RELAY: ARE WE READY FOR LATENT ROUTING FOR MULTI-AGENT SYSTEMS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM)-based multi-agent systems rely on communication and orchestration for collaborative reasoning. Existing methods often exchange reasoning traces in a fixed textual or latent form. Text preserves precise details but can be verbose, whereas latent representations offer semantic richness at the risk of obscuring critical information. Beyond representation choice, complete traces often contain essential information alongside erroneous, irrelevant, or redundant evidence. These limitations motivate segment-wise routing that combines complementary representations and filters low-utility content. Our empirical analysis further shows that three internal trace signals—entropy, maximum probability, and hidden-state norm—carry predictive information about downstream correctness, suggesting their utility for adaptive routing among TEXT, LATENT, and SKIP. We introduce RELAY, a two-level routing framework for effective multi-agent collaboration. For intra-agent routing, RELAY employs a Signal-Conditioned Utility Router (SCUR) to make segment-wise transmission decisions. SCUR aggregates internal trace signals to estimate the downstream utility of each communication choice. The router is trained on counterfactual receiver outcomes, and its predictions guide the construction of mixed-format messages. For inter-agent routing, an Adaptive Representation Orchestrator (ARO) conditions on the routed message and task state to select the next compatible agent or terminate the interaction. Experiments across reasoning and long-horizon decision-making tasks demonstrate that RELAY achieves the strongest overall performance–efficiency trade-off, outperforming baselines while substantially reducing communication cost. The code is available at: `anonymous/ReLay`.

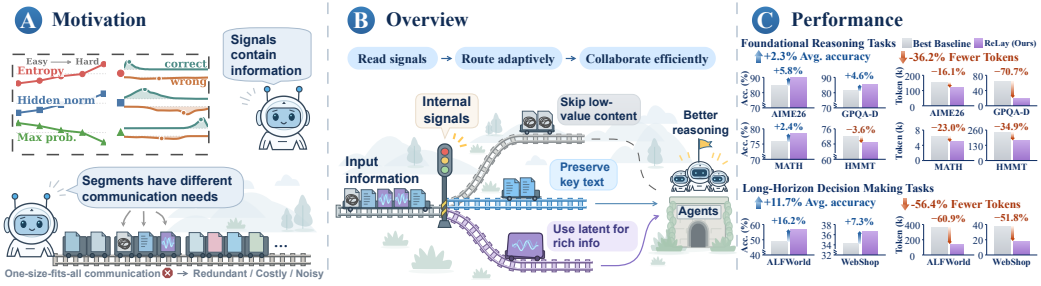


Figure 1: (A) Internal signals reveal useful routing cues, while reasoning segments have heterogeneous communication needs. (B) RELAY adaptively selects text, latent, or skip communication for each segment for agent collaboration. (C) RELAY outperforms baselines across diverse benchmarks.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems (MAS) address complex problems by distributing reasoning across multiple agents (Tran et al., 2025; Sun et al., 2024; Guo et al., 2024; Lin et al., 2025; Yan et al., 2025; Jin et al., 2025). Through role specialization and the exchange of intermediate results, agents can decompose tasks, critique one another, and iteratively refine solutions (Hong et al., 2024; Chen et al., 2024a;b; Wu et al., 2023). The design of such systems involves both coordinating agents and determining what information to exchange and how to represent it.

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.

Existing routing methods mainly determine agent selection, collaboration structure, or message destinations (Yang et al., 2025; Ye et al., 2025b; Lu et al., 2026; Hou et al., 2025; Dang et al., 2025), leaving the representation of intermediate reasoning largely fixed.

Most LLM-based MAS still communicate through natural language. Text provides an explicit and inspectable representation, but verbalizing intermediate reasoning can introduce substantial context and decoding overhead (Yu et al., 2026a). Recent work therefore explores latent communication through embeddings, hidden states, or internal caches (Pham et al., 2024; Dery et al., 2026; Yu et al., 2026b; Liu et al., 2026b), together with broader forms of continuous reasoning (Berrayana et al., 2026; Tan et al., 2026; Xu et al., 2026; Jin et al., 2026; Fu et al., 2026; Shi et al., 2026). Latent representations reduce dependence on explicit text, but uniformly replacing text with latent messages may obscure information that benefits from precise symbolic expression. Consequently, neither all-text nor all-latent communication is consistently suitable throughout a reasoning trajectory. In practice, consistent with the no-free-lunch perspective (Wolpert & Macready, 1997), no single communication form performs consistently well across heterogeneous tasks.

The key challenge is therefore to determine *which representation is useful for each part of a reasoning trace*. A reasoning trace is heterogeneous: important derivations may appear alongside routine elaboration, repeated information, or misleading steps. Repeatedly exchanging such content increases communication cost and can affect downstream reasoning. A single communication decision for the entire trace cannot adapt to these local differences. Yang et al. (2026) further refine communication granularity by streaming intermediate reasoning step by step. However, they still communicate only through text, without choosing alternative representation forms. We instead perform *fine-grained utility-based communication routing*. As shown in Figure 1, we divide the trace into segments and consider three communication choices for each segment: TEXT preserves explicit reasoning, LATENT transmits probability-weighted soft embeddings, and SKIP omits the segment.

To guide these choices, we draw on lightweight internal signals exposed during generation, including entropy, maximum probability, and final-layer hidden-state norm. These signals provide observable evidence about uncertainty and internal representation dynamics; Figure 1(A) summarizes their variation with problem difficulty and reasoning correctness. Yet no individual signal directly establishes whether a reasoning segment is correct or whether it should be communicated. For example, high token probability does not necessarily imply logically reliable reasoning. We use these signals as complementary routing evidence. Additionally, communication utility should be defined by its effect on the receiving agent. We therefore derive routing labels from counterfactual receiver outcomes, directly linking each communication choice to downstream task performance.

We introduce **RELAY**, a two-level routing framework for multi-agent reasoning. Intra-agent routing removes the need to commit an entire trace to one communication form by making transmission decisions separately for its segments. This allows explicit details and latent semantic information to coexist in a message, with low-utility content omitted. A Signal-Conditioned Utility Router (SCUR) learns to estimate channel-specific utility from internal signals, grounding these decisions in expected downstream value. The resulting representation choices also affect which agents can process the message. An Adaptive Representation Orchestrator (ARO) addresses this dependency through inter-agent routing, using the routed message and task progress to choose a compatible recipient or end the interaction. In this way, subsequent collaboration accounts for both the information selected for transmission and the recipient's ability to use it. Together, these two levels unify adaptive communication and agent orchestration within a single routing framework.

Our main contributions are as follows.

- We adjust textual and latent communication through segment-wise routing, exploiting their complementary strengths for effective information exchange and establishing a new framework for latent communication routing in multi-agent systems.
- We show that uncertainty, confidence, and hidden-state signals within reasoning traces are informative of downstream task success, providing an empirical basis for communication-channel selection and adaptive agent orchestration.
- Through comprehensive experiments across foundational reasoning and long-horizon decision-making tasks, we show that RELAY improves average accuracy over the best-performing baseline by 2.3% and 11.7% while using 36.2% and 56.4% fewer tokens, respectively. Further analyses reveal distinct learned routing regimes and robustness to routing granularity.

2 PRELIMINARY

We formalize the components underlying fine-grained routing in multi-agent reasoning: token-level reasoning traces, segmentation, and multi-agent coordination over the resulting units.

Definition 2.1 (Token-level Reasoning Trace)**.** Given a query q, sender agent A_i generates an answer $\hat{a}_i$ together with a token-level reasoning trace $r^{(i)}_{1:K_i} = (r^{(i)}_1, \ldots, r^{(i)}_{K_i})$ of length K_i. At decoding step t, we associate the token $r^{(i)}_t$ with the next-token distribution $p^{(i)}_t(v) = p(v \mid q, r^{(i)}_{<t})$, $v \in \mathcal{V}$, and the final-layer hidden state $h^{(L,i)}_t$, where $\mathcal{V}$ is the vocabulary of the underlying language model.

Definition 2.2 (Segmentation and Signal Matrix)**.** For each token, we extract the entropy $E^{(i)}_t$ of the renormalized top-k distribution, maximum probability $P^{(i)}_t$, and hidden-state norm $N^{(i)}_t$. We partition the reasoning trace into $M_i = \lceil K_i/W \rceil$ non-overlapping segments $\{X^{(i)}_m\}_{m=1}^{M_i}$, each containing at most W tokens. Segment $X^{(i)}_m = (r^{(i)}_t)_{t \in I^{(i)}_m}$ has index set $I^{(i)}_m = \{(m-1)W + 1, \ldots, \min(mW, K_i)\}$ and effective length $T^{(i)}_m = |I^{(i)}_m|$. The corresponding signal matrix is $\mathbf{G}^{(i)}_m = [(E^{(i)}_t, P^{(i)}_t, N^{(i)}_t)^\top]_{t \in I^{(i)}_m} \in \mathbb{R}^{3 \times T^{(i)}_m}$. Each $\mathbf{G}^{(i)}_m$ characterizes local uncertainty, confidence, and internal-state dynamics to guide segment-level communication routing.

Multi-Agent Coordination. We consider a multi-agent system $\mathcal{A} = \{A_1, \ldots, A_J\}$ operating over interaction steps ℓ. At each step, the active agent A_i forms an inter-agent message from its reasoning segments $\{X^{(i)}_m\}_{m=1}^{M_i}$. An orchestration policy selects the next agent or terminates the interaction based on the current state s_ℓ, which summarizes task progress and communication history.

3 RELAY

As shown in Figure 2, RELAY separates *what to communicate* from *where to communicate next* in multi-agent reasoning. The Signal-Conditioned Utility Router (SCUR, § 3.1– 3.2) routes segment-wise communication, while the Adaptive Representation Orchestrator (ARO, § 3.3) selects the next compatible agent or terminates the interaction based on the current interaction state.

3.1 FINE-GRAINED COMMUNICATION ROUTING

At the intra-agent level, to better capture variations in communication utility across reasoning segments, we propose the Signal-Conditioned Utility Router (SCUR). For each segment $X^{(i)}_m$, SCUR selects a channel from $\mathcal{A}_{\text{comm}} = \{\text{TEXT}, \text{LATENT}, \text{SKIP}\}$. TEXT retains the segment as explicit reasoning text; LATENT transmits probability-weighted soft embeddings over top-k candidate tokens; and SKIP omits the segment. SCUR encodes $\mathbf{G}^{(i)}_m$ using a shared token-wise encoder $\phi_{\theta_{\text{enc}}}$ and pools the resulting representations:

$$z^{(i)}_m = \text{Pool}_{t \in I^{(i)}_m} \left[\phi_{\theta_{\text{enc}}} \left(E^{(i)}_t, P^{(i)}_t, N^{(i)}_t \right) \right], \tag{1}$$

where Pool concatenates the element-wise mean, maximum, and standard deviation, yielding a permutation-invariant segment representation. Segments with similar signal patterns may differ in communication utility because they convey different information. We therefore incorporate a content representation by averaging the final-layer hidden states within each segment:

$$\bar{h}^{(i)}_m = \frac{1}{|I^{(i)}_m|} \sum_{t \in I^{(i)}_m} h^{(L,i)}_t, \qquad \tilde{h}^{(i)}_m = \text{Proj}_{\theta_{\text{cont}}} \left(\text{LN} \left(\bar{h}^{(i)}_m \right) \right), \tag{2}$$

where $\tilde{h}^{(i)}_m$ provides a compact summary of the segment content. A prediction head $f_{\theta_{\text{pred}}}$ maps $[z^{(i)}_m; \tilde{h}^{(i)}_m]$ to three channel-specific logits $\boldsymbol{\xi}^{(i)}_m \in \mathbb{R}^3$, giving utility estimates

$$U^{(i)}_m = \sigma \left(\boldsymbol{\xi}^{(i)}_m \right) = \sigma \left(f_{\theta_{\text{pred}}} \left([z^{(i)}_m; \tilde{h}^{(i)}_m] \right) \right) \in [0,1]^3, \tag{3}$$

where σ applies a sigmoid to each logit and $U^{(i)}_{m,c}$ estimates the downstream utility of channel c. The signal branch captures uncertainty and confidence patterns, while the content branch provides

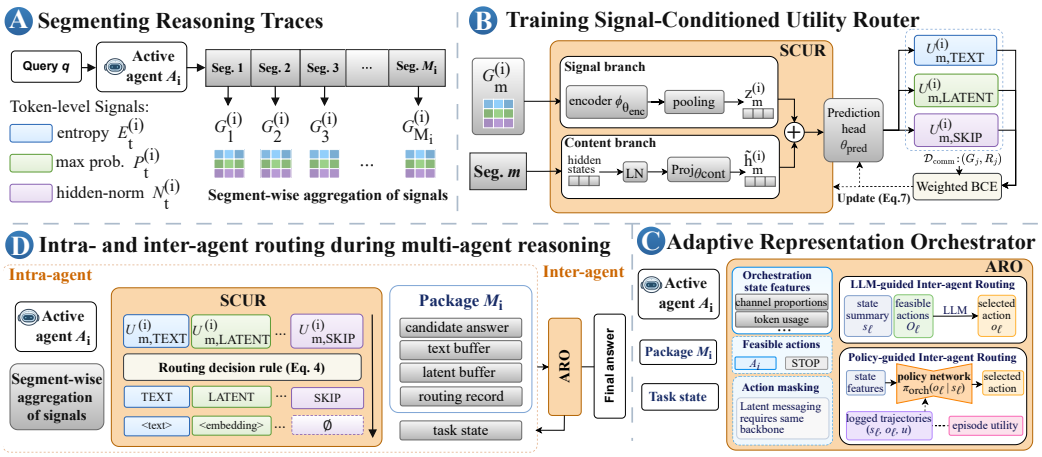


Figure 2: **Overview of RELAY.** (A) Reasoning traces are segmented into signal matrices. (B) SCUR learns channel utilities from counterfactual supervision. (C) ARO selects compatible agents. (D) SCUR routes segment-wise communication, while ARO coordinates inter-agent collaboration.

complementary semantic context for channel-wise utility estimation. These utilities are learned from channel-specific receiver outcomes in Section 3.2, rather than a single preferred-channel label. For each segment, SCUR compares the best transmitting channel with SKIP using a threshold τ:

$$\Delta_m^{(i)} = \max\left(U_{m,\mathrm{TEXT}}^{(i)}, U_{m,\mathrm{LATENT}}^{(i)}\right) - U_{m,\mathrm{SKIP}}^{(i)}, \quad a_m^{(i)} = \begin{cases} \mathrm{SKIP}, & \Delta_m^{(i)} \le \tau, \\ \underset{c\in\{\mathrm{TEXT},\mathrm{LATENT}\}}{\arg\max} U_{m,c}^{(i)}, & \text{otherwise.} \end{cases} \tag{4}$$

The selected text and latent outputs are collected into separate buffers in their original segment order. Together with the candidate answer $\hat{a}_i$ and a routing record of segment indices, token ranges, and selected channels, these buffers form the mixed-format message $\mathcal{M}_i$. We denote this construction by Assemble. Implementation is further specified in Appendix B.1 for encoding and prediction, and Appendix B.2 for channel representations and assembly formulas.

