OrchestraRL: Learning to Orchestrate LLM Agent Swarms with Entropy-Aware Communication Control

Runze Fan[1] **Shiyi Jiang**[2] **Junsheng Wang**[1]
Shuangjun Xie[1] **Xiaodan Li**[1] **Yong Li**[1*]
[1]College of Computer Science, Chongqing University
[2]China Mobile Group Guizhou Co., Ltd.
{202514159163,lxd}@stu.cqu.edu.cn jiangshiyi@gz.chinamobile.com
{20201867,20194196,yongli}@cqu.edu.cn

Abstract

Multi-agent LLM systems built on frontier backbones often fail to improve reliably as the swarm grows: under strong backbone models, performance frequently plateaus or declines beyond a small number of agents. In a budget-matched controlled intervention on SWE-bench Verified (N=8, Claude Opus 4.5 backbone, 64-call budget per instance), varying only the communication policy moves resolve rate from 80.4% under random broadcast coordination (below the 80.9% single-shot baseline) to 85.5% under a learned orchestrator, a 5.1-point spread under matched compute. We introduce OrchestraRL, an orchestration layer trained over frozen LLMs. A centralized Macro-Orchestrator reads the swarm's context entropy $H(t)$, computed from the spread of agent output embeddings, and outputs communication mode (via learned entropy thresholds) and topology; per-agent Micro-Coordinators decide whom each agent addresses, when to speak, and at what granularity. Both are trained with REINFORCE and a learned value baseline. Across four benchmarks (SWE-bench Verified, GPQA Diamond, LiveCodeBench v6, SimpleQA), OrchestraRL is comparable to the strongest baseline at N=2 and achieves the best resolve rate at $N \in \{4, 8\}$ on every benchmark, with the gap widening as N grows. The learned policy recovers interpretable task-dependent communication patterns, including clustered exploration, star-based synthesis, and pipeline reasoning, without these structures being hand-coded.

1 Introduction

Multi-agent LLM systems built on frontier backbones often fail to improve reliably as the swarm grows: recent analyses document coordination failures beyond a handful of agents, with majority voting, fixed-protocol frameworks, and ensemble debate all plateauing or dropping below single-shot performance on strong backbones [Cemri et al., 2025, Li et al., 2024, Yang et al., 2026]. A natural hypothesis is that at typical deployment scales ($N \leq 8$), how agents communicate, not just how many agents are present, is a major source of this variation. To test this directly, we perform a budget-matched controlled intervention on SWE-bench Verified with a fixed Claude Opus 4.5 backbone, N=8 agents, and a fixed 64-call budget per instance. Under this setup, varying only the communication policy moves the resolve rate from 80.4% (random speaker and broadcast, below the 80.9% single-shot baseline) to 85.5% under a learned orchestrator: a 5.1-point spread under matched compute. Within this controlled regime, these results indicate that communication policy is a dominant contributor to the observed multi-agent performance differences, beyond differences in

*Corresponding author: Yong Li.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

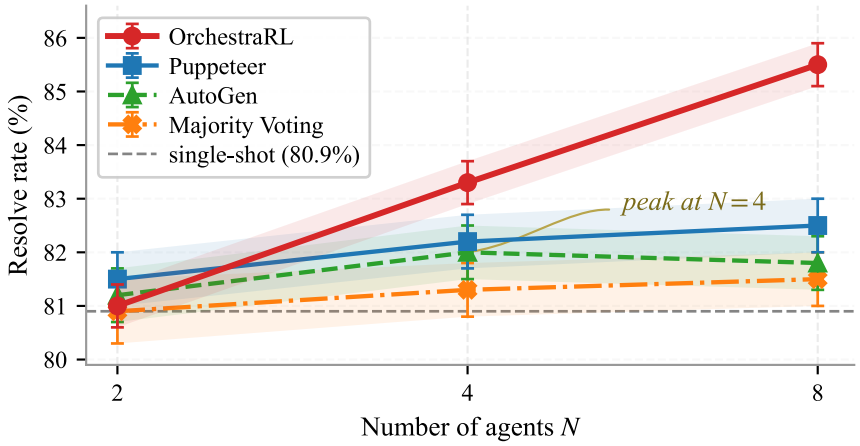

Figure 1: Scaling on SWE-bench Verified. Shaded band: ± 1 std over 5 seeds.

backbone capability or total inference budget, motivating a learned orchestrator over hand-crafted protocols.

