group-level segmentation (below) keeps co-dependent messages within the same candidate block.

Leave-One-Out supervision. For each m_i , we define

$$\Delta_i = d(A(M_t, q_t), A(M_t \setminus \{m_i\}, q_t)), \quad (3)$$

$$y_i = \mathbb{I}[\Delta_i > \epsilon],$$

where $d(\cdot)$ measures output deviation (exact match for tool calls and token- F_1 for text). To reduce cost, we restrict LOO to a similarity-ranked subset of candidates (top- K , proportional to context size and capped), producing importance labels that agree with exhaustive LOO with 0.98 F_1 .

Model. f_θ is a cross-encoder (DeBERTa-v3-base) trained with binary cross-entropy. Inference is fully batched (~ 200 ms/turn, no LLM calls).

Candidate selection. We segment M_t into interaction groups $\{G_j\}$ aligned with user turns, where each group corresponds to a user-initiated episode (including assistant reasoning and tool interactions until the next user input). We retrieve the top- K groups via embedding similarity and score only within them, reducing complexity from $\mathcal{O}(t)$ to $\mathcal{O}(|\tilde{C}_t|)$ with negligible recall loss.

Selection under budget. Messages are tiered using thresholds (τ_h, τ_l) : **L0** (full), **L1** (cached summary), **L2** (dropped). We select $\tilde{M}_t$ via

$$\tilde{M}_t = \arg \max_{S \subseteq M_t} \sum_{m_i \in S} f_\theta(q_t, m_i) \quad (4)$$

$$\text{s.t.} \quad \sum_{m_i \in S} |m_i| \leq B_M.$$

This objective is a 0/1 knapsack; solving it exactly at every turn is infeasible within our compile budget, so we approximate it greedily, admitting candidates in decreasing order of value density $f_\theta(q_t, m_i)/|m_i|$ until B_M is exhausted. On turns small enough to brute-force the exact optimum, the greedy solution attains about 96% of the optimal objective value and recovers the optimal set on about 84% of turns, with a downstream fidelity difference below 0.5 $\mathcal{F}$.

Safety floors. We always retain the last $r=3$ interaction groups, independent of f_θ .

3.3 SCOPE-Tools: Plan-Conditioned Tool Compression

Loading all n tool specifications (schema and documentation) incurs a cost of $\sum_j s_j$ tokens, while only one tool is typically invoked per turn. SCOPE-Tools submodule reduces this overhead by exploiting *tool-usage inertia* and selectively exposing tools based on predicted usage.

Plan-conditioned transition graph. From historical trajectories, we estimate a weighted directed graph $\mathcal{G}_T = (T, E, W)$ with transition probabilities

$$w_{ij}(z_t) = \Pr(\tau_j \mid \tau_i, z_t), \quad (5)$$

where τ_i is the previously invoked tool and z_t is a low-dimensional plan state. Concretely, $z_t = [\text{ENC}(q_t); \hat{y}_{\text{sub}}(q_t, \tau_{t-1})]$, where ENC is a frozen sentence encoder (MiniLM, 384-dim) and $\hat{y}_{\text{sub}}$ is a lightweight subtask classifier (single-layer MLP over $\text{ENC}(q_t) \oplus \tau_{t-1}$) producing a one-hot over eight workflow subtasks annotated from training trajectories. Conditioning on z_t captures intent-dependent transitions between tools.

Speculative top- k loading. At runtime, we load full specifications only for a small set of predicted tools $\hat{T}_k$, selected by cumulative transition probability (typically $k \in [3, 5]$, covering > 0.95 probability mass).

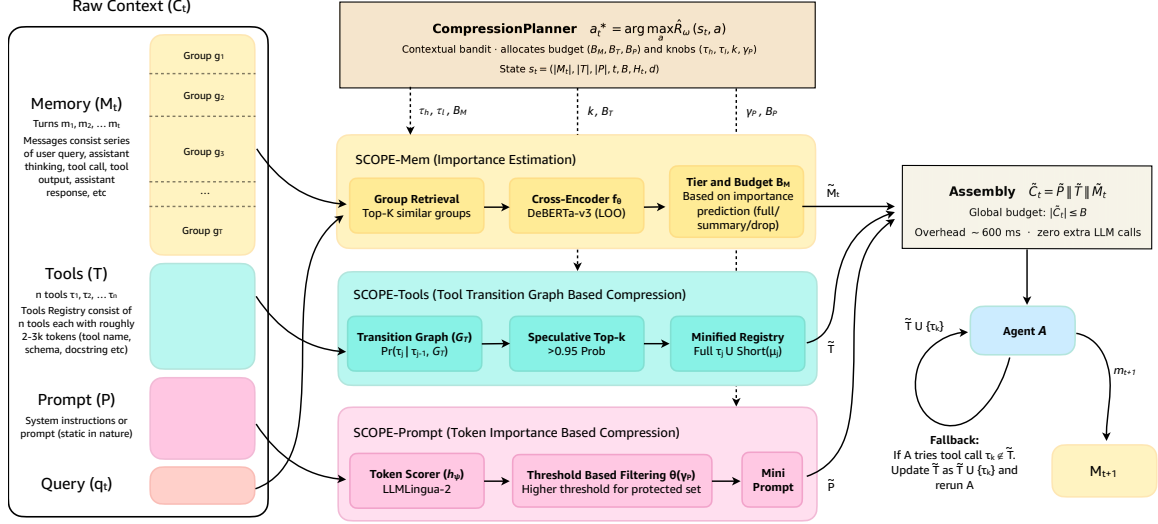


Figure 1: Block Diagram of SCOPE: Three parallel compression paths converging into a planner-controlled assembly

Minified tool representations. For tools $\tau_j \notin \hat{T}_k$, we include a lightweight signature μ_j consisting of the tool name, a short description, and parameter names (~ 30 tokens vs. $\sim 2K$ full specs). The resulting registry is

$$\tilde{T} = \bigcup_{\tau_j \in \hat{T}_k} \tau_j \cup \bigcup_{\tau_j \notin \hat{T}_k} \mu_j. \quad (6)$$

This ensures that all tools remain visible to the agent while significantly reducing token usage.