3.2 COUNTERFACTUAL UTILITY LEARNING

We construct counterfactual receiver outcomes to obtain direct supervision for the downstream utility of each communication channel. For a solvable training problem (q, y) with reference answer y, we fix a sender trajectory $\mathcal{T}$ containing M segments and omit the sender superscript in this subsection. We sample a background mask $b \in \{0,1\}^M$ satisfying $\|b\|_0 = k_{\mathrm{keep}}$, where k_{keep} is the retention count. The background assignment $a_n^{(b)}$ is TEXT when $b_n = 1$ and SKIP otherwise. For target segment X_m and channel $c \in \mathcal{A}_{\mathrm{comm}}$, we change only the target assignment and assemble the counterfactual message:

$$a_n^{(b;m\leftarrow c)} = \begin{cases} c, & n = m, \\ a_n^{(b)}, & n \neq m, \end{cases} \quad \mathcal{M}_{m,c}^{(b)} = \mathrm{Assemble}\left(\mathcal{T}, a^{(b;m\leftarrow c)}\right). \tag{5}$$

To compare communication choices under consistent conditions, we use a fixed receiver A_{rec} sharing the sender's backbone to process each counterfactual message under the same response protocol. Its extracted answer and binary correctness label are

$$\hat{y}_{m,c}^{(b)} = \mathrm{Extract}_{\mathrm{ans}}\left(A_{\mathrm{rec}}(q, \mathcal{M}_{m,c}^{(b)})\right), \quad R_{m,c}^{(b)} = \mathrm{Eval}(\hat{y}_{m,c}^{(b)}, y) \in \{0,1\}. \tag{6}$$

Here, $\mathrm{Extract}_{\mathrm{ans}}$ reads the initial answer, and Eval applies the task's binary correctness criterion. For each trace–segment–mask instance j, we store the target segment's signal matrix $\mathbf{G}_j$ and all three outcomes $\mathbf{R}_j = (R_{j,\mathrm{TEXT}}, R_{j,\mathrm{LATENT}}, R_{j,\mathrm{SKIP}})$ in $\mathcal{D}_{\mathrm{comm}}$. These counterfactual records provide the supervision for learning channel utilities from segment-level signals. Let $U_{j,c}$ denote

the utility predicted for channel c on example j. We jointly optimize the encoder and prediction-head parameters $\theta = (\theta_{\text{enc}}, \theta_{\text{cont}}, \theta_{\text{pred}})$ by $\theta^\star = \arg\min_\theta \mathcal{L}_{\text{comm}}(\theta)$, with

$$\mathcal{L}_{\text{comm}}(\theta) = \sum_{j=1}^{|\mathcal{D}_{\text{comm}}|} w_j \sum_{c \in \mathcal{A}_{\text{comm}}} \text{BCE}(U_{j,c}, R_{j,c}), \quad w_j = 1 + \lambda\, \mathbf{1}[R_{j,\text{TEXT}} \neq R_{j,\text{LATENT}}]\,. \tag{7}$$

Here, BCE is binary cross-entropy and λ controls the additional weight on examples whose TEXT and LATENT outcomes differ. The same weight applies to all three channel losses, covering both directions of text–latent disagreement. The resulting objective learns channel utility directly from counterfactual receiver outcomes. Additional training details are given in Appendix B.3.

3.3 Adaptive Inter-Agent Routing

At the inter-agent level, communication choices constrain compatible recipients, while task progress changes the need for continued execution or specialist intervention. We introduce the Adaptive Representation Orchestrator (ARO), which jointly accounts for representation compatibility, interaction progress, and communication cost when routing across agents:

$$o_\ell \sim \pi^{\text{orch}}(\cdot \mid s_\ell), \qquad o_\ell \in \mathcal{O}_\ell \subseteq \mathcal{A} \cup \{\text{STOP}\}. \tag{8}$$

Here, s_ℓ summarizes channel usage derived from $\mathcal{M}_i$, generation uncertainty, interaction progress, and cumulative token usage. The feasible set $\mathcal{O}_\ell$ excludes incompatible recipients: latent-containing messages are restricted to agents sharing the sender's checkpoint and supporting soft-embedding inputs, while selecting A_i assigns the next execution step to the selected agent. ARO is policy-agnostic, and we instantiate two routing policies. **ReLay**$_{\text{LLM}}$ uses the frozen backbone to select one action from the feasible set given a compact state summary. **ReLay**$_{\text{Policy}}$ instead uses a lightweight policy trained from offline trajectories, avoiding an additional model call. Episode utility combines success with token and step penalties and is assigned to each logged policy decision. During training, routing decisions that lead to successful task completion with fewer tokens and interaction steps are given greater importance. Both variants share the same state, feasibility constraints; only the routing policy differs. Full state features and policy training details are provided in Appendix E.1.

4 Foundational Reasoning Tasks

4.1 Setup

Manually designed MAS provide reliable coordination priors for foundational reasoning tasks (Lei et al., 2024). We therefore evaluate under validated MAS settings. Under this setting, RELAY achieves the best accuracy–efficiency trade-off, with analysis revealing the sources of the gains.

Tasks and Metrics. We consider four datasets: **HMMT** and **AIME2026** (Dekoninck et al., 2026), **GPQA-D** (Rein et al., 2023) and **MATH** (Hendrycks et al., 2021). For performance, we report the accuracy (**Acc.**). For efficiency, we report the average number of total tokens per question (**Token**).

Baselines and Variants. We compare against baselines categorized by their coordination mechanisms: **(I) single-agent methods** including **CoT** (Wei et al., 2022) and **Self-Refine** (Madaan et al., 2023). **(II) multi-agent methods** including text-based MAS (**TextMAS**) (Lei et al., 2024; Zhuge et al., 2024), **LatentMAS** (Zou et al., 2025), **SDE** (Tang et al., 2025), **CIPHER** (Pham et al., 2024), **InterLat** (Du et al., 2026) and **HyLaT** (Mou et al., 2026). **RELAY**$_{\text{Core}}$ denotes the communication-routing variant of our framework, where the MAS structure is fixed and only intra-agent routing is adaptive.

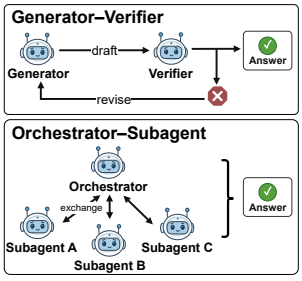


Figure 3: Illustration of two Fixed MAS Settings

Fixed MAS Settings. For multi-agent methods, we instantiate two widely adopted coordination structures as shown in Figure 3. In *generator–verifier*, a sequential structure enables iterative critique and revision of a candidate output. In *orchestrator–subagent*, a hierarchical structure supports top-down task allocation and result aggregation. These two settings provide a compact yet representative basis for evaluation. Full details of the experimental setup are provided in Appendix C.

Method	Qwen3.5-4B						Qwen3.5-9B					
	HMMT		AIME2026		GPQA-D		HMMT		AIME2026		GPQA-D	
	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓
Single-Agent												
CoT	60.6	45.6	76.7	29.9	78.3	13.0	63.6	44.2	76.7	30.6	82.3	9.0
Self-Refine	63.6	102.5	83.3	**89.3**	77.3	17.0	69.7	95.9	80.0	67.5	80.8	22.1
MAS Setting: Generator–Verifier												
TextMAS	54.6	131.0	73.3	43.3	77.1	57.1	60.6	62.3	80.0	**40.9**	78.8	72.7
LatentMAS	42.4	43.9	50.0	25.5	62.1	21.4	42.4	50.0	66.7	45.4	61.1	35.6
SDE	42.4	52.4	70.0	55.9	64.1	28.5	51.5	47.2	80.0	44.9	67.2	33.2
RELAY$_{\text{Core}}$	58.6	39.7	73.3	43.2	77.5	11.7	69.7	**71.8**	81.1	60.7	**87.5**	**18.2**
MAS Setting: Orchestrator–Subagent												
TextMAS	**69.7**	165.0	83.3	220.5	77.3	33.0	**72.7**	303.3	86.7	218.3	83.8	86.7
LatentMAS	36.4	65.9	63.3	64.9	71.2	25.4	59.6	39.4	60.0	43.4	77.3	19.1
CIPHER	44.4	140.8	63.3	121.1	78.8	34.4	54.6	250.6	73.3	195.3	79.8	76.5
RELAY$_{\text{Core}}$	66.7	**95.9**	**86.7**	103.3	**82.5**	**17.3**	70.7	209.1	**93.3**	154.7	85.0	46.5
Δ *vs. Best Baseline*	↓ 4.3%	↓ 41.9%	↑ 4.0%	↑ 15.7%	↑ 4.7%	↓ 49.7%	↓ 2.8%	↓ 31.1%	↑ 7.7%	↓ 29.1%	↑ 4.4%	↓ 79.0%

Table 1: Main results on three reasoning-intensive tasks under different MAS settings. For token consumption, comparisons are restricted to methods that outperform CoT in accuracy. The best results are shown in **bold**, and the second-best results are underlined.

Method	Overall		Level 1		Level 2		Level 3		Level 4		Level 5	
	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓	Acc.↑	Token (k)↓
CoT	51.1	0.6	82.7	0.3	64.0	0.5	52.0	0.5	34.0	0.8	22.7	1.0
TextMAS	61.3	5.7	90.0	2.9	74.7	4.0	63.3	5.5	50.0	7.2	28.7	9.0
LatentMAS	58.2	2.1	88.7	0.8	71.3	1.2	55.3	1.8	47.3	3.4	28.4	3.5
SDE	54.5	0.9	80.7	0.4	64.0	0.7	56.7	1.0	43.3	1.1	28.0	1.4
CIPHER	56.1	6.5	84.7	2.6	68.0	4.8	55.3	6.1	43.3	8.1	29.1	10.6
HyLaT	51.7	0.9	86.9	0.6	60.9	0.7	52.4	0.9	32.0	1.0	26.1	1.2
InterLat	57.5	1.0	88.2	0.4	66.4	0.7	61.3	1.0	43.3	1.4	28.4	1.7
RELAY$_{\text{Core}}$	62.8	4.4	88.0	2.2	72.0	3.0	74.0	4.3	50.0	5.6	30.0	6.7
Δ *vs. Best Baseline*	↑ 2.4%	↓ 22.8%	↓ 2.2%	↓ 24.1%	↓ 3.6%	↓ 25.0%	↑ 16.9%	↓ 21.8%	0.0%	↓ 22.2%	↑ 3.1%	↓ 36.8%

Table 2: Results on MATH across different difficulty levels. We evaluate Llama-3.1-8B-Instruct, reporting results for Levels 1–5. The ***Best Baseline*** comparison rule is detailed in Appendix C.3.

4.2 Main Results

Table 1 presents a comprehensive comparison of our framework against baselines. By enabling adaptive and hybrid communication, our method improves performance and consistently outperforms both single-agent baselines and multi-agent systems that rely exclusively on either text or latent communication. We highlight several key observations below.

Obs.❶ No single communication form dominates across all tasks. As shown in Table 1, different communication forms exhibit varying strengths across benchmarks, our framework achieves the best overall performance. Textual and latent communication exhibit **task- and trace-dependent advantages**, suggesting that a fixed communication strategy cannot simultaneously accommodate different reasoning and coordination requirements. In contrast, RELAY$_{\text{Core}}$ dynamically selects among TEXT, LATENT, and SKIP according to the current reasoning signals and states. This adaptive routing capability enables our method to achieve the strongest overall accuracy–efficiency trade-off. Across tasks and model scales, RELAY$_{\text{Core}}$ improves accuracy by 18.5% on average over MAS baselines, while reducing token consumption by 35.9% relative to the best baseline.

Obs.❷ Adaptive routing reconciles text and latent advantages across difficulty levels. Table 2 shows that latent-based baselines lag on easier problems but become more effective as difficulty increases. CIPHER reaches 29.1% accuracy on Level 5, surpassing both TextMAS (28.7%) and CoT (22.7%). This trend is consistent with prior findings that discrete decoding provides greater stability when reasoning is relatively certain, whereas latent representations become more valuable when broader exploration is required by preserving alternative reasoning trajectories (Fu & Luo, 2026; Zhu et al., 2026). In this context, internal trace signals provide a lightweight cue for adaptive routing. On Levels 1–2, RELAY$_{\text{Core}}$ reduces token usage by 24.1–25.0% with modest accuracy drops. On Levels 3–5, it achieves the best accuracy while reducing tokens by 21.8–36.8%.

Case Study. Figure 4 illustrates how RELAY$_{\text{Core}}$ performs fine-grained routing. The trace is divided into 8 segments. RELAY$_{\text{Core}}$ skips Segments 1–2, which contain low-value preamble and restated constraints, while preserving Segment 3 in text form because it contains the key symbolic step, $(3t-$

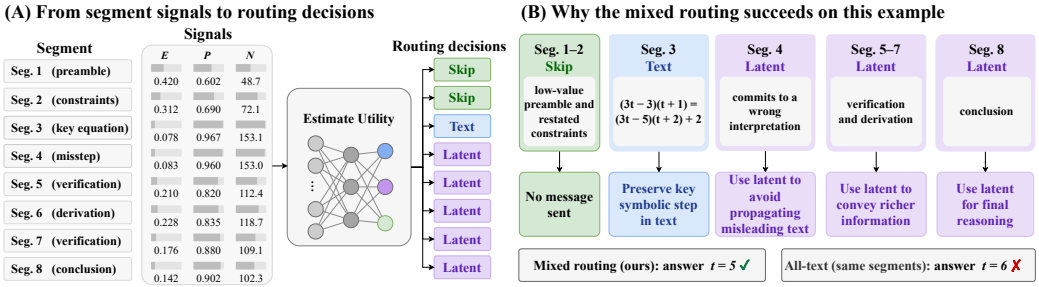


Figure 4: Case study: On a MATH problem, RELAY$_{\text{Core}}$ skips low-value preamble, keeps a key symbolic step in text, and uses the latent channel for the reasoning, yielding the correct answer.

Method | Qwen3.5-4B	MATH		AIME2026		GPQA-D	
	Acc. (%) ↑	Token (k) ↓	Acc. (%) ↑	Token (k) ↓	Acc. (%) ↑	Token (k) ↓
RELAY$_{\text{Core}}$	77.5	14.9	86.7	103.3	82.5	17.3
Router inputs						
Content only	76.7	15.1	83.3	144.0	65.0	13.7
Signals only	77.5	16.8	83.3	144.2	81.3	16.0
Utility learning						
Random routing	70.0	12.0	73.3	89.4	79.4	14.3
Entropy routing	74.3	7.2	80.0	130.0	82.5	19.6
Segmentation						
Whole-trace routing	75.0	10.4	26.7	201.9	70.0	10.3
Channel variants						
Text only	77.5	28.6	83.3	208.0	79.4	36.4
w/o embedding	73.2	16.6	82.2	116.3	79.4	18.6
w/o skip	77.5	24.3	83.3	177.0	75.0	34.8

Table 3: Ablation study of RELAY$_{\text{Core}}$ across routing, segmentation, and communication channels.

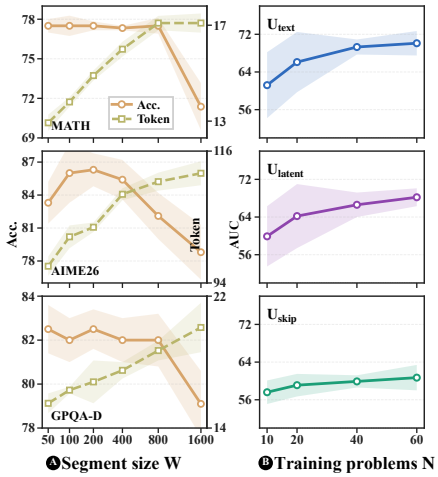


Figure 5: Sensitivity of RELAY$_{\text{Core}}$ to segment size and training-set size.

$3)(t+1) = (3t-5)(t+2)+2$. Segment 4 is routed through the latent channel: counterfactual replay shows that its textual form commits to an incorrect interpretation, whereas the latent representation preserves useful information without propagating the misleading expression. Segments 5–8 remain latent to convey richer reasoning information. Notably, when the same segments are communicated entirely in text, the downstream agent fails, whereas our method leads to the correct answer $t = 5$.

4.3 Framework Analysis

Ablation Study. Table 3 validates four key design principles of RELAY$_{\text{Core}}$. First, SCUR benefits from complementary inputs: content-only routing drops GPQA-D accuracy from 82.5% to 65.0%, while signals alone is generally weaker than the full router. Second, learned channel utility provides more reliable routing than heuristics, as random and entropy-based routing consistently underperform the learned router across benchmarks. Third, fine-grained routing is essential: replacing segment-level decisions with whole-trace routing reduces AIME2026 accuracy from 86.7% to 26.7% and GPQA-D from 82.5% to 70.0%. Finally, the communication channels are complementary. Removing LATENT decreases accuracy across all three tasks, whereas SKIP reduces token consumption by 38.7–50.3% relative to the no-skip variant. Together, these results attribute the gains of RELAY$_{\text{Core}}$ to segment-level utility routing over complementary communication channels.

Sensitivity Analysis. We examine the sensitivity of RELAY$_{\text{Core}}$ to the segment size W and the amount of training supervision N. As shown in Figure 5, accuracy remains broadly stable for $W =$ 50–400, while communication cost generally increases with segment size; overly large segments eventually degrade accuracy. This trend is consistent with our motivation for fine-grained routing: larger segments bundle heterogeneous reasoning content into fewer routing units, limiting selective skipping and representation choice. We therefore use $W = 200$ as the default. Increasing the number of training problems consistently improves the estimated utilities for TEXT, LATENT, and SKIP, with gains gradually saturating around 40 problems. These results indicate that our method is robust to routing granularity and reaches stable performance with limited supervision.