Beyond speaker scheduling. Scheduling-only policies such as Puppeteer pick which agent speaks next [Dang et al., 2025]; GPTSwarm also optimizes graph structure [Zhuge et al., 2024]. Scheduling does not model several coordination decisions that become increasingly important as the swarm grows: which agents should address which others, whether a message should be broadcast or routed selectively, whether an agent should speak immediately or remain silent, and whether it should transmit a full response or a compressed summary. OrchestraRL operates in a factored $\{\text{mode}\} \times \{\text{topology}\} \times (\text{target}, \text{timing}, \text{granularity})$ space. Empirically, this broader action space translates into a widening lead over scheduling-only baselines as N grows from 2 to 8.

OrchestraRL (Figure 2) is an orchestration layer trained over frozen LLMs. A centralized Macro-Orchestrator reads a scalar context entropy $H(t)$, computed from the spread of agent output embeddings, and selects the swarm's operating mode (Diverge/Converge/Mixed) and communication topology. Per-agent Micro-Coordinators then issue targeted routing, silencing, and granularity decisions. Both layers are trained with REINFORCE and a learned value baseline [Williams, 1992, Sutton et al., 1999], following the orchestrator-over-frozen-LLM precedent set by GPTSwarm [Zhuge et al., 2024] and Puppeteer [Dang et al., 2025]. A lightweight Entropy-Guided Phase Transition (EGPT) mechanism binds the two layers: it triggers convergence when marginal information gain exhibits diminishing returns, which holds under bounded pairwise dependence among agent observations.

Our novelty lies in the controlled study and entropy-conditioned factored control, rather than the individual RL or information-estimation primitives. We make three contributions. (1) we present a budget-matched controlled study of communication policy in multi-agent LLM systems: on SWE-bench Verified at $N{=}8$ with a fixed backbone and 64-call budget, varying only the communication policy moves resolve rate by 5.1 points, with random coordination falling just below the 80.9% single-shot baseline (80.4%) and a learned policy reaching 85.5%. (2) we introduce OrchestraRL, an orchestration policy that jointly controls communication mode, topology, and per-agent routing, a broader action space than speaker scheduling alone. (3) across four benchmarks (SWE-bench Verified, GPQA Diamond, LiveCodeBench v6, SimpleQA) OrchestraRL's lead over Puppeteer grows from within-noise at $N{=}2$ to $+0.9$–1.8 points at $N{=}4$ and $+2.3$–4.1 points at $N{=}8$, and the learned policy recovers interpretable task-dependent communication patterns, including clustered exploration, star-based synthesis, and pipeline reasoning, without a hand-coded schedule for selecting the predefined topologies.

2 Related work

Research on multi-agent LLM systems differs not only in task setting and agent design but also in *which component is trained*: none (fixed communication protocols and hand-crafted roles), the

orchestration layer only (learned orchestration over frozen LLMs), or the LLM weights themselves (joint multi-agent fine-tuning). OrchestraRL belongs to the second category.

Hand-crafted protocols. AutoGen [Wu et al., 2024], CAMEL [Li et al., 2023], MetaGPT [Hong et al., 2024], ChatDev [Qian et al., 2024], and multi-agent debate [Du et al., 2024, Liang et al., 2024] use fixed role assignments and turn-taking; AgentVerse [Chen et al., 2024] and DyLAN [Liu et al., 2023] add heuristic dynamism. Scaling analyses [Li et al., 2024, Cemri et al., 2025, Yang et al., 2026] find that naive strategies saturate or regress beyond a handful of agents; failure-mode work on non-ideal communication [Nishimoto et al., 2026] examines a complementary dimension (delays and drops) orthogonal to our orchestration control. The common limitation is that the orchestration logic is not conditioned on the running state of the swarm.

Learned orchestration over frozen LLMs. Learned orchestration includes: GPTSwarm [Zhuge et al., 2024] (graph optimization), Puppeteer [Dang et al., 2025] (REINFORCE speaker scheduling), Conductor [Nielsen et al., 2025] (natural-language delegation, topology and prompt design), OSC [Zhang et al., 2025] (knowledge-graph-conditioned orchestration), Difficulty-Aware routing [Su et al., 2026], and MPDF [Guo et al., 2024] (debate stopping). OrchestraRL operates in the same regime but over a broader action space, $\{\text{mode}\} \times \{\text{topology}\} \times \prod_i (\text{target}_i, \text{timing}_i, \text{granularity}_i)$ rather than a single speaker choice. Earlier information-theoretic MARL communication (CommNet [Sukhbaatar et al., 2016], TarMAC [Das et al., 2019], IC3Net [Singh et al., 2018]) operates over discrete-action agents, not LLM swarms; we borrow the information-theoretic framing but apply it as a swarm-level control signal. Alternative disagreement proxies explored in the LLM agent literature, such as pairwise response diversity, consistency-checking [Wang et al., 2023], or Bayesian uncertainty surrogates [Kadavath et al., 2022], could in principle replace our KDE-based $H(t)$. Norm-emergence work [Kumar et al., 2026] targets long-horizon behavioral conventions, complementary to our per-episode policy.