Fallback on cache miss. If the agent attempts to call a tool outside $\hat{T}_k$, we intercept the call, inject the full specification of the requested tool, and retry once. This adds overhead only on mispredictions and ensures that all tools remain accessible.

3.4 SCOPE-Prompt: Budget-Aware Instruction Compression

System prompts are static within a session, making them suitable for token-level compression. We use a pretrained compressor (LLMLingua-2 (Pan et al., 2024), a distilled xlm-roberta-base token classifier rather than a generative LLM) with constraint-aware pruning. Let $h_\psi(p_i) \in [0, 1]$ denote the keep probability of token p_i . The compressed prompt is

$$\tilde{P} = \{p_i \in P : h_\psi(p_i) > \tau_P(\gamma_P) \vee p_i \in \mathcal{P}_{\text{prot}}\}, \quad (7)$$

where $\tau_P(\gamma_P)$ is a threshold on h_ψ chosen to retain approximately a γ_P fraction of tokens.

The protected set $\mathcal{P}_{\text{prot}}$ includes (i) imperative clauses (e.g., *must*, *never*, *always*), (ii) tool-name

references and argument-structure cues (to remain consistent with the mixed full/minified registry $\tilde{T}$), and (iii) the role-definition block. This preserves critical instructions while allowing aggressive compression. Since P is static within a session, $\tilde{P}$ is computed once and cached, incurring no per-turn overhead.

3.5 CompressionPlanner: Joint Budget Allocation

Compression decisions across memory (τ_h, τ_l), tools (k), and prompt (γ_P) interact, as does the budget split (B_M, B_T, B_P). We optimise them jointly via a contextual bandit.

State and action. At turn t , the state is

$$s_t = (|M_t|, |T|, |P|, t, B, H_t, d), \quad (8)$$

where H_t is the entropy of tool-transition probabilities from $\mathcal{G}_T$. An action defines compression and allocation:

$$a_t = (\tau_h, \tau_l, k, \gamma_P, B_M, B_T, B_P), \quad (9)$$

s.t. $B_M + B_T + B_P = B$.

We discretise $\mathcal{A}$ into a small set of feasible configurations.

Training and inference. For each (s_t, a_t) , we compute fidelity to full-context execution:

$$R_t = \mathcal{U}(A(\tilde{C}_t, q_t), A(C_t, q_t)). \quad (10)$$

Using logged trajectories, we evaluate all $a_t \in \mathcal{A}$ offline and fit a reward model $\hat{R}_\omega(s, a)$ (small

MLP). At runtime,

$$a_t^* = \arg \max_{a \in \mathcal{A}} \hat{R}_\omega(s_t, a). \quad (11)$$

3.6 Context Assembly and Runtime

The final context is assembled as $\tilde{C}_t = \tilde{P} \parallel \tilde{T} \parallel \tilde{M}_t$. The planner ensures the global budget constraint:

$$|\tilde{C}_t| = |\tilde{M}_t| + |\tilde{T}| + |\tilde{P}| \leq B. \quad (12)$$

Compilation overhead is ~ 600 ms/turn (memory ~ 200 ms, tools ~ 10 ms, prompt ~ 350 ms amortised via session-level caching to effectively zero after the first turn), with no additional generative-LLM calls except rare tool retries. Algorithm 1 (App. A.1) summarises the per-turn compile step.

4 Experimental Setup

Data. We evaluate on de-identified logs from a copilot agent system. From ~ 9 K conversations we retain 3,142 long-horizon trajectories (median 22 turns). Each trajectory exposes a 45-tool registry (92K tokens), user-agent memory (177K), and a system prompt (18K), totalling ~ 287 K tokens per turn. We split 70/10/20% train/val/test at the trajectory level; MSC labels, the tool-transition graph $\mathcal{G}_T$, and the planner $\hat{R}_\omega$ are fit only on train. All supervision signals (LOO labels, transition counts, planner rewards) are derived using the *same* backbone model as evaluation, treating full-context execution as the oracle; this is self-distillation from the test-time model rather than leakage across the trajectory split. Generalisation is probed zero-shot on τ^2 -bench and LongMemEval (Sec. 5).

Baselines. We compare against published per-stratum methods and their naive compositions – no prior work jointly compresses all three strata. *Memory*: (i) **Last- $k\%$** recency window, (ii) **Top- $k\%$ RAG** by query-message similarity, (iii) **HiAgent** (Hu et al., 2024) adapted to our trace format with per-subgoal summarisation, (iv) **Mem0** (Chhikara et al., 2025) fact-extracted memory. *Tools*: (v) **Next- k** speculative top- k without fallback, (vi) **AutoTool** (Jia and Li, 2025) learned sequence-inertia retrieval. *Compositions*: we combine the strongest memory and tool methods as a practitioner would (Table 1).