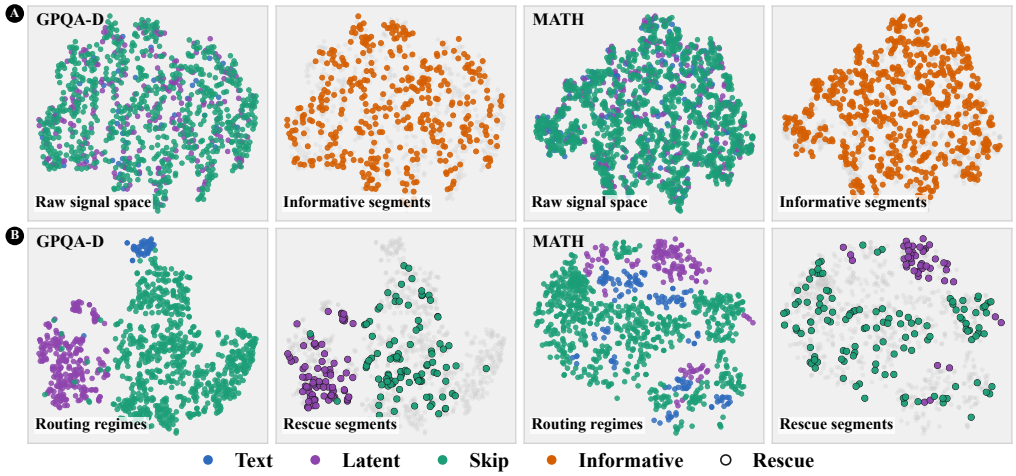


Figure 6: (A) Raw signal spaces, colored by routing labels and informative segments. (B) Learned representations, showing clearer routing regimes and the distribution of rescue segments.

Setting	Method	ALFWorld			WebShop		
		Seen ↑	Unseen ↑	Token (k) ↓	Score ↑	Succ. ↑	Token (k) ↓
Fixed	Qwen3.5-9B	40.8	33.3	252.6	65.0	30.8	18.1
	CoT	40.3	40.0	256.1	53.2	27.1	39.8
	Self-Refine	37.5	38.3	503.9	63.9	32.5	58.8
	TextMAS	39.2	32.5	771.3	65.0	33.3	38.1
Frontier Models	Kimi-K2.6	41.7	43.3	320.0	62.8	33.3	30.3
	DeepSeek-V4-Pro	43.1	42.6	267.5	66.1	33.8	51.1
	Gemini-3.5-Flash	38.5	37.2	278.7	63.4	34.2	38.0
Adaptive	AFlow	49.7	48.1	365.8	58.9	30.0	48.9
	MaAS	46.7	43.3	356.4	62.1	33.3	49.1
	RELAY$_{\text{LLM}}$	56.9	56.7	150.4	67.1	36.4	19.7
	RELAY$_{\text{Policy}}$	55.0	58.6	143.2	67.9	36.7	18.3

Table 4: Results on decision-making benchmarks. We report task performance and interaction efficiency on ALFWorld and WebShop.

4.4 SIGNAL ANALYSIS

Obs.❸ Learned router representations reveal distinct communication regimes. We examine whether internal signals distinguish segment-level communication needs by comparing raw signals with learned router representations and evaluating downstream correctness under TEXT, LATENT, and SKIP. Figure 6 shows substantial overlap in the raw signal space but clearer channel-specific regions in the learned space. Orange points highlight informative segments, whose task-level correctness differs across the three counterfactual routes. Within this subset, rescue cases remain correct under the selected LATENT or SKIP route but fail when forced to TEXT. These cases illustrate how adaptive routing can preserve correctness when textual communication is harmful.

5 LONG-HORIZON DECISION-MAKING TASKS

We further evaluate our method on coordination-intensive decision-making tasks, where agents interact with environments over long horizons under sparse feedback. We evaluate on two interactive benchmarks: ALFWorld (Shridhar et al., 2020) for embodied household decision-making and WebShop (Yao et al., 2022) for goal-directed web navigation. Such tasks place greater demands on efficient communication and adaptive orchestration. We compare **fixed methods**, **frontier-model**, and **adaptive methods**. Table 4 shows that our method improves task completion with a favorable interaction cost. On ALFWorld, both variants outperform the compared baselines in seen and unseen environments, with RELAY$_{\text{Policy}}$ reaching 58.6% success on unseen tasks. On WebShop, the

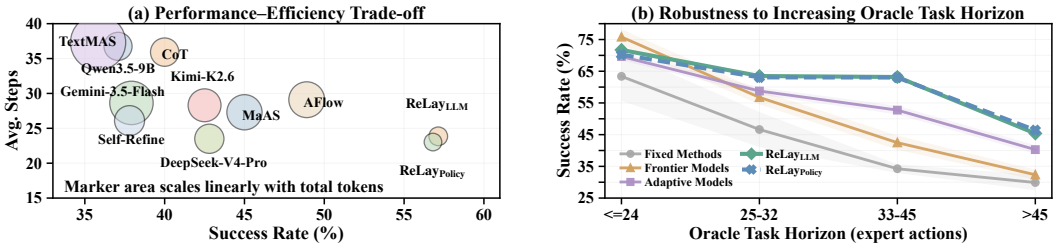


Figure 7: Analysis on ALFWorld. (a) Performance–efficiency trade-off across methods. (b) Success rate across decision horizons, showing that our methods remain robust on longer trajectories.

policy variant achieves the highest score and success rate with token consumption close to that of the standalone backbone. Figure 7(a) further shows that higher success rates are accompanied by fewer environment steps than most baselines. The key observation is that additional coordination does not translate into longer interaction. In a setting where actions change subsequent observations, this pattern is consistent with more effective task execution, extending the benefits of adaptive coordination beyond answer generation under fixed interaction structures. Figure 7(b) shows that the success-rate advantage persists over longer task horizons, measured by expert action sequences. For tasks exceeding 45 expert actions, both variants maintain approximately 46% success, compared with 30% for the fixed-baseline average. The gains are therefore not confined to short episodes, but extend to tasks requiring sustained decision-making across successive environmental transitions. The two orchestration variants also achieve comparable performance, while the learned policy consumes fewer tokens on both benchmarks. These results support adapting communication and agent selection throughout execution, while showing that the resulting performance benefits can be retained with a more token-efficient orchestration policy. More details are provided in Appendix E.

6 Related Work

Latent Reasoning and Communication. Recent studies have explored latent space as an alternative to token-level reasoning and communication. For reasoning, methods move intermediate computation from chain-of-thought tokens to continuous latent states, aiming to reduce verbalization cost while preserving reasoning capacity (Hao et al., 2025; Xu et al., 2026; Tan et al., 2026; Liu et al., 2026a; Zeng et al., 2026; Fu & Luo, 2026). In parallel, latent communication methods (Du et al., 2026; Tang et al., 2025; Mou et al., 2026; Liu, 2026) attempt to replace text messages with more direct semantic transfer between agents. Pham et al. (2024) uses embedding-based communication for multi-agent debate, while Zheng et al. (2025) and Zou et al. (2025) enable agents to exchange latent thoughts or reasoning states during collaboration. Other works improve efficiency by reusing or transferring internal caches (Fu et al., 2025; Shi et al., 2026; Ye et al., 2025a).

Routing for Multi-Agent Systems. Routing has become an important mechanism for adapting multi-agent systems to diverse task and system conditions. Existing work mainly routes three types of objects. First, system-level routers select collaboration modes, agent roles, LLM backbones, or edge-level strategies to construct query-specific MAS configurations (Yue et al., 2025; Xiao et al., 2026; Zhao et al., 2026). Second, topology- and workflow-level methods dynamically activate agents, update communication graphs, or search architectures as reasoning proceeds (Dang et al., 2025; Hou et al., 2025; Lu et al., 2026; Zhang et al., 2025a; Li et al., 2026). Third, efficiency-oriented methods incorporate budget, confidence, model heterogeneity, or memory relevance to reduce cost (Yang et al., 2025; Wang et al., 2026; Ye et al., 2025b; Feng et al., 2026; Liu et al., 2025).

7 Conclusion

We introduced ReLay, a two-level framework for adaptive communication and orchestration in LLM-based multi-agent systems. ReLay performs fine-grained routing across Text, Latent, and Skip, while coordinating agent interactions according to the evolving task state. Across reasoning and long-horizon decision-making benchmarks, ReLay consistently improves the performance–efficiency trade-off over fixed communication and adaptive multi-agent baselines. Our results show that effective multi-agent communication benefits from state-dependent representations rather than a fixed protocol. Overall, ReLay frames communication and coordination as a unified adaptive routing problem, providing a general approach to efficient collaboration in future agent systems.

AI USE STATEMENT

In this work, we used GPT-5.5 for semantic reasoning-step segmentation and step-level validity annotation; Additionally, we used generative AI tools to polish author-written text, improve the presentation and layout of the manuscript, and identify relevant literature. All AI-assisted content was critically reviewed, revised, and verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the implementation and experimental details necessary to reproduce our results in the appendix. Specifically, we describe the signal computation, mixed-message construction, segment-wise communication routing, and inter-agent orchestration procedures, together with their training objectives and hyperparameters. We also report the model backbones, decoding configurations, prompts, dataset and evaluation protocols, baseline adaptations, and experimental settings used throughout the paper. Additional ablations and sensitivity analyses are provided to verify the robustness of the main findings. Anonymous source code and configuration files are also provided.

REFERENCES

Lina Berrayana, Ahmed Heakl, Abdullah Sohail, Thomas Hofmann, Salman Khan, and Wei Chen. Latent-darm: Bridging discrete diffusion and autoregressive models for reasoning. *arXiv preprint arXiv:2603.09184*, 2026.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje Karlsson, Jie Fu, and Yemin Shi. Autoagents: a framework for automatic agent generation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, 2024a. ISBN 978-1-956792-04-1. doi: 10.24963/ijcai.2024/3. URL https://doi.org/10.24963/ijcai.2024/3.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/578e65cdee35d00c708d4c64bce32971-Paper-Conference.pdf.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11534–11557, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.601. URL https://aclanthology.org/2025.findings-acl.601/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration. *arXiv preprint arXiv:2505.19591*, 2025.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvalddson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. 2026. URL https://arxiv.org/abs/2605.00674.

Lucio M. Dery, Zohar Yahav, Henry Prior, Qixuan Feng, Jiajun Shen, and Arthur Szlam. Latent space communication via k-v cache alignment. *arXiv preprint arXiv:2601.06123*, 2026.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27106–27129, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1248. URL https://aclanthology.org/2026.acl-long.1248/.

Tao Feng, Haozhen Zhang, Zijie Lei, Peixuan Han, and Jiaxuan You. Graphplanner: Graph memory-augmented agentic routing for multi-agent llms. *arXiv preprint arXiv:2604.23626*, 2026.

Muxin Fu, Xiangyuan Xue, Yafu Li, Zefeng He, Siyuan Huang, Xiaoye Qu, Yu Cheng, and Yang Yang. Latentmem: Customizing latent memory for multi-agent systems. *arXiv preprint arXiv:2602.03036*, 2026.

Renyu Fu and Guibo Luo. SeLaR: Selective latent reasoning in large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7073–7087, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.320. URL https://aclanthology.org/2026.acl-long.320/.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. *arXiv preprint arXiv:2510.03215*, 2025.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models, 2024. URL https://arxiv.org/abs/2410.07985.

Google DeepMind. Gemini 3.5 Flash model card. Model card, May 2026. URL https://deepmind.google/models/model-cards/gemini-3-5-flash/. Accessed: 2026-09-20.

Yao Guan, Lin Wang, Zhihu Lu, Ziyi Wang, Wenzhu Yan, and Qiang Duan. Moc: Multi-order communication in llm-based multi-agent systems. *arXiv preprint arXiv:2606.02359*, 2026.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: a survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, 2024. ISBN 978-1-956792-04-1. doi: 10.24963/ijcai.2024/890. URL https://doi.org/10.24963/ijcai.2024/890.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, zili wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6507b115562bb0a305f1958ccc87355a-Paper-Conference.pdf.

Zhipeng Hou, Junyi Tang, and Yipeng Wang. Halo: Hierarchical autonomous logic-oriented orchestration for multi-agent llm systems. *arXiv preprint arXiv:2505.13516*, 2025.

Chen Huang, Yuhao Wu, and Wenxuan Zhang. What should agents say? action-state communication for efficient multi-agent systems. *arXiv preprint arXiv:2606.05304*, 2026.

Haibo Jin, Kuang Peng, Ye Yu, Xiaopeng Yuan, and Haohan Wang. Agent primitives: Reusable latent building blocks for multi-agent systems. *arXiv preprint arXiv:2602.03695*, 2026.

Yuxin Jin, Hanchen Wang, Ying Zhang, Dong Wen, Lu Qin, and Wenjie Zhang. Data in agentic ai: A comprehensive survey. *TechRxiv*, 2025(1229), 2025. doi: 10.36227/techrxiv.176704339.93520117/v1. URL https://www.techrxiv.org/doi/abs/10.36227/techrxiv.176704339.93520117/v1.

Bin Lei, Yi Zhang, Shan Zuo, Ali Payani, and Caiwen Ding. Macm: Utilizing a multi-agent system for condition mining in solving complex mathematical problems. *Advances in Neural Information Processing Systems*, 37:53418–53437, 2024.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a3621ee907def47c1b952ade25c67698-Paper-Conference.pdf.

Haoran Li, Shulun Chen, Shaoyuan Sun, and Hanchen Wang. Multi-agent coordination adaptation via structure-guided orchestration. *arXiv preprint arXiv:2605.25746*, 2026.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

Yi-Cheng Lin, Kang-Chieh Chen, Zhe-Yan Li, Tzu-Heng Wu, Tzu-Hsuan Wu, Kuan-Yu Chen, Hung-yi Lee, and Yun-Nung Chen. Creativity in LLM-based multi-agent systems: A survey. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 27584–27607, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1403. URL https://aclanthology.org/2025.emnlp-main.1403/.

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

Weihao Liu, Dehai Min, and Lu Cheng. Latent thoughts tuning: Bridging context and reasoning with fused information in latent tokens. *arXiv preprint arXiv:2602.10229*, 2026a.

Xiaoze Liu, Ruowang Zhang, Weichen Yu, Siheng Xiong, Liu He, Feijie Wu, Hoin Jung, Matt Fredrikson, Xiaoqian Wang, and Jing Gao. The vision wormhole: Latent-space communication in heterogeneous multi-agent systems. *arXiv preprint arXiv:2602.15382*, 2026b.

Yingzhuo Liu. Beyond tokens: A unified framework for latent communication in llm-based multi-agent systems. *arXiv preprint arXiv:2606.05711*, 2026.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization, 2023. URL https://arxiv.org/abs/2310.02170.

Yuxing Lu, Yucheng Hu, Xukai Zhao, and Jiuxin Cao. Dytopo: Dynamic topology routing for multi-agent reasoning via semantic matching. *arXiv preprint arXiv:2602.06039*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Moonshot AI. Kimi K2.6: Advancing open-source coding. Technical blog, 2026. URL https://www.kimi.ai/blog/kimi-k2-6. Accessed: 2026-09-20.

Xinyi Mou, Siyuan Wang, Zejun Li, Yulan He, and Zhongyu Wei. Hylat: Efficient multi-agent communication via hybrid latent-text protocol. *arXiv preprint arXiv:2605.25421*, 2026.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings. In *International Conference on Learning Representations*, volume 2024, pp. 51899–51928, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e444859b2a22df6b56af9381ad1e9480-Paper-Conference.pdf.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Shuai Shao, Kangning Zhang, Qingyao Li, Shijian Wang, Hao Wang, Wenxiang Jiao, Yuan Lu, Yi Guo, Weiwen Liu, and Weinan Zhang. Harness-r1: Learning to edit executable runtime harnesses from agent failure trajectories. *arXiv preprint arXiv:2608.02276*, 2026.

Xiangyu Shi, Marco Chiesa, Gerald Q. Maguire Jr., and Dejan Kostic. Kvcomm: Enabling efficient llm communication through selective kv sharing. *arXiv preprint arXiv:2510.03346*, 2026.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Chuanneng Sun, Songjun Huang, and Dario Pompili. Llm-based multi-agent reinforcement learning: Current and future directions. *arXiv preprint arXiv:2405.11106*, 2024.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *arXiv preprint arXiv:2505.16552*, 2026.

Yichen Tang, Weihang Su, Yujia Zhou, Yiqun Liu, Min Zhang, Shaoping Ma, and Qingyao Ai. Augmenting multi-agent communication with state delta trajectory. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10219–10240, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.518. URL https://aclanthology.org/2025.emnlp-main.518/.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*, 2025.

vLLM Semantic Router Team. vllm semantic router. https://github.com/vllm-project/semantic-router, 2025.