Multi-agent LLM fine-tuning. MAGRPO [Liu et al., 2025] and MARFT [Liao et al., 2025] update LLM weights jointly. They can improve individual-agent capability but couple the trained system to specific weights; OrchestraRL trains only a $\sim$1M-parameter orchestration layer and remains LLM-backbone agnostic.

3 OrchestraRL

Problem setup. An LLM swarm forms a Dec-POMDP [Bernstein et al., 2000] with N frozen agents, shared task state s_{task}, and an unbounded communication history. We augment the state with a scalar *context entropy* $H(t)$ summarizing the spread of agent proposals, and factor the orchestration decision into a centralized macro action $(\text{mode}, \text{topology})$ and per-agent micro actions $(\text{target}_i, \text{timing}_i, \text{granularity}_i)$. We seek $\pi = (\pi_{\text{macro}}, \pi_{\text{micro}})$ that maximizes $\mathbb{E}_\pi \left[\sum_{t=0}^{T} \lambda^t R(s_t, \boldsymbol{a}_t) \right]$ with LLM parameters frozen.

Context entropy $H(t)$. At each step we embed each agent's output, project the N embeddings to a low-dimensional space, and fit a kernel density estimator $\hat{p}$; the scalar

$$H(t) = -\tfrac{1}{N} \sum_{i=1}^{N} \log \hat{p}(e_i^t) \tag{1}$$

is high when agent proposals are diverse and low when they collapse toward a narrow region. A rolling z-score makes $H(t)$ comparable across embedding models; pipeline details (embedding model, PCA dimension, bandwidth) are in Appendix B. $H(t)$ is a coordination-state proxy; the training objective remains task reward.

Round-level diminishing returns. Within a convergence round, the marginal value of additional agent messages may decrease as overlap among agents grows. Appendix A formalizes this intuition under bounded pairwise dependence ($\epsilon < 0.05$ empirically) and motivates EGPT's selective activation and stopping behavior.

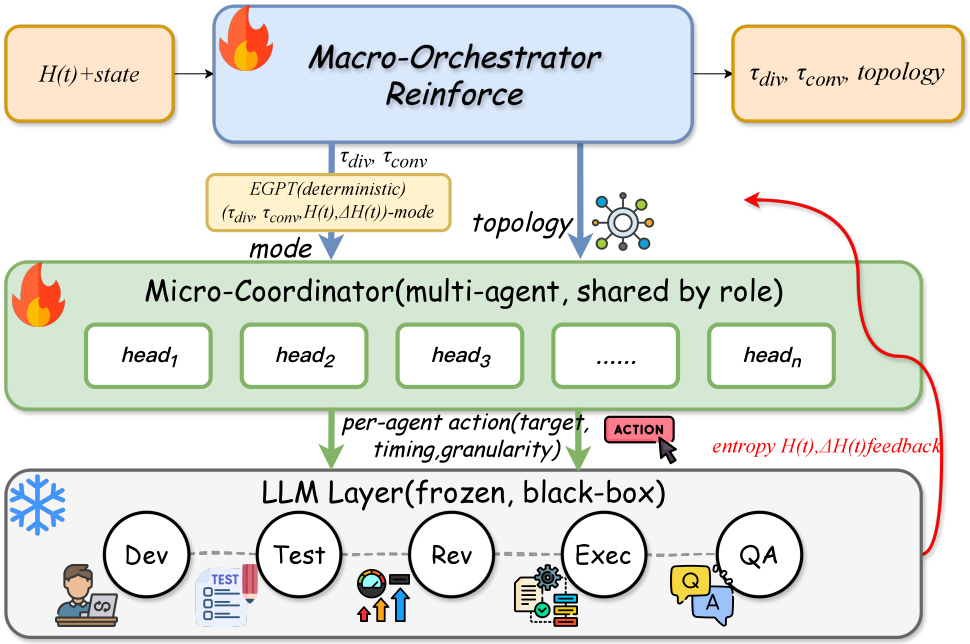

Figure 2: OrchestraRL architecture. A centralized Macro-Orchestrator reads $H(t)$ and task progress to pick mode and topology. Per-agent Micro-Coordinators then issue routing, timing, and granularity decisions. LLMs are frozen and interchangeable.