Metrics. All metrics are computed against the full-context trajectory (oracle). Let π^{ref} and π^{cmp} denote reference and candidate trajectories. *Decision fidelity* $\mathcal{F} \in [0, 100]$ is the composite

tool-DAG + answer-equivalence score (Alg. 7, App. A.7): we strip reasoning traces, canonicalise tool parameters, build data-dependency DAGs to allow order-invariance for independent read-only calls, score structural agreement via normalised graph-edit distance, and judge final-answer equivalence with a calibrated LLM ($\kappa=0.86$ against human). *Task success rate* (TSR) is trajectory-level goal completion: since our agent can retry failed tool calls and re-issue evicted lookups, TSR drops only on *unrecoverable* corruption (wrong entity, wrong final action). *Tool- F_1* is set-level agreement on invoked tools. *Cost* is measured as mean input tokens per turn $\mathcal{T}$ and total tokens per trajectory $\mathcal{T}_{\text{tot}}$, the latter capturing extra turns induced by memory re-calls and tool-miss retries. Compression ratio $\rho = \mathcal{T}_{\text{tot}}^{\text{full}} / \mathcal{T}_{\text{tot}}^{\text{cmp}}$.

Protocol. One frontier-class backbone with greedy decoding is used throughout to isolate compression effects from sampling noise. Each method runs the full trajectory end-to-end under its compression regime; oracle outputs are used only as fidelity targets. All comparisons are *offline replays* of logged trajectories: the trajectories are drawn from a deployed system, but SCOPE itself is not evaluated in an online A/B deployment, so the reported cost and task-success figures should not be read as measured user-facing deployment impact. Baselines are budget-tuned on the validation split to maximise $\mathcal{F}$ subject to a target $\mathcal{T}$; SCOPE sweeps $B \in \{30, 45, 60, 90\}$ K. Results are means over test trajectories. Ablations (-PLANNER, -MEM, -TOOLS, -PROMPT) and the internal MSC-vs-proxy comparison are reported in Sec. 5.

5 Results and Discussion

Cost-fidelity frontier. SCOPE dominates the frontier across all budgets (Table 1, Fig. 2). At $B=60$ K it matches full-context task success within 0.9% using only 22% of tokens, and reaches the fidelity of strong baselines at 2–3 $\times$ fewer tokens – gains driven by improved context efficiency, not allocation alone. The frontier shifts regime around $B \approx 60$ K: below it, fidelity rises steeply as compression removes critical context; above it, returns flatten to +1–2pt, indicating that task-relevant information concentrates in a small fraction of the full context.

Coordination, not composition. Each module improves decision fidelity by 9–10% over its base-

Table 1: Offline replay on the held-out production test set ($N=3,142$ long-horizon trajectories). Methods compress **Memory**, **Tools**, and/or **Prompt**. $\mathcal{F}$: decision fidelity; TSR: task success rate; Tool- F_1 : tool-call agreement; $\mathcal{T}$: tokens/turn (K); $\mathcal{T}_{\text{tot}}$: total tokens/trajectory (M); ρ : compression ratio. **SCOPE** jointly optimizes all three strata and achieves the best cost–fidelity trade-off.

Method	M	T	P	$\mathcal{F}$	TSR	Tool- F_1	$\mathcal{T}$ (K)	$\mathcal{T}_{\text{tot}}$ (M)	ρ
Full-Context (oracle)	–	–	–	100.0	91.7 ± 1.3	1.000	287.4	5.74	$1.0\times$
<i>Single-stratum baselines (published methods)</i>									
Last-15% Memory	✓			77.5 ± 2.3	81.2 ± 2.5	0.748 ± 0.030	120.5	3.64	$1.58\times$
Top-20% RAG (memory)	✓			84.2 ± 1.8	86.4 ± 2.1	0.812 ± 0.028	124.2	3.33	$1.73\times$
HiAgent [Modified] (Hu et al., 2024)	✓			87.9 ± 2.0	88.2 ± 1.8	0.809 ± 0.025	165.8	4.08	$1.41\times$
Mem0 (Chhikara et al., 2025)	✓			87.8 ± 1.6	89.6 ± 1.6	0.843 ± 0.022	132.4	2.92	$1.97\times$
AutoTool (Jia and Li, 2025)		✓		89.4 ± 1.6	91.1 ± 1.3	0.908 ± 0.018	205.2	4.51	$1.27\times$
<i>Naive multi-stratum compositions (no joint method exists in prior work)</i>									
Top-20% RAG + Next-5 Tools	✓	✓		84.9 ± 2.0	85.4 ± 2.3	0.860 ± 0.025	74.8	1.95	$2.94\times$
Top-20% RAG + AutoTool	✓	✓		85.6 ± 1.8	87.9 ± 1.8	0.871 ± 0.020	63.6	1.60	$3.59\times$
<i>SCOPE modules (ours), ablated in isolation without budget planner</i>									
SCOPE-Mem only	✓			97.2 ± 1.1	91.2 ± 1.3	0.962 ± 0.015	148.2	3.11	$1.85\times$
SCOPE-Tools only		✓		98.6 ± 0.8	91.5 ± 1.0	0.988 ± 0.010	206.4	4.25	$1.35\times$
SCOPE-Prompt only			✓	97.8 ± 1.1	91.0 ± 1.3	0.974 ± 0.013	279.6	5.59	$1.03\times$
<i>SCOPE (ours, coordinated joint compression)</i>									
SCOPE ($B=30\text{K}$, tight)	✓	✓	✓	84.8 ± 2.3	84.3 ± 2.5	0.856 ± 0.028	31.4	0.88	$6.54\times$
SCOPE ($B=45\text{K}$)	✓	✓	✓	93.5 ± 1.6	89.4 ± 1.6	0.932 ± 0.018	44.1	1.05	$5.48\times$
SCOPE ($B=60\text{K}$)	✓	✓	✓	96.8 ± 1.1	90.8 ± 1.3	0.965 ± 0.013	58.7	1.29	$4.46\times$
SCOPE ($B=90\text{K}$, loose)	✓	✓	✓	98.2 ± 0.8	91.4 ± 1.0	0.978 ± 0.013	86.8	1.81	$3.18\times$