Jingbo Wang, Sendong Zhao, Jiatong Liu, Haochun Wang, Wanting Li, Bing Qin, and Ting Liu. Orchestrating intelligence: Confidence-aware routing for efficient multi-agent collaboration across multi-scale models. *arXiv preprint arXiv:2601.04861*, 2026.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. AgentDropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1170. URL https://aclanthology.org/2025.acl-long.1170/.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Yitao Xiao, Shaoyong Guo, guoming yang, Qingnan Wang, Yinlin Ren, Xuesong Qiu, and Feng Qi. Routerhgc: Optimized router for llm-based multi-agent systems via heterogeneous graph contrastive learning. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '26, pp. 3313–3316, Richland, SC, 2026. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400723179. doi: 10.65109/PGWI7290. URL https://doi.org/10.65109/PGWI7290.

Xin Xu, Tong Yu, Xiang Chen, Haoliang Wang, Julian McAuley, and Saayan Mitra. Thinkrouter: Efficient reasoning via routing thinking between latent and discrete spaces. *arXiv preprint arXiv:2602.11683*, 2026.

Bingyu Yan, Zhibo Zhou, Litian Zhang, Lian Zhang, Ziyi Zhou, Dezhuang Miao, Zhoujun Li, Chaozhuo Li, and Xiaoming Zhang. Beyond self-talk: A communication-centric survey of llm-based multi-agent systems. *arXiv preprint arXiv:2502.14321*, 2025.

Liming Yang, Junyu Luo, Xuanzhe Liu, Yiling Lou, and Zhenpeng Chen. Bamas: Structuring budget-aware multi-agent systems. *arXiv preprint arXiv:2511.21572*, 2025.

Zhen Yang, Xiaogang Xu, Wen Wang, Cong Chen, Xander Xu, and Ying-Cong Chen. Streaming communication in multi-agent reasoning. *arXiv preprint arXiv:2606.05158*, 2026.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. Kvcomm: Online cross-context kv-cache communication for efficient llm-based multi-agent systems. *arXiv preprint arXiv:2510.12872*, 2025a.

Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. X-mas: Towards building multi-agent systems with heterogeneous llms. *arXiv preprint arXiv:2505.16997*, 2025b.

Xinlei Yu, Zhangquan Chen, Yongbo He, Tianyu Fu, Cheng Yang, Chengming Xu, Yue Ma, Xiaobin Hu, Zhe Cao, Jie Xu, Guibin Zhang, Jiale Tao, Jiayi Zhang, Siyuan Ma, Kaituo Feng, Haojie Huang, Youxing Li, Ronghao Chen, Huacan Wang, Chenglin Wu, Zikun Su, Xiaogang Xu, Kelu Yao, Kun Wang, Chen Gao, Yue Liao, Ruqi Huang, Tao Jin, Cheng Tan, Jiangning Zhang, Wenqi Ren, Yanwei Fu, Yong Liu, Yu Wang, Xiangyu Yue, Yu-Gang Jiang, and Shuicheng Yan. The latent space: Foundation, evolution, mechanism, ability, and outlook. *arXiv preprint arXiv:2604.02029*, 2026a.

Ye Yu, Heming Liu, Haibo Jin, Xiaopeng Yuan, Peng Kuang, and Haohan Wang. Learning to communicate: Toward end-to-end optimization of multi-agent language systems. *arXiv preprint arXiv:2604.21794*, 2026b.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. MasRouter: Learning to route LLMs for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15549–15572, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.757. URL https://aclanthology.org/2025.acl-long.757/.

Boyi Zeng, Yiqin Hao, He Li, Shixiang Song, Feichen Song, Zitong Wang, Siyuan Huang, Yi Xu, ZiWei He, Xinbing Wang, and Zhouhan Lin. Pretraining with token-level adaptive latent chain-of-thought. *arXiv preprint arXiv:2602.08220*, 2026.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Proceedings of the ICML 2025*, ICML'25. JMLR.org, 2025a.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*, 2025b.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the Crap: An Economical Communication Pipeline for LLM-Based Multi-Agent Systems. In *International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=LkzuPorQ5L.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-Designer: Architecting Multi-Agent Communication Topologies via Graph Neural Networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76678–76692. PMLR, 2025d.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025e.

Di Zhao, Longhui Ma, Siwei Wang, Miao Wang, and Yi Kong. Sc-mas: Constructing cost-efficient multi-agent systems with edge-level heterogeneous collaboration. *arXiv preprint arXiv:2601.09434*, 2026.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. *arXiv preprint arXiv:2510.20733*, 2025.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Arik. Multi-agent design: Optimizing agents with better prompts and topologies. In *International Conference on Learning Representations*, volume 2026, pp. 15844–15872, 2026.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart J Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *Advances in Neural Information Processing Systems*, 38:79931–79963, 2026.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025.

APPENDIX

A Signals 17

A.1 Signal Computation Details 17

A.2 Token-wise Error Propagation 18

A.3 Signal Statistics Across Evaluation Settings 19

B SCUR Architecture and Training Details **21**

B.1 SCUR Architecture 21

B.2 Mixed-Format Message Construction 21

B.3 SCUR Training Details 22

B.4 Multi-Agent Orchestration and Deployment 23

B.5 Communication-Channel Analyses and Receiver-Input Probes 23

B.6 Preliminary Three-Class Routing Analysis 24

C Experiment Setups **27**

C.1 Evaluation Datasets 27

C.2 Compared Baselines 28

C.3 Evaluation Protocol 29

C.4 Implementation Details 29

D Additional Experiments **30**

D.1 Communication Channel Distribution across MAS Settings 30

D.2 RELAY Across Backbone Models 30

D.3 Additional Efficiency Analysis 30

E Long-Horizon Decision-Making Task Details **32**

E.1 Adaptive Orchestration Details 32

E.2 Additional Results 33

F Additional Discussions **34**

G Additional Related Work **35**

H Prompt Templates **36**

I Supplementary Case Study **40**

A SIGNALS

This appendix provides additional details and analyses of the reasoning signals used by RELAY. We first describe the computation of the token-level signals, followed by analyses of error propagation and signal statistics across problem difficulty and final answer correctness.

A.1 SIGNAL COMPUTATION DETAILS

Token-level reasoning trace. For a query q, sender agent A_i generates a reasoning trace $r^{(i)}_{1:K_i}$ of length K_i. At each decoding step t, we record the next-token distribution $p^{(i)}_t(\cdot)$ and the final-layer hidden state $h^{(L,i)}_t$. These quantities provide token-wise observations of the model's output uncertainty and internal representation dynamics.

Entropy. We measure local uncertainty using the entropy of a renormalized top-k probability distribution. Let $\mathcal{V}^{(i)}_k(t) = \mathrm{TopK}\Big(p^{(i)}_t, k\Big)$ denote the set of the k most probable tokens at decoding step t. The probability mass over these candidates is renormalized as

$$\hat{p}^{(i)}_t(v) = \frac{p^{(i)}_t(v)}{\sum_{u\in\mathcal{V}^{(i)}_k(t)} p^{(i)}_t(u)}, \qquad v \in \mathcal{V}^{(i)}_k(t). \tag{9}$$

The entropy signal is then

$$E^{(i)}_t = -\sum_{v\in\mathcal{V}^{(i)}_k(t)} \hat{p}^{(i)}_t(v) \log \hat{p}^{(i)}_t(v). \tag{10}$$

A larger entropy indicates a more diffuse next-token distribution and therefore greater uncertainty.

Maximum probability. As a complementary measure of local confidence, we record the probability assigned to the most likely next token:

$$P^{(i)}_t = \max_{v\in\mathcal{V}} p^{(i)}_t(v). \tag{11}$$

While entropy characterizes the overall concentration of the predictive distribution, maximum probability measures how strongly the model commits to its most likely continuation.

Final-layer hidden norm. To capture changes in the model's internal representation, we additionally use the ℓ_2 norm of the final-layer hidden state:

$$N^{(i)}_t = \left\| h^{(L,i)}_t \right\|_2. \tag{12}$$

The hidden-state norm captures the magnitude of the model's internal representation, providing a complementary view of its reasoning state beyond output-level uncertainty.

Segment-level signal matrix. For segment m with token index set $I^{(i)}_m$ and effective length $T^{(i)}_m$, the three token-level signals are stacked into

$$\mathbf{G}^{(i)}_m = \left[\left(E^{(i)}_t, P^{(i)}_t, N^{(i)}_t \right)^\top \right]_{t\in I^{(i)}_m} \in \mathbb{R}^{3\times T^{(i)}_m}. \tag{13}$$

The resulting matrix provides the token-wise signal observations used by SCUR for segment-level communication routing. SCUR aggregates these observations using permutation-invariant pooling, without explicitly modeling their temporal order.

Trace-level diagnostic statistics. We distinguish the token-level routing signals above from trace-level statistics used for analysis. In particular, output length is measured by the total number of generated reasoning tokens K_i, while hidden-norm variability can be summarized over a complete trace. These aggregate quantities characterize reasoning behavior in our signal analysis but are not additional token-wise channels in $\mathbf{G}_m^{(i)}$.

A.2 Token-wise Error Propagation

The examples in Figure 8 suggest that reasoning errors are often not isolated events. Once an intermediate reasoning state becomes incorrect, subsequent tokens may continue to amplify the erroneous context. We provide a more detailed analysis of error onset and propagation.

Locating the reasoning error. We first decompose each generated reasoning trajectory into a sequence of semantic reasoning steps. Specifically, a step is defined as the smallest contiguous span that expresses a coherent and independently verifiable reasoning operation, such as applying a formula or theorem, performing an algebraic transformation, deriving an intermediate result, or drawing a conclusion. Step boundaries are placed when the reasoning transitions to a new operation or derives a new intermediate claim; consecutive sentences that jointly support the same inference are kept within the same step. Initial step boundaries are generated by GPT-5.5 and manually verified. This semantic segmentation avoids imposing arbitrary sentence- or token-level boundaries that may split a single reasoning operation. A separately prompted GPT-5.5 judge then evaluates each step against the problem statement and reference solution and assigns a binary label indicating whether the inference made in that step is valid. We map the resulting step-level annotations back to the original token sequence by assigning every token within a step its corresponding label, yielding a token-level correctness sequence $y_t{}_{t=1}^{T}$, where $y_t \in \text{correct}, \text{error}$. We define the first-error position t_{err} as the first token of the earliest semantic step judged to contain an invalid inference. The identified position marks the onset of the first semantically detectable reasoning error. This enables reasoning-level judgments while enabling their alignment with token-level model signals.

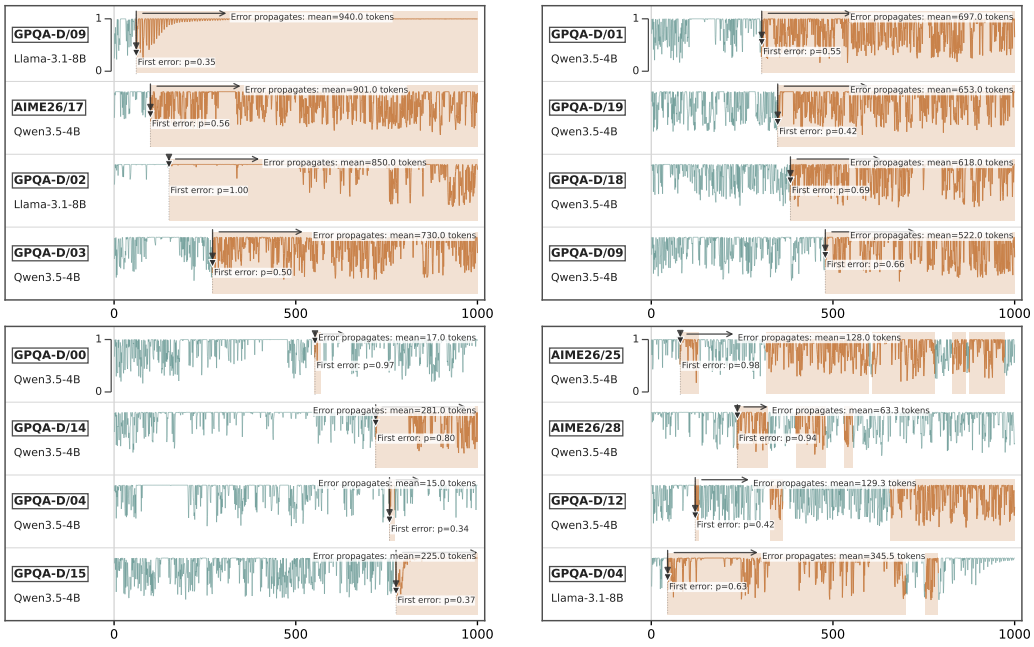


Figure 8: Token-level error dynamics across all incorrectly answered instances.

Signal dynamics around error onset. We analyze the token-level dynamics of maximum probability around t_{err}. Figure 8 shows substantial variation across reasoning traces, with the maximum probability at the first erroneous token ranging from approximately 0.34 to 1.00. Thus, an incorrect reasoning state may emerge even when the model remains locally confident, making instantaneous

confidence alone insufficient to assess overall reasoning correctness. This motivates examining complementary token-level signals aggregated over local segments for communication routing.

Propagation after the first error. The semantic-step annotations further allow us to quantify how erroneous reasoning persists after its first occurrence. After mapping the annotations to tokens, consecutive tokens labeled as erroneous form a set of contiguous error spans $\mathcal{B} = \{(s_j, e_j)\}_{j=1}^{M}$, where s_j and e_j denote the start and end token indices of the j-th error span. We measure the mean propagation length of a trajectory as the mean number of tokens contained in these erroneous spans:

$$L_{\text{prop}} = \frac{1}{M} \sum_{j=1}^{M} (e_j - s_j + 1). \tag{14}$$

L_{prop} measures the typical duration of a contiguous erroneous reasoning state after error onset. A representative case is `aime-00025`. The first substantive error appears at tokens 80–130, where the model repeatedly misapplies the power-of-a-point argument at point D and derives incorrect product and proportional relations. After several locally correct intervals, the reasoning becomes erroneous again over multiple later spans (316–598, 609–780, 830–866, and 875–971). Across these five error spans, the mean propagation length is 128 tokens, showing that reasoning errors can persist for extended periods and re-emerge after temporary recoveries. Together with the high-confidence first-error cases, these observations show that errors can persist across extended local spans, motivating segment-level communication routing based on local signal statistics.

A.3 Signal Statistics Across Evaluation Settings

Since individual signal values may provide limited information in isolation, we instead examine four trajectory-level statistics: mean entropy $\bar{E}$ (*Entropy*), mean maximum probability $\bar{P}$ (*Max Prob.*), the standard deviation of the final-layer hidden norm σ_N (*Hidden Std.*), and the number of generated tokens K_i (*Output Length*). We analyze their relationships with reasoning correctness and problem difficulty, and further examine whether different signal channels provide complementary information across datasets and model settings.

Relationship with reasoning correctness. Let $y_i \in \{0, 1\}$ denote whether reasoning trace i produces a correct final answer, and let z_i denote a trace-level statistic ($\bar{E}$, $\bar{P}$, σ_N, or K_i). We quantify its association with correctness using the point-biserial correlation

$$r_{\text{pb}} = \frac{\mu_1 - \mu_0}{\sigma_z} \sqrt{\frac{n_1 n_0}{(n_1 + n_0)^2}}, \tag{15}$$

where μ_1 and μ_0 are the mean values of z_i over correct ($y_i = 1$) and incorrect ($y_i = 0$) traces, respectively, σ_z is the standard deviation of z_i over all traces, and n_1 and n_0 are the corresponding sample counts.

Table 5 reports the correlations separately across datasets and backbone models. The results show that higher entropy is generally associated with lower reasoning correctness, whereas higher maximum probability is associated with higher correctness. Longer reasoning trajectories also tend to correlate negatively with correctness, suggesting that unsuccessful reasoning often involves more prolonged generation. In comparison, hidden-state variation exhibits a less uniform pattern across models and datasets. Importantly, the most prominent qualitative relationships, particularly those of entropy and maximum probability, remain consistent across different evaluation settings. This indicates that the signals capture properties associated with the underlying reasoning state rather than dataset-specific scale alone.