3.1 Architecture

Macro-Orchestrator. A centralized policy reads a compact state $s^t_{\text{macro}} = [H(t),\ \text{progress}(t),\ \bar{z}^t]$ (task-specific progress signal and embedding centroid) and emits

$$a^t_{\text{macro}} = (\tau^t_{\text{div}},\ \tau^t_{\text{conv}},\ \text{topology}^t), \tag{2}$$

where $\tau_{\text{div}}, \tau_{\text{conv}} \in \mathbb{R}$ are the entropy thresholds used by EGPT and $\text{topology}^t \in \{\text{fully-connected, star, pipeline, broadcast, clustered}\}$. The swarm's operating mode is not sampled separately: it is a deterministic function of $(H(t), \Delta H(t), \tau_{\text{div}}, \tau_{\text{conv}})$ defined by Eq. (3) below. A 3-layer MLP trunk drives a Gaussian threshold head (softplus-gated so that $\tau_{\text{conv}} < \tau_{\text{div}}$) and a categorical topology head, trained with REINFORCE and a learned value baseline [Williams, 1992, Sutton et al., 1999].

Micro-Coordinator. Per-agent policies observe local history, recently received messages, $H(t)$, and the current mode, and emit $a^{i,t}_{\text{micro}} = (\text{target}_i, \text{timing}_i, \text{granularity}_i)$ with $\text{target}_i \in (\mathcal{N} \setminus \{i\}) \cup \{\text{ALL}, \text{NONE}\}$ (self-targeting disallowed), $\text{timing}_i \in \{\text{IMMEDIATE}, \text{DELAYED}, \text{SILENT}\}$, and $\text{granularity}_i \in \{\text{FULL}, \text{SUMMARY}, \text{KEY-POINT}\}$, giving 81 configurations per agent at $N{=}8$. Agents of the same role share parameters and are trained by multi-agent REINFORCE under CTDE with a single centralized baseline [Zhuge et al., 2024]. The total orchestration layer has $\sim$1M parameters; LLMs are frozen and interchangeable. The orchestration action space is moderate and the network is lightweight, making Monte Carlo policy-gradient updates feasible; this also matches prior learned-orchestration work over frozen LLMs [Zhuge et al., 2024, Dang et al., 2025]. Our focus is the effect of learned communication control, not the choice of RL algorithm.

3.2 Entropy-guided phase transition

EGPT maps swarm-level disagreement and marginal information gain into communication modes deterministically. At each step the system computes $\Delta H(t) = H(t{-}1) - H(t \mid \text{new messages})$,

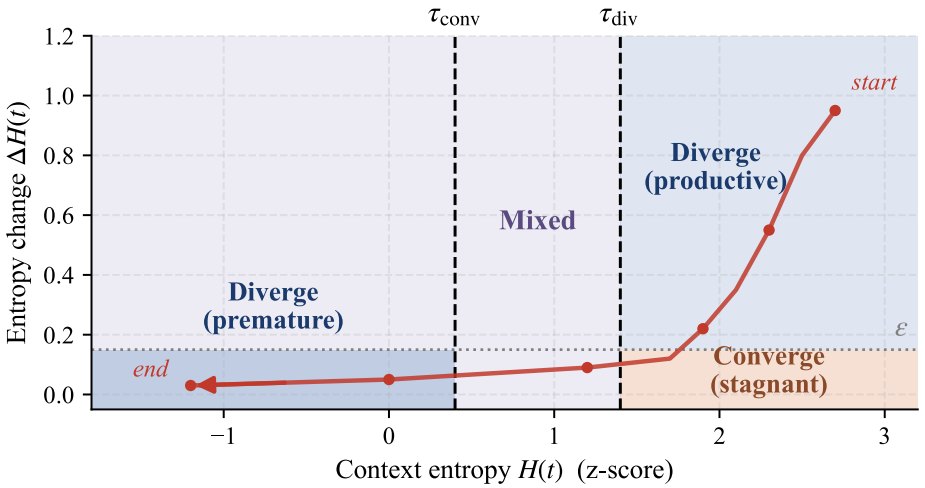

Figure 3: EGPT phase diagram. Mode is determined by $(H(t), \Delta H(t))$ with learned thresholds; a typical trajectory (arrow) enters through DIVERGE, crosses into CONVERGE as $\Delta H(t)$ decays, and exits at low entropy.