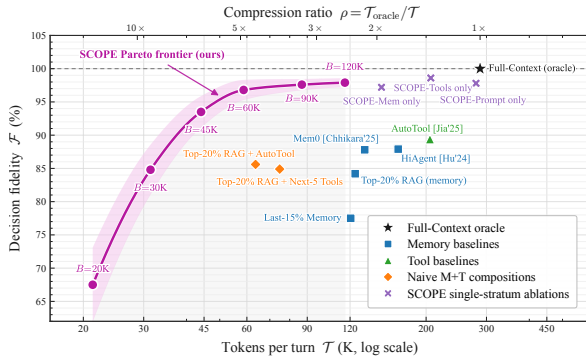


Figure 2: **Cost–fidelity Pareto frontier.** Decision fidelity ($\mathcal{F}$, y-axis) vs. tokens per turn ($\mathcal{T}$, x-axis, log scale). SCOPE (pink) Pareto-dominates all baselines across budgets. Top axis shows compression ratio ρ . At $B=60\text{K}$, SCOPE achieves 96.8% fidelity at $4.46\times$ compression.

line at similar or lower cost, confirming that decision-preserving supervision (MSC) beats proxy objectives like recency or similarity. These gains do not compose naively: stacking strong memory and tool baselines yields only marginal improvement as independent errors compound, whereas SCOPE’s joint budget allocation adds +11% at tighter compression. The planner is what supplies this coordination: replacing it with the best validation-tuned fixed budget split lowers $\mathcal{F}$ from 96.8 to 93.9 at $B=60\text{K}$, and a uniform $B/3$ split to 89.2, with both gaps widening at tighter budgets where allocation matters more.

Fidelity vs. task success. The two metrics are not interchangeable, and the gap between them is by construction: $\mathcal{F}$ penalises *any* divergence from the full-context trajectory, whereas TSR only asks whether the goal was reached. Most turns that $\mathcal{F}$ counts as deviations self-recover later in the trajectory – a lookup re-issued after eviction, or an alternative tool sequence reaching the same outcome – so the 3.2-point $\mathcal{F}$ drop at $B=60\text{K}$ leaves only the 0.9-point TSR gap ($91.7 \rightarrow 90.8$) as net unrecoverable. The distance between the two therefore indicates a conservative operating point rather than an inconsistency. Irrecoverable failures concentrate under aggressive prompt compression, and residual errors cluster around fine-grained numeric recall and unseen tool transitions, both degrading gracefully.

Zero-shot generalization. We test transfer beyond the production domain by evaluating SCOPE *zero-shot* – no component re-trained – on τ^2 -bench (Barres et al., 2025) (tool-agentic, dual-control) and LongMemEval (Wu et al., 2025) (long-horizon conversational memory), transferring f_θ , $\mathcal{G}_T$, and $\hat{R}_\omega$ as-is against the strongest applicable baseline at matched budget (Table 2).

Without adaptation, SCOPE stays within 1.2% (τ^2 -bench) and 1.7% (LongMemEval) of full-context success at $>4\times$ compression, ahead of the best baseline by 2–5% while compressing $\sim 2\times$

Table 2: Zero-shot transfer (no re-training). Success is the benchmark-native metric (τ^2 pass rate; LongMemEval QA accuracy); ρ : compression ratio. SCOPE retains most of its in-domain advantage despite no adaptation.

Benchmark	Method	Success	ρ
τ^2 -bench	Full-Context (oracle)	61.4	1.0×
	AutoTool	57.9	1.9×
	SCOPE	60.2	4.1×
LongMemEval	Full-Context (oracle)	88.7	1.0×
	Top-20% RAG	82.4	2.2×
	SCOPE	87.0	4.3×

more aggressively. The residual gap to full context tracks distribution shift in tool inventories and dialogue style, largest on LongMemEval where unseen long-range coreference stresses the reranker. SCOPE thus captures domain-general structure rather than overfitting a single registry, subject to the single-backbone caveats of Sec. 6.

6 Conclusion

We presented **SCOPE**, a unified framework for context compilation in long-horizon LLM agents that jointly compresses memory, tools, and prompts under a shared token budget. By directly optimizing for output preservation and coordinating decisions via a lightweight planner, SCOPE consistently outperforms strong single-stratum methods and naive compositions, achieving up to $4.46\times$ compression at near-parity task success. Our results demonstrate that context efficiency is fundamentally a *systems problem*: substantial gains arise not from better local heuristics, but from global, budget-aware coordination across context strata.

Limitations

Our evaluation is conducted on a conversational agent dataset with domain-specific prompts, a 45-tool registry, and long multi-step tool-call chains. Although SCOPE is designed as a modular framework, deployment to substantially different agent ecosystems (e.g., code-generation or open-domain web agents) will require re-fitting the learned components, particularly the tool-transition graph and planner, and the optimal operating point may vary across domains. The learned reranker, planner, and tool-transition graph are trained from historical trajectories and full-context executions of a single backbone model. Changing the backbone therefore requires re-fitting these components – in our experi-

ence the reranker threshold and the planner reward in particular – and we report no cross-backbone results here; multi-backbone evaluation remains future work. Our evaluation is also an offline replay of logged trajectories rather than an online deployment study, so it does not measure user-facing impact. SCOPE also introduces approximately 600 ms of compile-time overhead, which can be reduced by replacing the cross-encoder reranker with similarity-based retrieval at the cost of lower fidelity.