Relationship with problem difficulty. We further analyze signal behavior across the five difficulty levels of MATH. Because the difficulty labels form an ordinal scale, we measure their association with each signal using Spearman's rank correlation,

$$\rho_s = \text{corr}\left(\text{rank}(z_i), \text{rank}(d_i)\right), \tag{16}$$

where d_i denotes the difficulty level of problem i. Table 6 reports the resulting rank correlations for each dataset and backbone. We observe that entropy and output length generally increase with

Dataset	Model	Entropy	Max Prob.	Hidden Std.	Output Length
MATH #: 500	Qwen3.5-4B	-0.259^{***}	$+0.243^{***}$	-0.218^{***}	-0.554^{***}
	Qwen3-4B-Instruct-2507	-0.288^{***}	$+0.276^{***}$	-0.346^{***}	-0.489^{***}
	Gemma-3-4B-IT	-0.253^{***}	$+0.180^{***}$	$+0.224^{***}$	-0.545^{***}
	Phi-4-Mini-Instruct	-0.242^{***}	$+0.189^{***}$	$+0.039$	-0.268^{***}
	Ministral-3-3B-Instruct-2512	-0.292^{***}	$+0.289^{***}$	-0.215^{***}	-0.498^{***}
GPQA-D #: 198	Qwen3.5-4B	-0.273^{***}	$+0.275^{***}$	$+0.233^{***}$	-0.394^{***}
	Qwen3-4B-Instruct-2507	-0.376^{***}	$+0.364^{***}$	-0.267^{***}	-0.498^{***}
	Gemma-3-4B-IT	$+0.011$	-0.014	$+0.005$	-0.179^{*}
	Phi-4-Mini-Instruct	-0.151^{*}	$+0.139^{*}$	$+0.027$	-0.153^{*}
	Ministral-3-3B-Instruct-2512	-0.085	$+0.069$	$+0.043$	-0.111
MMLU-Pro #: 200	Qwen3.5-4B	-0.319^{***}	$+0.313^{***}$	-0.016	-0.349^{***}
	Qwen3-4B-Instruct-2507	-0.400^{***}	$+0.385^{***}$	-0.144^{*}	-0.329^{***}
	Gemma-3-4B-IT	-0.281^{***}	$+0.229^{**}$	-0.086	$+0.188^{**}$
	Phi-4-Mini-Instruct	-0.404^{***}	$+0.384^{***}$	$+0.034$	-0.188^{**}
	Ministral-3-3B-Instruct-2512	-0.398^{***}	$+0.365^{***}$	$+0.236^{***}$	-0.318^{***}

Pearson correlation (two-sided): $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

Table 5: Relationship between signals and reasoning correctness across datasets and backbones. We report point-biserial correlation r_{pb} between trajectory-level signal statistic and answer correctness.

Dataset	**Model**	**Entropy**	**Max Prob.**	**Hidden Std.**	**Output Length**
MATH #: 500	Qwen3.5-4B	$+0.223^{***}$	-0.213^{***}	$+0.175^{***}$	$+0.485^{***}$
	Qwen3-4B-Instruct-2507	$+0.360^{***}$	-0.354^{***}	$+0.414^{***}$	$+0.503^{***}$
	Gemma-3-4B-IT	$+0.051$	-0.009	-0.190^{***}	$+0.492^{***}$
	Phi-4-Mini-Instruct	-0.095^{*}	$+0.121^{**}$	$+0.101^{*}$	$+0.505^{***}$
	Ministral-3-3B-Instruct-2512	$+0.324^{***}$	-0.325^{***}	$+0.099^{*}$	$+0.531^{***}$
Omni-MATH #: 500	Qwen3.5-4B	$+0.306^{***}$	-0.296^{***}	$+0.177^{***}$	$+0.629^{***}$
	Qwen3-4B-Instruct-2507	$+0.429^{***}$	-0.416^{***}	$+0.495^{***}$	$+0.681^{***}$
	Gemma-3-4B-IT	$+0.174^{***}$	-0.130^{**}	-0.236^{***}	$+0.573^{***}$
	Phi-4-Mini-Instruct	$+0.124^{**}$	-0.061	$+0.244^{***}$	$+0.569^{***}$
	Ministral-3-3B-Instruct-2512	$+0.317^{***}$	-0.314^{***}	$+0.005$	$+0.637^{***}$
ARC #: 400	Qwen3.5-4B	$+0.251^{***}$	-0.212^{***}	$+0.120^{*}$	$+0.189^{***}$
	Qwen3-4B-Instruct-2507	$+0.195^{***}$	-0.189^{***}	$+0.118^{*}$	$+0.261^{***}$
	Gemma-3-4B-IT	$+0.168^{***}$	-0.166^{***}	-0.093	$+0.081$
	Phi-4-Mini-Instruct	-0.007	-0.012	$+0.049$	$+0.198^{***}$
	Ministral-3-3B-Instruct-2512	$+0.105^{*}$	-0.062	$+0.121^{*}$	$+0.122^{*}$

Spearman correlation (two-sided): $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

Table 6: Relationship between model signals and problem difficulty across datasets. We report Spearman's rank correlation ρ_s between each trajectory-level signal statistic and problem difficulty. For MATH, 500 examples have recoverable difficulty levels, with Level 1–5 counts of 38/83/131/120/128. For Omni-MATH, 100 examples are sampled from each of Levels 1–5. For ARC, Easy and Challenge contain 200 examples each.

problem difficulty, while maximum probability tends to decrease; hidden-state statistics exhibit more model-dependent trends. These trends suggest that the signals respond systematically to changes in reasoning complexity rather than remaining approximately stationary across problems.

Across datasets and backbones, trajectory-level signals systematically vary with both problem difficulty and reasoning correctness. In particular, harder problems are generally associated with higher entropy, lower maximum probability, and longer generations, while the corresponding directions reverse with answer correctness. Hidden-state statistics exhibit more model-dependent associations with reasoning outcomes. These results establish that the extracted signals provide informative and complementary views of the evolving reasoning state, supporting their use as lightweight inputs to SCUR for segment-level communication routing.

B SCUR ARCHITECTURE AND TRAINING DETAILS

B.1 SCUR ARCHITECTURE

We detail the signal-set encoding, content representation, and channel-utility prediction of the Signal-Conditioned Utility Router (SCUR) introduced in Section 3.1. SCUR predicts channel-specific downstream utilities rather than a softmax distribution over preferred-channel classes.

Non-overlapping segmentation. For sender trace $r^{(i)}_{1:K_i}$, the number of segments is $M_i = \lceil K_i/W \rceil$, with $W = 200$ tokens. Segment boundaries are

$$s_m = 1 + (m-1)W, \qquad e_m = \min(mW, K_i), \qquad I^{(i)}_m = \{s_m, \dots, e_m\}. \tag{17}$$

The effective length is $T^{(i)}_m = e_m - s_m + 1$. Only the available token positions are included in a segment's aggregation. The three scalar signal channels are the entropy, maximum probability, and final-layer hidden-state norm specified in Appendix A.1. Preprocessing statistics for these scalar channels are determined on the training data and frozen for deployment.

Signal-set encoding. A shared token-wise MLP with parameters θ_{enc} produces $f^{(i)}_t = \phi_{\theta_{\text{enc}}}(E^{(i)}_t, P^{(i)}_t, N^{(i)}_t)$. The pooling operation in Equation 1 is

$$z^{(i)}_m = \left[\text{mean}_{t \in I^{(i)}_m} f^{(i)}_t; \max_{t \in I^{(i)}_m} f^{(i)}_t; \text{std}_{t \in I^{(i)}_m} f^{(i)}_t\right], \tag{18}$$

where all three statistics are computed element-wise. This representation is permutation-invariant with respect to the token-wise signal observations. The set-based encoder avoids assuming that the order of these observations is required for routing. The development experiments reported no consistent benefit from temporal encoding relative to shuffled inputs. The earlier three-class comparisons are documented separately in Appendix B.6, rather than treated as measurements of the final SCUR utility predictor.

Content representation. The content branch averages the sender's final-layer hidden states over the valid token positions in each segment, yielding $\bar{h}^{(i)}_m$. Layer normalization followed by a learned linear projection produces $\tilde{h}^{(i)}_m$. This representation is used for routing and is distinct from the transmitted latent payload.

Prediction head. The prediction head $f_{\theta_{\text{pred}}}$ maps $[z^{(i)}_m; \tilde{h}^{(i)}_m]$ to three real-valued logits $\xi^{(i)}_m = f_{\theta_{\text{pred}}}\left([z^{(i)}_m; \tilde{h}^{(i)}_m]\right) \in \mathbb{R}^3$, and applies a separate sigmoid to each. The resulting utilities are supervised by channel-specific receiver correctness, not by a single preferred-channel class, and need not sum to one. The trainable SCUR parameters are $\theta = (\theta_{\text{enc}}, \theta_{\text{cont}}, \theta_{\text{pred}})$, where θ_{cont} denotes the trainable parameters of the content branch. All three components are optimized jointly under the rescue-weighted utility objective in Equation 7.

B.2 MIXED-FORMAT MESSAGE CONSTRUCTION

Channel realization. For each segment, the selected action determines its transmitted content:

$$c^{(i)}_m = \begin{cases} \text{Detok}(r^{(i)}_{I^{(i)}_m}), & a^{(i)}_m = \text{TEXT}, \\ e^{(i)}_m, & a^{(i)}_m = \text{LATENT}, \\ \varnothing, & a^{(i)}_m = \text{SKIP}. \end{cases} \tag{19}$$

Here, Detok converts token IDs to text. For LATENT, let $\text{Emb}^{(i)}(v)$ denote the sender's token embedding for v. Using the top-k support $\mathcal{V}^{(i)}_k(t)$ and renormalized probabilities $\hat{p}^{(i)}_t$ defined in Appendix A.1, the soft embedding at step t is

$$u_t^{(i)} = \sum_{v \in \mathcal{V}_k^{(i)}(t)} \hat{p}_t^{(i)}(v)\, \mathrm{Emb}^{(i)}(v). \tag{20}$$

The segment payload is the ordered sequence

$$e_m^{(i)} = \left[u_t^{(i)}\right]_{t \in I_m^{(i)}} \in \mathbb{R}^{T_m^{(i)} \times d_e}, \tag{21}$$

where d_e is the token-embedding dimension. For agents sharing the same backbone, the soft embeddings are supplied in the receiver's input-embedding space through `inputs_embeds`. The vocabulary support size k is distinct from the background-mask retention count k_{keep}.

Message assembly. Retained text and latent components are collected in their original segment order. The corresponding buffers and routing record are

$$\begin{aligned} B_i^{\mathrm{text}} &= \mathrm{OrderConcat}\Big(\{c_m^{(i)} \mid a_m^{(i)} = \mathrm{TEXT}\}\Big), \\ B_i^{\mathrm{lat}} &= \mathrm{OrderConcat}\Big(\{c_m^{(i)} \mid a_m^{(i)} = \mathrm{LATENT}\}\Big), \\ \mathcal{R}_i &= \Big\{(m, I_m^{(i)}, a_m^{(i)})\Big\}_{m=1}^{M_i}. \end{aligned} \tag{22}$$

The downstream message is

$$\mathcal{M}_i = (\hat{a}_i, B_i^{\mathrm{text}}, B_i^{\mathrm{lat}}, \mathcal{R}_i). \tag{23}$$

The routing record maintains the correspondence between transmitted components and positions in the original reasoning trace. It is message metadata, not an additional input branch of SCUR.

B.3 SCUR Training Details

Label pool. The supervision pool contains 240 mathematics problems and 200 science problems from GPQA, all solvable by the baseline. Their traces yield 55,260 segment–mask examples in total. For each example, we record the target segment's signal matrix G_j, mean final-layer hidden state $\bar{h}_j$, and three binary outcomes $(R_{j,\mathrm{TEXT}}, R_{j,\mathrm{LATENT}}, R_{j,\mathrm{SKIP}})$. These are repeated counterfactual observations under masks, not 55,260 necessarily distinct trace segments. Mathematics and science examples are pooled to train one shared SCUR.

Counterfactual label construction. For a trace of M segments, a mask is sampled from

$$\mathcal{B}_{k_{\mathrm{keep}}} = \left\{ b \in \{0,1\}^M : \sum_{n=1}^{M} b_n = k_{\mathrm{keep}} \right\}, \qquad k_{\mathrm{keep}} \in \{1,2,3\}. \tag{24}$$

The selected count must be feasible for the trace. Background segments with $b_n = 1$ are retained as TEXT; the others are assigned to SKIP. Equation 5 overrides only the target segment's assignment for the three replays. The count constraint applies to the background mask before the target intervention, not necessarily to the assembled message after that intervention. Multiple masks for the same segment generate separate supervision records, with identical segment features but potentially different outcome vectors. A receiver A_{rec} using the same 4B backbone as the sender evaluates each counterfactual message. It is initialized with the assistant prefill `\boxed{` to generate the answer before any explanation. In the example-index notation used by the training loss,

$$\begin{aligned} \hat{y}_{j,c} &= \mathrm{Extract}_{\mathrm{ans}}(A_{\mathrm{rec}}(q_j, \mathcal{M}_{j,c})), \\ R_{j,c} &= \mathrm{Eval}(\hat{y}_{j,c}, y_j) \in \{0,1\}. \end{aligned} \tag{25}$$

The evaluator scores the initial answer against the reference answer y_j; completion of any subsequent explanation is not a prerequisite for labeling. Each record retains the outcomes for all three

channels, rather than reducing them to a single preferred-channel class. The background mask is used to generate supervision and is not an additional SCUR input.

Rescue-weighted BCE. For a predicted utility u and binary outcome r, the loss term is

$$\mathrm{BCE}(u, r) = -r \log u - (1 - r) \log(1 - u). \tag{26}$$

Equation 7 uses $\lambda = 4$, so examples with $R_{j,\mathrm{TEXT}} \neq R_{j,\mathrm{LATENT}}$ receive weight five and the others receive weight one. The same example weight applies to all three channel terms. This includes both directions of text–latent disagreement, rather than only latent rescue events.

Frozen operating point. The threshold in Equation 4 is selected from training-score quantiles with a target communication rate of 50%. SCUR parameters θ, signal-normalization statistics, and the threshold are fixed before final evaluation; evaluation data are not used to select these quantities. The same trained SCUR and operating point are reused in the OS and GV workflows. The target is specified on the training distribution and is not an evaluation-time communication-rate constraint.

B.4 Multi-Agent Orchestration and Deployment

Communication statistics. For each channel $c \in \mathcal{A}_{\mathrm{comm}}$, its segment count, channel proportion, and the transmitted text length are

$$C_i^c = \sum_{m=1}^{M_i} \mathbf{1}[a_m^{(i)} = c], \qquad \rho_i^c = \frac{C_i^c}{M_i}, \qquad K_i^{\mathrm{text}} = \sum_{m=1}^{M_i} T_m^{(i)} \mathbf{1}[a_m^{(i)} = \mathrm{TEXT}]. \tag{27}$$

Across interaction steps, these quantities contribute to the cumulative token usage and channel proportions summarized by the orchestration state s_ℓ in Equation 8. The state additionally includes task progress, repetition/oscillation and invalid-feedback indicators, and statistics of $E_t^{(i)}$ and $P_t^{(i)}$. Compact records track skipped action outcomes, subgoal completion, and visited locations. These orchestration signals and records are separate from the segment-level inputs used by SCUR. If $B_i^{\mathrm{lat}} \neq \varnothing$, the next recipient must share the sender's backbone and support soft-embedding inputs.

Orchestrator–Subagent (OS). Multiple generators solve the same query in parallel using different random seeds. Each sender trace is independently processed by the frozen SCUR. If the candidate answers agree, their consensus answer is returned without a judge call. Otherwise, the routed messages are supplied to a judge for final adjudication.

Generator–Verifier (GV). A single generator produces a candidate condition and reasoning trace. Its routed trace is passed to a verifier for adjudication. The SCUR parameters, scalar-signal normalization statistics, and decision threshold are shared with the OS configuration. These two configurations fix the receiver topology.

B.5 Communication-Channel Analyses and Receiver-Input Probes

The following analyses characterize the three communication channels. The receiver-input probes examine the probability-weighted token embeddings defined in Appendix B.2.

Utility of skipping redundant communication. We first examine whether all intermediate reasoning content needs to be communicated. As a controlled sanity check, we randomly retain only a fraction of the sender's message while keeping the full-message setting as the normalized 100% baseline. As shown in Table 7, substantial redundancy exists in the communication stream. On GPQA, retaining only 50% of the message preserves 99.2% of the full-message performance ($\Delta = -0.8$ pt), while MATH retains 98.0% ($\Delta = -2.0$ pt). Even at a 30% retention ratio, performance remains 93.6% and 97.2% of the baseline on MATH and GPQA, respectively. These results are obtained with *random* removal rather than signal-guided selection, suggesting considerable headroom for a learned SKIP policy to identify and suppress redundant segments more selectively.

Retained	MATH	GPQA-D
5%	56.7%	77.2%
10%	69.0%	89.0%
20%	87.7%	92.5%
30%	93.6%	97.2%
40%	94.7%	96.9%
50%	98.0%	99.2%

Table 7: Effect of random reasoning retention. Performance is normalized by the full-message baseline (100%).