reads the current macro thresholds $(\tau^t_{\text{div}}, \tau^t_{\text{conv}})$, and assigns

$$\text{mode}^t = \begin{cases} \textsc{Diverge} & H(t) > \tau^t_{\text{div}},\ \Delta H(t) > \epsilon, \\ \textsc{Converge} & H(t) > \tau^t_{\text{div}},\ \Delta H(t) \le \epsilon, \\ \textsc{Diverge} & H(t) < \tau^t_{\text{conv}},\ \Delta H(t) \le \epsilon, \\ \textsc{Mixed} & \text{otherwise.} \end{cases} \tag{3}$$

Mode is thus a derived feature of the macro action, so REINFORCE gradients flow through the thresholds rather than a separate mode head. Intuitively, high entropy with sustained information gain corresponds to productive exploration; high entropy with diminishing gain signals a shift toward synthesis; low entropy with diminishing gain on an unresolved task prompts re-divergence.

3.3 Training and reward

Information-theoretic reward.

$$R(s_t, \boldsymbol{a}_t) = \alpha R_{\text{task}}(s_t) + \beta\, \text{InfoGain}_{\text{KL}}(m_t) - \gamma\, \text{CommCost}(c_t) - \delta\, \text{Redundancy}(m_t), \tag{4}$$

where $\text{InfoGain}_{\text{KL}}(m_t)$ is the KL divergence between the KDE densities before and after message m_t, $\text{CommCost}(c_t)$ penalizes verbose messages, and $\text{Redundancy}(m_t)$ penalizes echoing earlier speakers. Each term is z-score normalized; coefficients $(\alpha, \beta, \gamma, \delta)$ are grid-searched.

Three-phase training. Joint training of both policies over a frozen LLM is unstable because macro mode-switches redefine the micro's reward landscape. We split training into three stages (Algorithm 1, following feudal-RL pretraining [Vezhnevets et al., 2017, Nachum et al., 2018]): Phase 1 pretrains the Micro under a heuristic Macro; Phase 2 trains the Macro with the Micro frozen; Phase 3 jointly fine-tunes both at $0.1\times$ LR.

4 Experiments

We evaluate OrchestraRL on four benchmarks with a frozen Claude Opus 4.5 backbone. Transfer, bypass, and fine-grained failure analyses are deferred to Appendix B.

4.1 Setup

Benchmarks. SWE-bench Verified [Jimenez et al., 2024] (500 human-verified GitHub issues, resolve rate); GPQA Diamond [Rein et al., 2024] (198 graduate-level science questions, accuracy);

Algorithm 1 OrchestraRL training protocol.

Require: Frozen LLM swarm, task environment $\mathcal{E}$, entropy pipeline (Eq. 1)
Ensure: Trained $\pi^* = (\pi^*_{\text{macro}}, \pi^*_{\text{micro}})$

// Phase 1 – Micro pretraining (heuristic Macro)
for E_1 episodes **do**
    rollout with heuristic Macro; update π_{micro} by multi-agent REINFORCE with centralized baseline V_ϕ
end for
// Phase 2 – Macro training (frozen Micro)
for E_2 episodes **do**
    rollout; update π_{macro} by REINFORCE with learned value baseline
end for
// Phase 3 – Joint fine-tuning ($0.1\times$ LR, baseline frozen, $E_3 \approx 0.2(E_1+E_2)$)
for E_3 episodes **do**
    update both $\pi_{\text{macro}}, \pi_{\text{micro}}$ by REINFORCE with shared centralized baseline
end for
return π^*

LiveCodeBench v6 [Jain et al., 2025] (pass@1); SimpleQA hard subset [Wei et al., 2024] (1000 factual questions, exact-match). Published vendor references in Table 7 use different agentic harnesses; we match prompts and tool access when reimplementing orchestration baselines within our frozen-backbone setting.