Broader Impact

SCOPE cuts the context reprocessed each turn from roughly 287K to 59K tokens (79% fewer per turn; $4.46\times$ over full trajectories once compression-induced retries are counted) at 96.8% decision fidelity, within 0.9% of full-context task success. Since agent inference cost scales with input size ($\text{Cost}(C_t) \approx \alpha|C_t|$, Sec. 3.1), this yields proportional savings in serving cost, energy, and carbon with no model change: for an agent serving 1M turns/day, SCOPE avoids reprocessing $\sim 2.3 \times 10^{11}$ input tokens daily ($\sim 8.3 \times 10^{13}$ /year), on the order of \$83M/year in compute avoided per million daily turns at \$1 per million tokens, scaling linearly with traffic.¹ Because SCOPE compresses only the input context and leaves generation length unchanged, we claim cost and throughput benefits rather than latency improvements. The residual 3.2% fidelity gap means a small fraction of turns may diverge from full-context behaviour, so high-stakes deployments should tune the operating point conservatively and retain the minified-fallback hierarchy that keeps every tool reachable within one retry.

References

- Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2 -bench: Evaluating conversational agents in a dual-control environment. *Preprint*, arXiv:2506.07982.
- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *Preprint*, arXiv:2504.19413.

¹Illustrative arithmetic from the measured 79% per-turn token reduction, not savings realised in an online deployment; absolute figures depend on traffic and pricing, but the $\sim 4.5\times$ proportional saving is not.

- Weizhi Fei, Xueyan Niu, Guoqing Xie, Yingqing Liu, Bo Bai, and Wei Han. 2025. [Efficient prompt compression with evaluator heads for long-context transformer inference](#). *Preprint*, arXiv:2501.12959.
- Uria Franko. 2025. [Dynamic system instructions and tool exposure for efficient agentic llms](#). *Preprint*, arXiv:2602.17046.
- Karan Gupta, Pranav Vajreshwari, Yash Pandya, Raghav Magazine, Akshay Nambi, and Ahmed Awadallah. 2026. [Scaling agentic capabilities, not context: Efficient reinforcement finetuning for large toolspaces](#). *Preprint*, arXiv:2603.06713.
- Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2024. [Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model](#). *Preprint*, arXiv:2408.09559.
- Jingyi Jia and Qinbin Li. 2025. [Autotool: Efficient tool selection for large language model agents](#). *Preprint*, arXiv:2511.14650.
- Dongming Jiang, Yi Li, Songtao Wei, Jinxin Yang, Ayushi Kishore, Alysa Zhao, Dingyi Kang, Xu Hu, Feng Chen, Qiannan Li, and Bingzhe Li. 2026. [Anatomy of agentic memory: Taxonomy and empirical analysis of evaluation and system limitations](#). *Preprint*, arXiv:2602.19320.
- Tomoya Kawabe and Rin Takano. 2026. [Hierarchical llm-based multi-agent framework with prompt optimization for multi-robot task planning](#). *Preprint*, arXiv:2602.21670.
- Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. 2024. [Prompt compression for large language models: A survey](#). *Preprint*, arXiv:2410.12388.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. [Lost in the middle: How language models use long contexts](#). *Preprint*, arXiv:2307.03172.
- Pei Liu, Xin Liu, Ruoyu Yao, Junming Liu, Siyuan Meng, Ding Wang, and Jun Ma. 2025. [Hm-rag: Hierarchical multi-agent multimodal retrieval augmented generation](#). *Preprint*, arXiv:2504.12330.
- Andy Nguyen, Danh Doan, Hoang Pham, Bao Ha, Dat Pham, Linh Nguyen, Hieu Nguyen, Thien Nguyen, Cuong Do, Phat Nguyen, and Toan Nguyen. 2026. [Byterover: Agent-native memory through llm-curated hierarchical context](#). *Preprint*, arXiv:2604.01599.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. [Memgpt: Towards llms as operating systems](#). *Preprint*, arXiv:2310.08560.
- Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. [Llmllingua-2: Data distillation for efficient and faithful task-agnostic prompt compression](#). *Preprint*, arXiv:2403.12968.
- Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. [Gorilla: Large language model connected with massive apis](#). *Preprint*, arXiv:2305.15334.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. [Toolllm: Facilitating large language models to master 16000+ real-world apis](#). *Preprint*, arXiv:2307.16789.
- Saptarshi Sengupta, Zhengyu Zhou, Jun Araki, Xingbo Wang, Bingqing Wang, Suhang Wang, and Zhe Feng. 2026. [Tooldreamer: Instilling llm reasoning into tool retrievers](#). *Preprint*, arXiv:2510.19791.
- Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. [Longmemeval: Benchmarking chat assistants on long-term interactive memory](#). *Preprint*, arXiv:2410.10813.
- Zhentao Xu, Fengyi Li, Albert Chen, and Xiaofeng Wang. 2025. [Procut: Llm prompt compression via attribution estimation](#). *Preprint*, arXiv:2508.02053.
- Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. 2026. [Agentic memory: Learning unified long-term and short-term memory management for large language model agents](#). *Preprint*, arXiv:2601.01885.
- Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. 2026. [Memory as action: Autonomous context curation for long-horizon agentic tasks](#). *Preprint*, arXiv:2510.12635.
- Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025. [Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents](#). *Preprint*, arXiv:2506.15841.