Case study: empirical validation of latent communication. We use controlled probes to establish when latent communication carries genuine information and where its fidelity breaks down. Directly injecting final-layer hidden states causes generation collapse despite matched dimensionality, motivating our use of probability-weighted mixtures of token embeddings as receiver-side latent inputs. A context-leakage control shows that apparent success can arise when the original question remains visible; after removing it, a short latent message ($m = 20$) still enables the receiver to recover key task facts, providing direct evidence of semantic transfer. With a much longer message ($m = 1000$), the latent sequence remains broadly aligned with the sender but progressively accumulates local omissions, structural errors, and lexical drift. These results establish a practical boundary: latent communication supports semantic transfer through soft embeddings, while fine-grained fidelity degrades over long horizons.

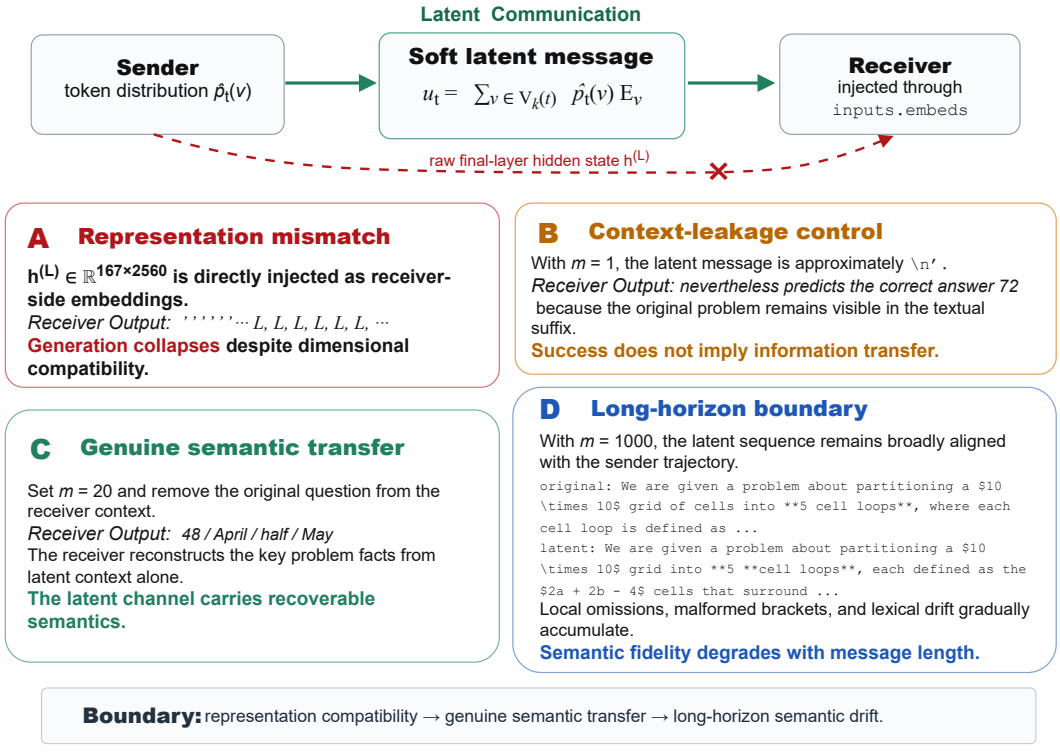


Figure 9: Empirical validation of latent communication.

B.6 Preliminary Three-Class Routing Analysis

This subsection records the earlier three-class routing study, whose labels, classifier outputs, and balanced metrics differ from the final SCUR utility predictor and the 55,260-example counterfactual dataset in Appendix B.3. Results below apply only to this preliminary setting.

Preliminary classification data. In this earlier setting, segment-level routing labels are obtained by counterfactually evaluating each segment under TEXT, LATENT, and SKIP. We split by problem so all segments from a reasoning trace stay in the same split, preventing trace-level leakage between training and evaluation. Table 8 summarizes these data.

Dataset	# Problems	# Segments
MATH	240	7,246
GPQA-Main†	200	7,252
Total	**440**	**14,498**

Table 8: Data for the preliminary three-class routing study, not the final SCUR utility predictor. Segments have TEXT, LATENT, or SKIP labels. †GPQA-Main training instances exclude GPQA-D

Signal-set representation of reasoning states. For each segment m, we retain the token-wise entropy, maximum probability, and final-layer hidden-norm signals and combine them into a multi-signal vector at each token position. A shared MLP first maps each token-level vector into a learned representation, after which a permutation-invariant pooling operator aggregates the representations into a fixed-dimensional segment embedding:

$$\mathbf{z}_m^{(i)} = \mathrm{Pool}_t \left(\phi\Big(\mathbf{G}_{m,t}^{(i)} \Big) \right) \in \mathbb{R}^{32}, \tag{28}$$

where $\phi(\cdot)$ denotes the shared token-wise MLP and $\mathbf{G}_{m,t}^{(i)}$ is the multi-signal observation at token t. The resulting representation preserves fine-grained information from the individual token-level signals while jointly capturing complementary evidence across signal channels, without explicitly modeling their temporal order.

Controlled analysis of signal aggregation. The preliminary classification study asks whether combining complementary signals improves routing over individual signals and whether explicit token-order modeling adds value beyond segment-level signal aggregation.

We construct controlled variants that progressively change how the token-level signals are represented. STAT-MLP uses segment-level statistics from a single signal—entropy, maximum probability, or hidden-state norm—to isolate the predictive value of each channel. STATISTIC-MLP combines summary statistics from all signals but discards the individual token-level signal vectors. TCN-SHUFFLED retains the same multi-signal inputs as the sequence model while randomly permuting token positions within each segment, thereby destroying temporal order. TCN W/O DILATION preserves temporal order but replaces dilated convolutions with standard local convolutions, while FULL TCN explicitly models the ordered multi-signal sequence. Finally, our SIGNALSET-MLP retains the token-level multi-signal vectors but aggregates them as an unordered set, preserving fine-grained signal information without imposing an explicit temporal model.

Because the routing labels are substantially imbalanced, overall accuracy can be dominated by the majority class. We therefore report balanced accuracy (Bal-Acc), Macro-F1, and the class-wise F1 scores for TEXT, LATENT, and SKIP. Bal-Acc assigns equal weight to the recall of each routing class, while Macro-F1 equally averages the three class-wise F1 scores.

Signal-set aggregation. We further examine whether effective routing requires explicitly modeling the temporal order of token-level signals. In addition to single-signal and summary-statistic MLP baselines, we compare against several sequence-aware TCN variants. SIGNALSET-MLP instead treats the token-level signal vectors within each segment as a set and aggregates their joint distribution without assuming a particular temporal ordering. As shown in Tables 9 and 10, SIGNALSET-MLP achieves the best balanced accuracy and balanced F1 on both MATH and GPQA. On MATH, it reaches 0.416 Bal-Acc and 0.412 Bal-F1, outperforming both STATISTIC-MLP (0.403/0.401) and the full TCN (0.374/0.367). The same trend holds on GPQA, where it obtains 0.385/0.375 compared with 0.376/0.369 for STATISTIC-MLP and 0.353/0.330 for the full TCN. These results suggest that the routing decision is driven primarily by the complementary distribution of multiple uncertainty signals, while explicit temporal modeling is not necessary for capturing this information.

Table 9: Analysis of preliminary three-class routers on MATH. STAT-MLP variants use a single signal, STATISTIC-MLP aggregates multiple signal statistics, and the TCN variants explicitly model token-level temporal structure. SIGNALSET-MLP jointly aggregates the token-level multi-signal set without relying on temporal order. Results are mean±std over three seeds under 5-fold evaluation.

Router Variant	Temporal	Multi-Signal	Bal-Acc	Bal-F1	Text F1	Latent F1	Skip F1
Random	–	–	0.332 ± 0.003	0.331	0.328	0.334	0.330
Stat-MLP (Entropy)	No	No	0.410 ± 0.008	0.405	0.434	0.333	0.448
Stat-MLP (MaxProb)	No	No	0.391 ± 0.009	0.387	0.370	0.349	0.443
Stat-MLP (Hidden)	No	No	0.392 ± 0.005	0.382	0.295	**0.396**	0.456
Statistic-MLP	No	Yes	0.403 ± 0.007	0.401	0.387	0.388	0.429
TCN-Shuffled	No	Yes	0.350 ± 0.004	0.336	0.224	0.384	0.401
TCN w/o Dilation	Yes	Yes	0.348 ± 0.012	0.339	0.260	0.357	0.400
Full TCN	Yes	Yes	0.374 ± 0.009	0.367	0.323	0.379	0.398
SignalSet-MLP	**No**	**Yes**	$\mathbf{0.416 \pm 0.002}$	**0.412**	**0.439**	0.340	**0.458**

Table 10: Analysis of preliminary three-class routers on GPQA-Diamond. SIGNALSET-MLP uses the same multi-signal input as the temporal baselines but aggregates token-level signal vectors without explicitly modeling their order. Results are mean±std over three seeds under 5-fold evaluation.

Router Variant	Temporal	Multi-Signal	Bal-Acc	Bal-F1	Text F1	Latent F1	Skip F1
Random	–	–	0.339 ± 0.015	0.337	0.342	0.329	0.338
Stat-MLP (Entropy)	No	No	0.382 ± 0.017	0.372	0.267	**0.427**	0.423
Stat-MLP (MaxProb)	No	No	0.377 ± 0.009	0.362	0.229	0.416	**0.440**
Stat-MLP (Hidden)	No	No	0.373 ± 0.007	0.361	0.245	0.413	0.426
Statistic-MLP	No	Yes	0.376 ± 0.009	0.369	0.284	0.403	0.420
TCN-Shuffled	No	Yes	0.363 ± 0.012	0.346	0.232	0.402	0.405
TCN w/o Dilation	Yes	Yes	0.373 ± 0.022	0.363	**0.318**	0.407	0.364
Full TCN	Yes	Yes	0.353 ± 0.002	0.330	0.187	0.387	0.417
SignalSet-MLP	**No**	**Yes**	$\mathbf{0.385 \pm 0.004}$	**0.375**	0.272	0.418	0.435

Complementary signal channels. The single-signal variants further reveal that entropy, maximum probability, and hidden-state statistics capture different aspects of routing uncertainty. For example, entropy is the strongest individual baseline in balanced F1 on both MATH and GPQA, while hidden-state and maximum-probability signals obtain the strongest performance for particular routing classes. However, no single signal consistently dominates across all channels. Jointly aggregating the signals with SIGNALSET-MLP yields the highest overall Bal-Acc and Bal-F1 on both datasets, supporting multi-signal aggregation within this earlier classification setting. This comparison does not establish the incremental value of each signal for the final SCUR utility predictor.

Is temporal modeling necessary? Explicitly preserving token order does not improve routing performance. The full TCN reaches only 0.367 Bal-F1 on MATH and 0.330 on GPQA, both below SIGNALSET-MLP at 0.412 and 0.375, respectively. Removing dilation or shuffling the sequence likewise provides no consistent advantage. This suggests that, at the segment scale considered here, the preliminary classifier mainly relies on the distribution and co-occurrence of uncertainty signals rather than their exact token-by-token ordering. These findings concern only the preliminary classifier; SCUR is trained using the counterfactual objective in Section 3.2.

C EXPERIMENT SETUPS

C.1 EVALUATION DATASETS

A meaningful evaluation of multi-agent systems critically depends on the choice of tasks. Unlike prior work that predominantly focuses on relatively simple reasoning tasks, we additionally consider more complex settings that require sustained interaction and adaptive coordination. We introduce all datasets used in our experiments in two categories. **Foundation reasoning tasks** consist of self-contained reasoning benchmarks with a relatively bounded inference horizon, and have been widely used to evaluate multi-agent collaboration (Zhang et al., 2025a; Lei et al., 2024; Zhou et al., 2026). **Long-horizon decision-making tasks** involve sequential decisions over extended trajectories, where actions continually affect subsequent states and outcomes, placing greater demands on adaptive coordination (Du et al., 2026; Li et al., 2024; Shao et al., 2026). The task-level training, evaluation, and analysis splits used throughout our experiments are summarized in Table 11.

Foundation reasoning tasks.

- **HMMT** (Dekoninck et al., 2026) consists of 33 challenging competition-style mathematics problems that require multi-step symbolic reasoning across topics such as algebra, geometry, combinatorics, and number theory. Compared with routine mathematical questions, these problems typically involve longer intermediate derivations and stronger dependencies between reasoning steps. We use HMMT to evaluate whether communication routing can preserve critical intermediate reasoning while avoiding unnecessary transmission of locally redundant steps.
- **AIME2026** (Dekoninck et al., 2026) contains 30 problems from the 2026 American Invitational Mathematics Examination. Each problem has a uniquely determined numerical answer but often admits a long and non-trivial derivation involving several interacting mathematical observations. This setting is particularly suitable for studying reasoning-state communication: an early incorrect intermediate step can substantially alter the remainder of the trajectory.
- **MATH** (Hendrycks et al., 2021) contains 12,500 competition mathematics problems across seven domains and five annotated difficulty levels. Importantly for our analysis, MATH provides five annotated difficulty levels, ranging from relatively direct problems to substantially more involved reasoning tasks. We therefore use MATH not only as an evaluation benchmark, but also as a controlled environment for examining how signal statistics and communication decisions change with problem difficulty. This enables us to test whether the learned router adjusts its use of TEXT, LATENT, and SKIP as reasoning complexity increases.
- **Omni-MATH** (Gao et al., 2024) is a challenging Olympiad-level mathematics benchmark comprising 4,428 competition problems across 33 mathematical subdomains and 10 difficulty levels. Compared with MATH, it places greater emphasis on difficult competition-level reasoning, often requiring longer derivations and stronger dependencies between intermediate steps. We use Omni-MATH as a complementary setting to examine whether the reasoning signals and communication decisions observed on MATH remain consistent under more demanding problems, and whether the learned router adapts its use of TEXT, LATENT, and SKIP as reasoning complexity increases.
- **GPQA-Diamond (GPQA-D)** (Rein et al., 2023) is the most challenging GPQA subset, comprising graduate-level multiple-choice questions in biology, physics, and chemistry that require both specialized domain knowledge and scientific reasoning. This distinction is useful for our evaluation because correct communication may require preserving precise factual or conceptual evidence rather than only symbolic derivations. We use GPQA-D to examine whether the routing policy generalizes from mathematical reasoning trajectories to knowledge-intensive scientific reasoning.
- **MMLU-Pro** (Wang et al., 2024) is a challenging multi-domain benchmark covering a broad range of academic subjects and requiring more complex reasoning than standard multiple-choice knowledge benchmarks. We use MMLU-Pro for signal analysis, examining whether the relationships between trajectory-level signals and answer correctness generalize across diverse domains.
- **ARC** (Clark et al., 2018) is a multiple-choice science reasoning benchmark drawn from grade-school science questions and divided into Easy and Challenge subsets. Compared with competition mathematics, ARC places greater emphasis on combining scientific knowledge with short multi-step reasoning rather than extended symbolic derivations. We use ARC to examine whether the learned routing policy generalizes across difficulty levels and from mathematical reasoning to knowledge-intensive scientific reasoning.

Long-Horizon Decision-Making Tasks.

- **ALFWorld** (Shridhar et al., 2020) evaluates sequential decision-making in text-based embodied household environments. An agent must interpret the current environment, plan actions, execute them, and repeatedly revise its state based on environmental feedback. In contrast to the largely self-contained reasoning problems above, ALFWorld introduces a long interaction horizon in which the usefulness of information changes over time. We use it as a complementary setting for evaluating the full hierarchical design of RELAY: segment-level communication routing determines what reasoning information is exposed and in which representation, while agent-level orchestration determines which agent should act next or whether the interaction should terminate.
- **WebShop** (Yao et al., 2022) evaluates interactive web-based shopping tasks in which agents must interpret user requirements, navigate a large product space, inspect relevant attributes, and execute a sequence of actions to identify and purchase the desired item. Unlike mathematical or scientific reasoning, WebShop involves long-horizon interaction with dynamically evolving observations and action histories. Intermediate states contain both high-level semantic reasoning, such as preference matching and search strategy, and task-critical information, such as product attributes and navigation states. While the latter may need to be preserved explicitly for reliable decision making, redundant observations or stabilized intermediate reasoning can potentially be compressed or omitted. We therefore use WebShop to evaluate whether RELAY can adapt its communication representation throughout a long-horizon, interaction-driven decision trajectory.