Algorithm 1 SCOPE context compilation at turn t

Require: Context $C_t = (M_t, T, P)$, query q_t , budget B , models $(f_\theta, \mathcal{G}_T, h_\psi, \hat{R}_\omega)$

Ensure: Compressed context $\tilde{C}_t$

```

1:  $(\tau_h, \tau_l, k, \gamma_P, B_M, B_T, B_P) \leftarrow \arg \max_a \hat{R}_\omega(s_t, a) \triangleright$ 
   planner
2:  $\{G_j\} \leftarrow \text{SEGMENT}(M_t)$ 
3: Retrieve top- $K$  groups  $\mathcal{G}^*$  by  $\text{sim}(E(q_t), E(G_j))$ 
4: Score  $f_\theta(q_t, m_i)$  for  $m_i \in \bigcup \mathcal{G}^*$ 
5: Tier by  $(\tau_h, \tau_l)$  and greedily select by value density under
    $B_M$ 
6: Add safety floors (last  $r$  groups):  $\tilde{M}_t$ 
7: Compute  $z_t$  and  $\hat{T}_k \leftarrow \text{TOPK}(\text{Pr}(\cdot \mid \tau_{t-1}, z_t))$ 
8:  $\hat{T} \leftarrow \{\tau_j\}_{\tau_j \in \hat{T}_k} \cup \{\mu_j\}_{\tau_j \notin \hat{T}_k}$ 
9:  $\tilde{P} \leftarrow \{p_i : h_\psi(p_i) > \tau_P(\gamma_P) \vee p_i \in \mathcal{P}_{\text{prot}}\}$ 
10: if  $\tau_j \notin \hat{T}_k$  then
11:   Inject full  $\tau_j$  and retry once
12: end if
13: return  $\tilde{P} \parallel \hat{T} \parallel \tilde{M}_t$ 

```

A Appendix

This appendix expands the sub-procedures referenced in Section 3. Algorithm 1 captures the end-to-end compile step; the remaining subsections detail the SCOPE-Mem training and inference pipeline, the SCOPE-Tools prediction routine, and the CompressionPlanner action-selection procedure.

A.1 End-to-End Compile Step

Algorithm 1 summarises the per-turn compile invoked at runtime. The planner (line 1) selects per-stratum thresholds and the budget split; lines 2–4 instantiate SCOPE-Mem, line 5 instantiates SCOPE-Tools, and line 6 instantiates SCOPE-Prompt. The pre-execution hook below the rule guarantees recoverability of any tool not in the speculative top- k set.

A.2 SCOPE-Mem: LOO Label Construction

Algorithm 2 details offline label generation. For each trajectory turn we pre-filter candidates with similarity-based top- K retrieval (default $K=50$) before issuing LOO agent calls, yielding a $5\times$ cost reduction with 0.98 label-quality F_1 versus exhaustive LOO. Labels are produced once, with greedy decoding, and cached.

A.3 SCOPE-Mem: Cross-Encoder Training

Algorithm 3 reports the supervised fine-tuning loop for the MSC reranker f_θ (DeBERTa-v3-base, $\sim 60\text{M}$ params), trained with AdamW at $\text{lr}=2\text{e-}5$, batch size 32, 3 epochs on the labels from Algorithm 2.

Algorithm 2 LOO-based label construction for SCOPE-Mem

Require: Trajectory dataset $\mathcal{D}$, agent $A(\cdot)$, tolerance ϵ , cap K

Ensure: Training set $\mathcal{D}_{\text{tr}} = \{(q_t, m_i, y_i)\}$

```

1:  $\mathcal{D}_{\text{tr}} \leftarrow \emptyset$ 
2: for each trajectory  $\tau \in \mathcal{D}$  and turn  $t \in \tau$  do
3:    $a_t \leftarrow A(M_t, q_t)$   $\triangleright$  cached from original log
4:    $\mathcal{C}_t \leftarrow \text{TopK}_i(\text{sim}(q_t, m_i), K)$ 
5:   for each  $m_i \in \mathcal{C}_t$  do
6:      $a_{-i} \leftarrow A(M_t \setminus \{m_i\}, q_t)$ 
7:      $\Delta_i \leftarrow d(a_t, a_{-i})$ 
8:      $y_i \leftarrow \mathbb{I}[\Delta_i > \epsilon]$ 
9:      $\mathcal{D}_{\text{tr}} \leftarrow \mathcal{D}_{\text{tr}} \cup \{(q_t, m_i, y_i)\}$ 
10:  end for
11: end for
12: return  $\mathcal{D}_{\text{tr}}$ 

```

Algorithm 3 Training the MSC reranker f_θ

Require: Labels $\mathcal{D}_{\text{tr}}$, model f_θ , epochs E

Ensure: Trained parameters θ

```

1: for  $e = 1, \dots, E$  do
2:   for each batch  $\mathcal{B} \subset \mathcal{D}_{\text{tr}}$  do
3:      $I_i \leftarrow f_\theta(q_t, m_i), \forall (q_t, m_i) \in \mathcal{B}$ 
4:      $\mathcal{L} \leftarrow -\sum_i [y_i \log I_i + (1 - y_i) \log(1 - I_i)]$ 
5:      $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$ 
6:   end for
7: end for
8: return  $\theta$ 

```

A.4 SCOPE-Mem: Two-Stage Inference with Safety Floors

Algorithm 4 combines group-level retrieval, cross-encoder reranking, budget-constrained greedy selection, tier assignment, and unconditional safety floors. Per-turn latency is dominated by the batched cross-encoder pass (~ 20 ms on A10G over 50–200 pruned candidates).

A.5 SCOPE-Tools: Top- k Prediction with Hook-Based Rollback

Algorithm A.5 predicts $\hat{T}_k$ by walking the plan-conditioned transition graph $\mathcal{G}_T$, emits minified signatures μ_j for the remainder, and specifies the pre-execution hook that corrects cache misses within one retry.

A.6 CompressionPlanner: Action Selection

Algorithm ?? is the online inference step; offline training is a standard regression fit over (s_t, a_t, R_t) tuples collected by grid-sweeping a_t over historical trajectories, removing the exploration problem entirely.