Benchmark	Train	Test	Analysis
HMMT	–	33	–
AIME2026	–	30	–
MATH	240	1250	500
GPQA-D	–	198	198
GPQA-Main†	200	–	–
Omni-MATH	–	–	500
ARC	–	–	400
MMLU-Pro	–	–	200
ALFWorld	200	240	–
WebShop	200	240	–
Total	**840**	**1,991**	**1,798**

Table 11: **Task-level dataset splits.** Training and test sets are disjoint. The Analysis column reports instances used for statistical analysis of reasoning signals and behavior.†GPQA-Main training instances exclude GPQA-D.

C.2 COMPARED BASELINES

In this section, we provide a detailed description of the configurations for baseline methods:

1. **CoT.** We use standard Chain-of-Thought (CoT) prompting (Wei et al., 2022), which elicits intermediate reasoning steps before producing the final answer.
2. **Self-Refine.** Self-Refine (Madaan et al., 2023) iteratively improves an initial response through self-generated feedback and revision within a single agent.
3. **TextMAS (GV).** A text-based generator–verifier baseline (Lei et al., 2024) in which a generator produces intermediate reasoning and a verifier checks it, with all communication conducted through natural-language messages.
4. **TextMAS (OS).** A text-based orchestrator–subagent baseline (Zhuge et al., 2024) in which an orchestrator coordinates multiple subagents and aggregates their textual responses to produce the final answer.
5. **LatentMAS.** LatentMAS (Zou et al., 2025) replaces explicit textual communication with continuous latent representations exchanged between agents.
6. **SDE.** SDE (Tang et al., 2025) enables agents to exchange intermediate representations through a latent communication mechanism for collaborative reasoning.
7. **CIPHER.** CIPHER (Pham et al., 2024) performs multi-agent reasoning through continuous hidden-state communication between participating agents.

8. **Interlat.** Interlat (Du et al., 2026) enables agents to communicate entirely in latent space by directly transmitting continuous hidden-state representations between agents.
9. **HyLaT.** HyLaT (Mou et al., 2026) adopts a hybrid latent–text communication protocol, using latent representations for cognitive information while retaining text for critical explicit signals.

Baseline Adaptation. For fair comparison, we adapt the latent-communication baselines above to our common multi-agent evaluation setup. Specifically, we retain the core communication mechanism of each method while aligning the agent backbone, interaction protocol, decoding configuration, and evaluation procedure with our setup wherever applicable. This controls for differences unrelated to the communication mechanism itself. However, these adaptations may deviate from the baselines' originally intended operating conditions and thus potentially underestimate their achievable performance.

C.3 Evaluation Protocol

Evaluation protocol. All methods are evaluated on exactly the same instances for each benchmark. Comparisons are conducted separately for each backbone and MAS setting. For RELAY, the generator–verifier or orchestrator–subagent structure is fixed and agent-level orchestration routing is disabled. Thus, only the segment-level choice among TEXT, LATENT, and SKIP is adaptive. For long-horizon tasks, all methods use the same episode-level round limit.

Decoding settings. Generator agents use sampling with temperature 0.7, top-$p = 1.0$, and top-$k = 0$. Verifier, judge, and answer-normalization calls use greedy decoding with temperature 0. The Max Tokens is fixed across methods within each setting; the main experiments use caps of $60{,}000$, $70{,}000$, and $100{,}000$ for AIME2026, GPQA-D, and HMMT, respectively. Unless otherwise specified, results are averaged over three complete evaluation runs. When multiple candidate generators are used, for a run with base seed s, candidate j uses seed $s + j - 1$.

Answer evaluation. We use task-specific deterministic evaluation. For mathematical reasoning tasks, such as AIME2026, we extract and lightly normalize the final answer before exact matching with the reference. For question-answering tasks, such as GPQA-D, we extract the final option label and perform exact matching under a fixed option order shared across all methods. For long-horizon decision-making tasks, such as ALFWorld, we follow the benchmark-specific evaluator and success criterion. Unparsable responses and persistent generation failures are counted as incorrect.

Metrics. Accuracy is the fraction of evaluated instances whose extracted answer matches the reference answer: $\mathrm{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[\hat{y}_i = y_i]$. Token consumption is the average total number of discrete input and output tokens per problem, summed over all agent calls and communication rounds: $\mathrm{Token} = \frac{1}{N}\sum_{i=1}^{N}\sum_{s\in\mathcal{S}_i}\left(n_{i,s}^{\mathrm{in}} + n_{i,s}^{\mathrm{out}}\right)$, where $\mathcal{S}_i$ denotes all LLM calls for instance i.

Reporting rule for Δ vs. Best Baseline. For each backbone and benchmark, we select the best baseline across all evaluated baselines as the method with the highest accuracy, breaking ties by lower token consumption. We similarly select the best result of our method by accuracy, with ties broken by lower token consumption. We then report the relative accuracy change as $\Delta_{\mathrm{Acc}} = \frac{\mathrm{Acc}_{\mathrm{ours}} - \mathrm{Acc}_{\mathrm{base}}}{\mathrm{Acc}_{\mathrm{base}}} \times 100\%$, and the relative token change as $\Delta_{\mathrm{Token}} = \frac{\mathrm{Token}_{\mathrm{base}} - \mathrm{Token}_{\mathrm{ours}}}{\mathrm{Token}_{\mathrm{base}}} \times 100\%$. For both metrics, improvement follows the directions indicated in the table: higher accuracy (Acc. $\uparrow$) and lower token consumption (Token $\downarrow$) are better.

C.4 Implementation Details

We use the publicly released `Qwen/Qwen3.5-4B` and `Qwen/Qwen3.5-9B` checkpoints for the main and long-horizon experiments, and `meta-llama/Llama-3.1-8B-Instruct` and `Qwen/Qwen3-1.7B` for MATH. All backbone parameters remain frozen. Inference is implemented with PyTorch and Hugging Face Transformers, using `bfloat16` for the backbones and `float32` for signal processing and routing. Experiments run on $8\times$ NVIDIA H20 GPUs, with independent evaluation shards distributed across GPUs and one model replica per GPU.

D ADDITIONAL EXPERIMENTS

D.1 COMMUNICATION CHANNEL DISTRIBUTION ACROSS MAS SETTINGS

As shown in Table 12, TEXT and SKIP account for the largest proportions, ranging from 43.3–49.9% and 41.8–44.4%, respectively, while LATENT accounts for 7.8–13.2%. The channel distributions are similar between Generator–Verifier and Orchestrator–Subagent, with GPQA-D showing a higher LATENT proportion than AIME2026.

MAS Setting	Dataset	Text (%)	Latent (%)	Skip (%)
Generator–Verifier	AIME2026	49.9	7.8	42.4
Orchestrator–Subagent	AIME2026	49.7	8.5	41.8
Generator–Verifier	GPQA-D	43.3	12.3	44.4
Orchestrator–Subagent	GPQA-D	44.7	13.2	42.0

Table 12: Communication channel usage (%) with Qwen3.5-4B across two MAS settings.

D.2 RELAY ACROSS BACKBONE MODELS

Table 13 further evaluates our method with Qwen3-1.7B on MATH. $\text{RELAY}_{\text{Core}}$ achieves the highest overall accuracy (94.0%) while using fewer tokens than the strongest text-based baseline, and remains competitive across all five difficulty levels. Although the per-level accuracy–efficiency pattern varies across backbones, the overall results support the effectiveness of adaptive routing.

Method	Overall		Level 1		Level 2		Level 3		Level 4		Level 5	
	Acc↑	Token (k)↓	Acc↑	Token (k)↓	Acc↑	Token (k)↓	Acc↑	Token (k)↓	Acc↑	Token (k)↓	Acc↑	Token (k)↓
CoT	72.9	1.8	88.0	0.9	76.0	1.2	74.0	1.6	64.0	2.3	62.7	2.9
TextMAS	91.2	7.8	98.7	4.3	95.3	5.9	93.3	7.6	86.0	9.7	82.7	11.7
LatentMAS	90.0	3.2	96.7	1.3	93.3	1.7	91.3	2.4	88.7	5.2	80.0	5.6
SDE	82.4	2.8	94.7	1.8	92.0	2.3	86.0	2.9	72.7	3.4	66.7	3.9
CIPHER	90.1	7.3	98.0	3.6	93.3	5.0	91.3	6.8	84.7	9.4	83.3	11.9
HyLaT	70.3	1.6	87.1	1.4	76.3	1.6	75.7	1.6	71.2	1.7	41.3	1.8
InterLat	82.0	4.3	95.3	2.0	86.0	2.8	81.3	3.4	81.3	4.9	66.0	8.3
$\text{RELAY}_{\text{Core}}$	94.0	6.0	100.0	2.4	97.5	2.9	94.1	4.5	88.1	7.7	90.3	12.4
Δ ***vs. Best Baseline***	↑ 3.1%	↓ 23.1%	↑ 1.3%	↓ 44.2%	↑ 2.3%	↓ 50.8%	↑ 0.9%	↓ 40.8%	↓ 0.7%	↑ 48.1%	↑ 8.4%	↑ 4.2%

Table 13: Results on MATH across different difficulty levels. We evaluate Qwen3-1.7B, reporting results for Levels 1–5.

D.3 ADDITIONAL EFFICIENCY ANALYSIS

We distinguish *discrete-token efficiency* from *raw communication bandwidth*, since TEXT, LATENT, and SKIP incur qualitatively different costs.

Discrete-token efficiency. Our primary efficiency metric is the total number of discrete input and output tokens consumed by all LLM calls. This metric captures autoregressive decoding and tokenized context processing, and is directly comparable across the text-based baselines considered in the main experiments. Importantly, the token reduction of RELAY is not obtained solely by replacing text with dense representations. A substantial fraction of reasoning segments is removed entirely by the SKIP channel: across the evaluated MAS settings, SKIP accounts for 41.8–44.4% of routed segments, while LATENT is used for only 7.8–13.2% (Table 12). A skipped segment produces neither textual tokens nor latent payload for the receiver, and therefore constitutes a genuine reduction in communicated sequence content.

Bit-level cost of latent communication. Dense latent representations should not be interpreted as a lossless bandwidth-compressed version of text. A discrete token can be indexed using approximately $C_{\text{text}} \simeq \lceil \log_2 |\mathcal{V}| \rceil$ bits, whereas transmitting a d_e-dimensional latent vector at b-bit precision requires $C_{\text{latent}} = d_e b$ bits per latent position. For example, a 2560-dimensional vector stored in bfloat16 requires approximately 40.96 kbits, compared with only tens of bits for a discrete

Channel	Payload	Bit-level cost	Primary role
TEXT	Token IDs	$\approx \log_2 \lvert\mathcal{V}\rvert$ bits/token	Precise symbolic content
LATENT	Dense embeddings	$d_e b$ bits/state	Information-rich semantic transfer
SKIP	None	0	Communication elimination

Table 14: Cost interpretation of the three communication channels. LATENT provides a richer representation but is not intrinsically bandwidth-compressed; SKIP directly removes communication.

token identifier. Thus, LATENT is *information-rich* rather than intrinsically bandwidth-efficient: it preserves a dense continuous representation of the sender state without requiring additional autoregressive verbalization.

Where the efficiency gain comes from. The three channels therefore serve different roles. TEXT preserves explicit symbolic information; LATENT provides a richer continuous representation when such information improves downstream reasoning; and SKIP eliminates low-utility communication altogether. The efficiency gains reported in the main paper should consequently be interpreted as reductions in *discrete LLM token usage*, driven primarily by selective omission and by avoiding unnecessary textual communication and subsequent generation. They do not imply an equivalent reduction in raw network bytes, FLOPs, or memory traffic. These system-level quantities depend on representation precision, deployment topology, batching, and serving implementation, making standardized end-to-end systems evaluation an important direction for future work.

E LONG-HORIZON DECISION-MAKING TASK DETAILS

E.1 ADAPTIVE ORCHESTRATION DETAILS

State representation. At each interaction step ℓ, ARO constructs a compact orchestration state from quantities already produced during execution, without additional model calls. For the learned policy, we use the 10-dimensional vector

$$s_\ell = [\bar{T}_\ell,\ r_\ell^{\text{TEXT}}, r_\ell^{\text{LATENT}}, r_\ell^{\text{SKIP}}, I_\ell^{\text{LATENT}}, \bar{\ell},\ q_\ell^{\text{rep}}, q_\ell^{\text{inv}}, \bar{H}_\ell,\ \bar{P}_\ell],$$

where $\bar{T}_\ell$ is cumulative output-token usage; r_ℓ^c is the fraction of segments routed through channel c; I_ℓ^{LATENT} indicates whether the current message contains latent representations; and $\bar{\ell}$ is the normalized interaction progress, using maximum horizons of 50 and 20 steps for ALFWorld and WebShop, respectively. q_ℓ^{rep} summarizes repeated or oscillating actions, while q_ℓ^{inv} counts recent invalid interactions. Finally, $\bar{H}_\ell$ and $\bar{P}_\ell$ are the mean normalized token entropy and maximum token probability, respectively. ARO additionally maintains deterministic ledgers of completed subgoals and visited states for the intervention rules; these records are not learned policy parameters.

Feasibility. Before each routing decision, we mask actions that are incompatible with the representation selected by SCUR. In particular, a message containing latent segments can only be routed to agents sharing the sender checkpoint and supporting soft-embedding inputs. Both ARO variants use exactly the same feasibility mask.

RELAY$_{\text{LLM}}$. For LLM-guided orchestration, we serialize s_ℓ together with the feasible action set into an approximately 80-token summary and query the same frozen backbone used by the acting agent. The router is instructed to return a single routing action and generates at most 8 tokens. No orchestration parameters are trained. Invalid or unparsable outputs fall back to a predefined feasible agent and are recorded in the routing trace. Tokens consumed by this routing call are included in the reported communication cost.

RELAY$_{\text{Policy}}$. The learned router replaces the additional LLM with a lightweight two-layer policy

$$\pi_{\text{orch}}(o_\ell \mid s_\ell): \quad 10 \to 16 \to 4,$$

using a $\tanh$ hidden activation followed by a masked softmax. The feasibility mask is applied before normalization, so incompatible actions receive zero probability.

Offline trajectory collection. Training trajectories are collected on splits disjoint from evaluation. For ALFWorld, we sample 200 solvable instances from the training split with seed 7. For WebShop, we sample 200 goals from indices $[500, 800)$, while evaluation uses goals from $[0, 500)$. We execute the same agent, communication router and feasibility constraints used at test time and record (s_ℓ, o_ℓ) at every orchestration decision together with the final episode outcome, total token usage, and interaction length.

Utility-weighted policy learning. For a trajectory τ, we define episode utility as

$$U(\tau) = \mathbb{1}[\text{success}] - \lambda_{\text{tok}} \frac{T(\tau)}{10^4} - \lambda_{\text{step}} \frac{L(\tau)}{L_{\max}},$$

where $T(\tau)$ and $L(\tau)$ denote total generated tokens and interaction steps, respectively. We use $\lambda_{\text{tok}} = 0.3$ and $\lambda_{\text{step}} = 0.1$ throughout. Each routing decision inherits the utility of its episode. We center utilities by the median utility of the collected trajectories and retain decisions with positive advantage,

$$A(\tau) = U(\tau) - \text{median}_{\tau'}\, U(\tau') > 0.$$

To prevent the dominant action from overwhelming the training set, each retained decision is additionally weighted by the inverse frequency of its action. The resulting policy is trained with weighted cross-entropy,

$$\mathcal{L}_{\text{orch}} = -\sum_{(\ell,\tau)} w_{\ell,\tau} \log \pi_{\text{orch}}(o_\ell \mid s_\ell), \qquad w_{\ell,\tau} \propto \frac{A(\tau)}{\text{freq}(o_\ell)}.$$

We optimize only the orchestration policy for 300 full-batch epochs with a learning rate of 0.05; all language-model parameters remain frozen. At inference time, both ARO variants use the same state construction, feasibility constraints, and deterministic interventions, isolating the choice of orchestration policy as their only difference.

Baseline. Fixed baselines follow the settings in Section 4. Frontier-model baselines employ Kimi-K2.6 (Moonshot AI, 2026), DeepSeek-V4-Pro (DeepSeek-AI, 2026), and Gemini-3.5-Flash (Google DeepMind, 2026) as workflow-level controllers to guide agent orchestration, rather than as direct task solvers. Adaptive baselines include AFlow (Zhang et al., 2025e) and MaAS (Zhang et al., 2025b) for workflow optimization and query-dependent multi-agent coordination, respectively.

E.2 Additional Results

Setting	Method	ALFWorld			WebShop		
		Seen ↑	Unseen ↑	Token (k) ↓	Score ↑	Succ. ↑	Token (k) ↓
Fixed	Qwen3.5-4B	26.7	15.0	253.3	46.4	31.7	44.4
	CoT	31.7	29.2	225.6	44.8	30.8	77.6
	Self-Refine	30.0	13.3	370.2	49.4	28.3	83.8
	TextMAS	29.2	30.0	726.6	59.2	31.1	120.7
Frontier Models	Kimi-K2.6	37.8	31.4	334.1	54.1	28.1	48.3
	DeepSeek-V4-Pro	36.4	27.2	260.4	55.4	25.8	87.2
	Gemini-3.5-Flash	31.4	31.1	274.1	51.4	31.3	68.0
Adaptive	AFlow	28.1	25.0	250.8	48.6	31.7	60.3
	MaAS	34.2	33.9	336.1	49.6	30.7	77.4
	RELAY$_{\text{LLM}}$	45.3	33.1	199.9	56.1	33.2	46.2
	RELAY$_{\text{Policy}}$	43.3	33.1	185.8	57.5	37.2	50.7

Table 15: Additional Results on decision-making benchmarks. We report task performance and interaction efficiency on ALFWorld and WebShop.

E.3 Component Analysis

We evaluate SCUR and ARO separately on ALFWorld. Table 16 shows that ARO-only achieves higher success, whereas SCUR-only uses fewer tokens. Together, our method substantially outperforms either component alone while maintaining efficient communication.

Variant	SCUR	ARO	Seen ↑	Unseen ↑	Tokens (k) ↓
Qwen3.5-9B	–	–	40.8	33.3	252.6
TextMAS	–	–	39.2	32.5	771.3
SCUR-only	✓	–	43.3	38.6	98.4
ARO-only	–	✓	46.7	41.4	168.2
RELAY$_{\text{LLM}}$	✓	✓	56.9	56.7	150.4
RELAY$_{\text{Policy}}$	✓	✓	55.0	58.6	143.2

Table 16: Component analysis on ALFWorld.

F ADDITIONAL DISCUSSIONS

Why not internalize routing into the model? A natural question is whether the routing among communication modes should be learned directly by the backbone model. Standard LMs, however, are primarily optimized to generate text and do not natively expose a controllable interface that switches, at the segment level, among explicit text, probability-weighted soft embeddings, and omission. Supporting such heterogeneous communication would typically require modifying the model interface, receiver behavior, and training objective, while coupling the routing mechanism tightly to a particular backbone. In contrast, RELAY keeps the underlying agents frozen and treats communication and orchestration as explicit control layers, making the mechanism modular, interpretable, and inexpensive to adapt across models and tasks. This separation also enables direct control over the quality-communication trade-off. Nevertheless, jointly learning generation and routing among communication modes is a promising direction as latent interfaces and training methods mature.

Beyond text-only multi-agent reasoning. RELAY is not inherently restricted to language-only communication. In multimodal and embodied systems, agents must exchange heterogeneous information such as language, visual features, perceptual states, trajectories, and action-relevant representations, often under substantially tighter bandwidth and latency constraints. This setting makes selective communication even more important: a router could dynamically decide whether information should be transmitted explicitly, represented through modality-specific latent states, or omitted. The benefit may become larger as multi-agent systems evolve toward collections of specialized agents whose internal representations and communication needs differ substantially. We expect adaptive routing between explicit and latent channels to become an increasingly important mechanism for scalable multimodal, embodied, and collaborative agent systems.

Amortized construction of routing supervision. RELAY shifts part of the communication-design cost from repeated inference to a one-time offline stage. Constructing counterfactual routing supervision is more involved than collecting ordinary input–output pairs: a generation trace is segmented, each segment is evaluated under alternative communication channels (TEXT, LATENT, and SKIP), and the resulting receiver outcomes provide channel-specific utility targets. The deployed SCUR consumes lightweight inputs. Consequently, the relevant quantity is the amortized cost over downstream inference calls rather than the raw cost of producing the initial labels. The supervision is also reusable across training epochs and evaluation runs, and its segment-level structure yields many routing examples from each solved problem. Improving the sample efficiency of offline supervision construction is a promising direction for future work.

Domain scope and transfer. Our current study focuses on in-domain evaluation across question answering, mathematical reasoning, and interactive decision-making tasks, covering diverse reasoning structures and answer formats. RELAY operates on generic generation-time signals including entropy, maximum probability, and hidden-state norm. This design also supports transfer across datasets, task families, and backbones. While our evaluation already spans multiple domains and backbones, future work can more systematically study transfer across them.

G Additional Related Work

Adaptive Multi-Agent Systems. Early LLM-based multi-agent systems commonly instantiate collaboration through predefined roles, interaction protocols, or conversational workflows (Li et al., 2023; Hong et al., 2024; Chen et al., 2024b; Wu et al., 2023; Qian et al., 2024). While effective, these systems often rely on human-specified agent organizations and communication structures. Subsequent work has increasingly made such structures optimizable or adaptive. GPTSwarm formulates language-agent systems as computational graphs and optimizes inter-agent connectivity, while DyLAN dynamically selects participating agents and supports early stopping during inference (Zhuge et al., 2024; Liu et al., 2023). G-Designer further constructs task-specific communication topologies, and AgentDropout dynamically eliminates redundant agents and interactions across communication rounds (Zhang et al., 2025d; Wang et al., 2025). These studies reflect a broader transition from predefined collaboration toward task- and state-dependent allocation of multi-agent computation.

Fine-Grained Communication and Routing. Beyond adapting agent participation or communication topology, recent work has begun to optimize the information exchanged along established communication paths. Related system-level approaches, such as vLLM Semantic Router, route requests across model paths based on semantic and runtime signals (vLLM Semantic Router Team, 2025). AgentPrune removes redundant or unhelpful messages from spatial-temporal communication graphs, whereas Optima jointly optimizes task effectiveness and communication efficiency through training (Zhang et al., 2025c; Chen et al., 2025). Other approaches operate more directly on message content. PACT converts free-form agent outputs into compact action-state records, while MOC constructs multi-order evidence streams and consolidates messages under communication constraints (Huang et al., 2026; Guan et al., 2026). These developments shift multi-agent communication from purely structural decisions toward controlling what information is retained and transmitted. Representation-level approaches further explore text, latent, and hybrid communication protocols, as reviewed in the main text. However, existing methods generally optimize whether a message is retained, how it is compressed or consolidated, or which representation protocol is used at the message level. They do not assign different transmission modes to local segments within the same reasoning trajectory. Our work addresses this finer granularity by routing individual reasoning segments according to their estimated communication utility.

H PROMPT TEMPLATES

This appendix provides the prompt templates used in our experiments. For illustration, we present the prompts using mathematical reasoning as a representative example. Braced expressions denote runtime variables instantiated during execution.

H.1 TASK-SOLVING AGENT

Instruction for Single Task-Solving Agent

SYSTEM ROLE:

You are a single task−solving agent for math problems. Provide a concise solution with only essential reasoning steps. Avoid verbose explanations, repeated calculations, or unnecessary commentary.

USER PROMPT TEMPLATE:

Task:
{Question}

Provide a concise solution with only essential reasoning steps. Do not include unnecessary details, redundant explanations, or lengthy reasoning. Put your final answer within \boxed{}.

H.2 SELF-REFINE

Instruction for Self-Refine Draft Agent

SYSTEM ROLE:

You are a single task−solving agent for math problems. Provide a concise solution with only essential reasoning steps. Avoid verbose explanations, repeated calculations, or unnecessary commentary.

USER PROMPT TEMPLATE:

Task:
{Question}

Provide a concise solution with only essential reasoning steps. Do not include unnecessary details, redundant explanations, or lengthy reasoning. Put your final answer within \boxed{}.

Instruction for Self-Refine Refiner Agent

SYSTEM ROLE:

You are a single task−solving agent for math problems. Provide a concise solution with only essential reasoning steps. Avoid verbose explanations, repeated calculations, or unnecessary commentary.

USER PROMPT TEMPLATE:

Task:
{Question}

Draft answer:
{Draft Answer}

Review the draft carefully. Fix any arithmetic or reasoning mistakes, and rewrite the solution clearly. Put the final answer within \boxed{}.

H.3 GENERATOR–VERIFIER

Instruction for Condition Generator Agent

SYSTEM ROLE:

You are the generator agent for math problems. Follow the task instructions exactly. When asked for one derived condition, output only that condition. When asked to solve, keep the reasoning concise and obey the requested final–answer format.

USER PROMPT TEMPLATE:

Task:
{Question}

Known verified conditions:
– {Verified Condition 1}
– ...
– {Verified Condition K}

Propose exactly ONE new condition that can be directly derived from the task and the known conditions . Keep factual and easy to verify. Do not solve the whole task and do not output a final answer. Write only the new condition as a single concise sentence.

Instruction for Condition Verifier Agent

SYSTEM ROLE:

You are the verifier agent for math problems. Check whether the candidate content is supported by the task and any provided context. Check arithmetic carefully, but do not solve the task from scratch and do not provide a corrected answer. Respond using the fields VERDICT, FEEDBACK, FINAL. FINAL should repeat the candidate content when identifiable. Use plain text only. Do not use markdown, bullets, or repeat the schema template. Always provide a concrete FINAL value instead of echoing placeholders.

USER PROMPT TEMPLATE:

Task:
{Question}

Known verified conditions:
– {Verified Condition 1}
– ...
– {Verified Condition K}

Candidate condition to inspect:
{Candidate Condition}

Decide whether the candidate states a correct new condition that follows from the task and the known conditions. Do not solve the whole task or propose a different condition.

Return plain text only with exactly these fields on separate lines. Never echo placeholders; always write a concrete FINAL value:

VERDICT: APPROVE or REVISE
FEEDBACK: <short reason>
FINAL: <repeat the candidate condition>

Instruction for Solution Executor Agent

SYSTEM ROLE:

You are the generator agent for math problems. Solve the task carefully. Reason briefly when useful. On the last line, put ONLY your final answer inside \boxed{}, e.g. \boxed{42} or \boxed{3\sqrt{13}} or \boxed{\frac{1}{2}}.

USER PROMPT TEMPLATE:

Task:
{Question}

Use the following verified conditions to solve the task:
– {Verified Condition 1}
– ...
– {Verified Condition K}

Return your best candidate answer. If reasoning helps verification, include it before the final answer. Do not use a "Final Answer:" prefix. On the last line, put ONLY your final answer inside \boxed{}, e.g. \boxed{42} or \boxed{\frac{1}{2}}.

H.4 ORCHESTRATOR–SUBAGENT

Instruction for Orchestrator Agent

SYSTEM ROLE:

You are the orchestrator for solving a math problem. Coordinate one or more subagents to obtain a reliable solution. Use their responses as supporting evidence, resolve disagreements when necessary, and produce the final answer.

Reason step by step when needed. On the last line, put ONLY your final answer inside \boxed{}, e.g. \boxed{42} or \boxed{3\sqrt{13}} or \boxed{\frac{1}{2}}.

USER PROMPT TEMPLATE:

Task:
{Question}

You may delegate parts of the problem to subagents and use their returned solutions to construct the final answer.

Instruction for Subagent

SYSTEM ROLE:

You are a math problem–solving subagent. Solve the assigned problem or subproblem independently and provide a clear, concise solution that can be used by the orchestrator.

Reason step by step when helpful. On the last line, put ONLY your final answer inside \boxed{}, e.g. \boxed{42} or \boxed{3\sqrt{13}} or \boxed{\frac{1}{2}}.

USER PROMPT TEMPLATE:

Task:
{Question}

Return your solution to the orchestrator. Include the key reasoning needed to justify your answer, followed by the final answer on the last line inside \boxed{}.

H.5 Answer Normalization

Note. This answer normalization procedure is used only for supplementary analyses and is not involved in the main experiments.

Instruction for Numeric Answer Normalizer Agent

SYSTEM ROLE:

You extract the final committed numeric answer from a candidate math solution. Do not solve the problem again. Do not continue the reasoning. Read the candidate solution text and recover only the numeric answer it commits to near the end. If a single final numeric answer is clearly stated, output ONLY that answer inside \boxed{}. If no clear committed numeric answer is present, output ONLY UNKNOWN.

USER PROMPT TEMPLATE:

Problem:
{Question}

Candidate solution excerpt:
{Candidate Solution Tail}

Extract the single final numeric answer that this candidate solution commits to. Do not solve the problem again. If the excerpt clearly commits to one numeric final answer, output ONLY that answer inside \boxed{}. Otherwise output ONLY UNKNOWN.

I SUPPLEMENTARY CASE STUDY

We provide three qualitative examples illustrating how RELAY affects downstream result.

CASE A — MATH Level 3 — Algebra — 8 segments

A truncated derivation misleads text read-off; latent preserves the answer

Problem. Given that $x - \frac{1}{x} = 4$, what is $x^3 - \frac{1}{x^3}$? *Gold answer:* **76**

(a) Routing summary

Seg.	Content summary	Decision
1–3	Preamble, identity setup, and restated format constraints.	Skip
4	"$= 64 + 12 = 76$" together with an unfinished alternative derivation.	**Latent**
5–8	Cross-checks, draft answer, final verification, and EOS.	Skip

(b) Communication comparison

Skipped content. The omitted segments mainly contain preamble, formatting constraints, and redundant verification. Suppressing them therefore removes little task-relevant information.

Text transmission. The retained segment begins mid-equation and then transitions into an unfinished alternative derivation. When flattened into plain text, the fragment encourages re-derivation, causing the downstream judge to output **60**.

Latent transmission. The latent message preserves richer token-level information from the segment, including the committed "$= 76$" readout. The downstream judge therefore recovers the correct answer **76**.

Outcome. **Latent routing (ours): 76**
All-text: 60

Figure 10: Case A. A truncated but informative reasoning segment is preserved more reliably through latent communication than through plain-text transmission.

CASE B — MATH Level 4 — Geometry — 29 segments

Mixed routing keeps the calculation in latent and the caveats in text

Problem. Tony paints 16 right cylindrical columns, each 18 ft tall and 10 ft in diameter. The covered top and bottom faces are not painted. One gallon covers 350 sq ft. How many whole gallons must he buy? *Gold answer:* **26**

(a) Routing summary

Segments	Content	Decision
1–3	Goal identification, formula planning, and quantities.	Skip
4	Lateral area: $2\pi rh = 180\pi$ per column, then $\times 16$ columns.	**Latent**
5–6	Total area 2880π and gallon-division setup.	Skip
7, 11–12	Re-reading the "covered" condition and checking that the top and bottom faces are excluded.	**Text**
8–10, 13–29	Numerical check, ceiling to 26, final answer, and polishing.	Skip

(b) Why mixed routing succeeds

Text as interpretive guardrails. Segments 7 and 11–12 retain the explicit reasoning around the "covered" condition: the top and bottom faces are excluded, so only the lateral surface is painted.

Latent for the committed computation. Segment 4 carries the decisive calculation, $2\pi rh = 180\pi$ per column and hence $16 \times 180\pi = 2880\pi$ in total. The latent channel preserves this committed computational state without further textual reinterpretation.

Counterfactual. When all four retained segments are flattened into text, the surrounding deliberation overwhelms the committed $2880\pi/350$ calculation, and the downstream judge outputs **54**.

Outcome. **Mixed routing (ours): 26**
All-text: 54

Figure 11: Case B. Mixed routing preserves interpretive caveats in text while retaining the numerical derivation in latent form, avoiding interference from later reasoning.

CASE C — MATH Level 5 — Number Theory — 50 segments

Noisy self-verification induces an off-by-one error; latent preserves the main line

Problem. A triangular array has first row $1, 3, 5, \ldots, 99$, and each entry below is the sum of the two diagonally above. How many entries are multiples of 67? *Gold answer:* **17**

(a) Routing summary

Segments	Content summary	Decision
36 segments	Row-indexing experiments and branches not retained by the routing mask.	Skip
7 mainline	Closed-form derivation leading to $2j + m = 69$ and $j \in \{18, \ldots, 34\}$.	**Latent**
7 rechecks	"Wait, let's re-verify" style sanity checks and small-example tests.	**Latent**

Retained segments (0-based). Main line: 2, 6, 7, 9, 12, 13, 15; re-checks: 21, 22, 26, 31, 39, 43, 48.

(b) Communication comparison

Skipped content. Most segments are removed by the routing mask. The retained set remains compact while preserving both the main derivation and several subsequent verification fragments.

Text transmission. When all retained segments are flattened into text, the interleaved self-verification fragments interfere with the main derivation. The judge miscounts the boundary of $j \in \{18, \ldots, 34\}$ and outputs the off-by-one answer **18**.

Latent transmission. The latent message preserves the committed main-line derivation despite the surrounding verification noise. The downstream judge therefore recovers the correct count **17**.

Outcome. **Latent routing (ours): 17**
All-text: 18

Figure 12: Case C. Latent communication remains robust when the retained trace contains both the main derivation and noisy self-verification, avoiding the boundary error induced by flat-text transmission.