A.7 Trajectory Fidelity Computation

This section details how the decision fidelity $\mathcal{F}$ reported in Section 5 is computed between a refer-

Algorithm 4 SCOPE-Mem inference at turn t

Require: Memory M_t , query q_t , budget B_M , thresholds (τ_h, τ_l) , reranker f_θ

Ensure: Tiered memory $\tilde{M}_t$

- 1: $\{G_j\} \leftarrow \text{SEGMENT}(M_t)$ $\triangleright$ topic-shift boundaries
- 2: $\mathcal{G}^* \leftarrow \text{TOPK}_j(\text{sim}(E(q_t), E(G_j)))$
- 3: $\mathcal{C}_t \leftarrow \bigcup_{G_j \in \mathcal{G}^*} G_j$
- 4: **for** each $m_i \in \mathcal{C}_t$ **do**
- 5: $I_i \leftarrow f_\theta(q_t, m_i)$
- 6: **end for**
- 7: Sort $\mathcal{C}_t$ by decreasing $I_i/|m_i|$ $\triangleright$ value density
- 8: $\tilde{M}_t \leftarrow \emptyset$, tok $\leftarrow 0$
- 9: **for** each m_i in sorted $\mathcal{C}_t$ **do**
- 10: **if** tok + $|m_i| \leq B_M$ **then**
- 11: $\tilde{M}_t \leftarrow \tilde{M}_t \cup \{m_i\}$
- 12: tok $\leftarrow$ tok + $|m_i|$
- 13: **end if**
- 14: **end for**
- 15: **for** each $m_i \in \tilde{M}_t$ **do**
- 16: **if** $I_i > \tau_h$ **then**
- 17: Assign tier L0
- 18: **else if** $I_i > \tau_l$ **then**
- 19: Assign tier L1
- 20: **else**
- 21: Assign tier L2
- 22: **end if**
- 23: **end for**
- 24: Promote last $r = 3$ interaction groups, q_t , and tool-error messages to L0
- 25: **return** $\tilde{M}_t$

Algorithm 5 SCOPE-Tools compile and rollback

Require: Registry T , previous tool τ_{t-1} , plan state z_t , graph $\mathcal{G}_T, k, p$

Ensure: Registry $\tilde{T}$

- 1: $\mathbf{w} \leftarrow \mathcal{G}_T[\tau_{t-1}, z_t]$
- 2: $\hat{T}_k \leftarrow \text{PREFIX}(\mathbf{w}, p, k)$
- 3: $\tilde{T} \leftarrow \{\tau_j\}_{\tau_j \in \hat{T}_k} \cup \{\mu_j\}_{\tau_j \notin \hat{T}_k}$
- 4: **if** emitted tool $\tau_j \notin \hat{T}_k$ **then**
- 5: Inject full schema
- 6: Retry once
- 7: **end if**
- 8: **return** $\tilde{T}$

ence trajectory π^{ref} (produced under full context) and a compressed trajectory π^{cmp} (produced under $\tilde{C}_t$). A naive sequence-equality check is too strict: agents frequently re-order read-only tool calls, add or omit reasoning (“thinking”) steps, and phrase final answers slightly differently while remaining semantically equivalent. We therefore decompose fidelity into three components – thinking-invariance, tool-call equivalence modulo data dependencies, and answer equivalence – and combine them into a single score.

Trajectory representation. A trajectory π is a sequence of typed events $e_1, e_2, \dots, e_L$ where each e_j is one of: user message u , assistant thinking block θ , tool call $c=(\tau, \mathbf{p})$ with tool name τ and

Algorithm 6 CompressionPlanner action selection

Require: State $s_t = (|M_t|, |T|, |P|, t, B, H_t, d)$, reward model $\hat{R}_\omega$, action grid $\mathcal{A}$

Ensure: Action $a_t^* = (\tau_h, \tau_l, k, \gamma_P, B_M, B_T, B_P)$

- 1: **for** each $a \in \mathcal{A}$ with $B_M + B_T + B_P = B$ **do**
- 2: $r_a \leftarrow \hat{R}_\omega(s_t, a)$
- 3: **end for**
- 4: **return** $\arg \max_a r_a$

canonicalised parameter dict $\mathbf{p}$, tool result r , or final assistant answer a . We write $\text{TC}(\pi)$ for the ordered list of tool calls in π and $\text{ANS}(\pi)$ for its final answer.

Running example. Consider the three trajectories from the main text:

$$\begin{aligned} \pi_1 : U_1 \rightarrow \Theta_1 \rightarrow T_6 \rightarrow R_6 \rightarrow \Theta_2 \\ \rightarrow T_4 \rightarrow R_4 \rightarrow T_7 \rightarrow R_7 \rightarrow \Theta_3 \rightarrow A_1, \\ \pi_2 : U_1 \rightarrow \Theta_4 \rightarrow T_4 \rightarrow R_4 \\ \rightarrow T_6 \rightarrow R_6 \rightarrow T_7 \rightarrow R_7 \rightarrow A_2, \\ \pi_3 : U_1 \rightarrow \Theta_1 \rightarrow T_6 \rightarrow R_6 \\ \rightarrow T_4 \rightarrow R_4 \rightarrow T_7 \rightarrow R_7 \rightarrow \Theta_3 \rightarrow A_1. \end{aligned}$$

Trajectories π_1 and π_3 differ only by the intermediate thinking step Θ_2 and are therefore considered equivalent. Trajectories π_1 and π_2 execute T_6 and T_4 in different orders; if neither tool depends on the other’s output, the executions remain equivalent. Likewise, if A_1 and A_2 are judged semantically equivalent, the trajectories receive a high fidelity score despite surface differences.

Step 1: Thinking-block elision. We strip all thinking events, yielding the *externalised* trajectory $\bar{\pi}$. This reflects that internal reasoning does not affect environment state or user-visible behaviour and that compression methods routinely change reasoning traces without changing decisions.

Step 2: Parameter canonicalisation. For a tool call $c=(\tau, \mathbf{p})$, $\mathbf{p}$ is a JSON-like dict. We canonicalise by: sorting keys; lowercasing and whitespace-trimming string values; normalising numeric and date formats (ISO-8601); and coercing booleans. Two calls agree parametrically iff $\tau_a=\tau_b$ and $\text{CANON}(\mathbf{p}_a)=\text{CANON}(\mathbf{p}_b)$.

Step 3: Data-dependency DAG. Tool-call order is semantically significant only when a later call consumes an earlier call’s output. We build the *data-dependency DAG* $D(\bar{\pi}) = (V, E)$ with $V = \text{TC}(\bar{\pi})$ and $(c_a \rightarrow c_b) \in E$ iff any value in $\mathbf{p}_b$ is traceable to the output r_a of an earlier c_a (by

substring / ID / entity match over the canonicalised JSON). The *tool-call equivalence* between two trajectories is the graph-edit-distance-normalised similarity

$$\text{TCE}(\pi^{\text{ref}}, \pi^{\text{cmp}}) = 1 - \frac{\text{GED}(D(\bar{\pi}^{\text{ref}}), D(\bar{\pi}^{\text{cmp}}))}{|V^{\text{ref}}| + |V^{\text{cmp}}|}, \quad (13)$$

where node matches require parametric agreement (Step 2) and edge matches require agreement on the induced dependency relation. For trajectories whose tool-call DAGs are small (typically ≤ 15 nodes in our traces), GED is tractable; we use the VF2-style approximation with a 1-second timeout and fall back to bag-of-calls F_1 otherwise. This formulation gives $\pi_1 \equiv \pi_3$ exactly (identical DAG), and $\pi_1 \simeq \pi_2$ up to the $T_4 \leftrightarrow T_6$ swap whenever the dependency check reports no edge between them.

Step 4: Answer equivalence. For the final answer, exact match and token- F_1 both penalise benign rephrasings. We use an LLM-as-judge with a 3-way verdict (EQUIVALENT, PARTIAL, DIFFERENT) and log-probability-weighted confidence, collapsed to $\text{AEQ} \in [0, 1]$. The judge is prompted with the user query, the reference answer, and the candidate answer, and is explicitly instructed to ignore stylistic variation, to penalise factual disagreement, and to output a structured verdict plus one-line justification. We validate the judge against 500 human-labelled pairs, reporting $\kappa=0.86$ inter-rater agreement.

Step 5: Composite fidelity. We combine the two components:

$$\mathcal{F}(\pi^{\text{ref}}, \pi^{\text{cmp}}) = \alpha \cdot \text{TCE} + (1 - \alpha) \cdot \text{AEQ}, \quad (14)$$

with default $\alpha=0.6$ reflecting that in an agentic setting correct tool execution is load-bearing for correct answers. We ablate $\alpha \in \{0.4, 0.6, 0.8\}$ and find rankings between methods stable; results in the main paper use $\alpha=0.6$. Trajectory-level fidelity is averaged over turns, and aggregated fidelity is the macro-average over test trajectories.

Dependency detection details. `BUILDDepDAG` scans tool calls in temporal order; for each call c_b , every value in `CANON`($\mathbf{p}_b$) is checked against the canonicalised outputs r_a of preceding calls via (i) exact-ID match on fields tagged as identifiers in the tool schema (deal-id, order-id, sku, etc.), and (ii) substring match with a minimum token length of 4 for free-text fields. We add an edge $c_a \rightarrow c_b$ on

Algorithm 7 Trajectory fidelity computation

Require: Reference trajectory π^{ref} , candidate trajectory π^{cmp} , judge $\mathcal{J}$, weight α
Ensure: Fidelity score $\mathcal{F} \in [0, 1]$
1: $\bar{\pi}^{\text{ref}} \leftarrow \text{STRIPTHINKING}(\pi^{\text{ref}})$
2: $\bar{\pi}^{\text{cmp}} \leftarrow \text{STRIPTHINKING}(\pi^{\text{cmp}})$
3: **for** each $c = (\tau, \mathbf{p}) \in \text{TC}(\bar{\pi}^{\text{ref}}) \cup \text{TC}(\bar{\pi}^{\text{cmp}})$ **do**
4: $\mathbf{p} \leftarrow \text{CANON}(\mathbf{p})$ $\triangleright$ sort keys, normalize types
5: **end for**
6: $D^{\text{ref}} \leftarrow \text{BUILDDepDAG}(\bar{\pi}^{\text{ref}})$
7: $D^{\text{cmp}} \leftarrow \text{BUILDDepDAG}(\bar{\pi}^{\text{cmp}})$
8: $\text{TCE} \leftarrow 1 - \text{NORMGED}(D^{\text{ref}}, D^{\text{cmp}})$ $\triangleright$ bag- F_1 fallback
9: $v \leftarrow \mathcal{J}(\text{ANS}(\pi^{\text{ref}}), \text{ANS}(\pi^{\text{cmp}}), q)$
10: $\text{AEQ} \leftarrow \text{SCORE}(v)$
11: **return** $\alpha \text{TCE} + (1 - \alpha) \text{AEQ}$

any hit. This heuristic is intentionally slightly over-permissive: false-positive edges only hurt the compressed trajectory when it legitimately reordered independent calls, and the resulting small penalty is preferable to under-detecting real dependencies.

Judge calibration. Before using $\mathcal{J}$ in evaluation, we calibrate it on a held-out set of 500 trajectory pairs hand-labelled by two authors (agreement $\kappa=0.86$). The judge achieves $F_1=0.91$ against human consensus on the 3-way verdict; residual errors concentrate in numeric-rounding cases (e.g., prices differing by \$0.01 due to upstream quantisation), which we handle by a post-processing rule that marks numeric-only differences under a relative tolerance as EQUIVALENT. We use greedy decoding and two judge invocations per pair with order-swapped inputs; disagreements are resolved by a third invocation.