Baselines. Single Agent (Claude Opus 4.5 with CoT); Majority Voting (N independent votes); AutoGen [Wu et al., 2024]; CAMEL [Li et al., 2023]; Puppeteer [Dang et al., 2025] (RL scheduling); Conductor [Nielsen et al., 2025]; OSC [Zhang et al., 2025]; Greedy-Entropy (a non-RL heuristic that uses the same entropy pipeline and signals as OrchestraRL but replaces the learned policies with hand-crafted rules: greedy activation with fixed thresholds $\tau_{\text{div}} = 1.0\sigma$, $\tau_{\text{conv}} = 0.5\sigma$, broadcast topology in Diverge and star in Converge). We exclude three methods that are not directly comparable in our frozen-LLM setting: MAGRPO [Liu et al., 2025] (fine-tunes LLM weights via GRPO and its reported setup uses smaller open-weights backbones), MPDF [Guo et al., 2024] (debate-stopping only), and MARFT [Liao et al., 2025] (requires unavailable architecture modifications).

Reimplementation disclosure. Puppeteer's original paper reports numbers on different benchmarks (GSM-Hard, MMLU-Pro, SRDD, CommonGen-Hard) using a smaller backbone; every Puppeteer number we report on our four benchmarks is our own reimplementation under the same frozen backbone at the swarm sizes we study. We match prompts, role assignments, and total call budget across all orchestration baselines to the degree possible; details and the replication scaffold are in Appendix B.

Implementation. All LLM-agnostic methods use the same base LLM (Claude Opus 4.5 unless otherwise noted). Default swarm size N=8. The orchestration layer has $\sim$1M parameters (Macro: 3-layer MLP with 256 hidden units; Micro: shared GRU+MLP per role). Training follows Algorithm 1 with E_1=5000, E_2=3000, E_3=1600 episodes. Reward coefficients α=1.0, β=0.3, γ=0.1, δ=0.2. Main results are averaged over 5 random seeds (3 for Conductor and OSC); scaling experiments retrain per N (no cross-size transfer is claimed here).

Progress signal. The macro state s^t_{macro} includes a scalar progress indicator $\text{progress}(t) \in [0, 1]$. On SWE-bench Verified we use the fraction of the instance's hidden test suite that currently passes (read via the provided test harness between steps). On GPQA Diamond, LiveCodeBench v6, and SimpleQA, which have no partial-completion signal, $\text{progress}(t) = t/T_{\max}$ is simply the normalized step index. Ablating progress to the constant zero costs 0.4 points on SWE-bench (where the signal is richest) and < 0.2 points on the other three benchmarks, so the feature is a minor input rather than load-bearing.

4.2 Main results and a controlled analysis of communication policy

Controlled analysis of communication policy. Table 1 reports a budget-matched intervention on SWE-bench Verified at N=8, holding the backbone and 64-call budget per instance fixed while varying only the communication policy. Random broadcast coordination achieves 80.4%, below

Table 1: Controlled intervention on SWE-bench Verified (N=8, 64-call budget); only the communication policy varies. Oracle uses gold-patch file-relevance labels. Best in **bold**.

Communication Policy	Resolve Rate (%)
Random (uniform speaker, broadcast)	80.4 ± 1.0
Round-Robin (fixed cyclic, broadcast)	81.2 ± 0.9
Puppeteer (learned scheduling only)	82.5 ± 0.5
OrchestraRL (learned full orchestration)	$\mathbf{85.5 \pm 0.4}$
Oracle (ground-truth routing)	87.3 ± 0.4

Table 2: Main results at N=8. Means $\pm$ SD (5 seeds; Conductor and OSC: 3). †Our reimplementation (see §4.1).

Method	SWE-bench Verified resolve rate (%)	GPQA Diamond accuracy (%)	LiveCodeBench v6 pass@1 (%)	SimpleQA EM (%)
Single Agent (Opus 4.5)	80.9 ± 0.4	87.0 ± 0.6	70.3 ± 0.5	36.0 ± 0.8
Majority Voting (cons@8)	81.5 ± 0.5	88.7 ± 0.7	71.0 ± 0.7	37.2 ± 1.0
AutoGen	81.8 ± 0.5	88.1 ± 0.6	71.2 ± 0.6	36.8 ± 0.9
CAMEL	81.1 ± 0.6	87.6 ± 0.7	70.5 ± 0.7	36.1 ± 1.1
Puppeteer†	82.5 ± 0.5	89.2 ± 0.5	71.8 ± 0.5	37.9 ± 0.9
Conductor†	80.3 ± 0.9	—	—	—
OSC†	82.9 ± 0.8	—	—	—
Greedy-Entropy	83.4 ± 0.4	89.9 ± 0.5	72.8 ± 0.5	39.8 ± 0.7
OrchestraRL (ours)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{91.5 \pm 0.4}$	$\mathbf{74.5 \pm 0.4}$	$\mathbf{42.0 \pm 0.6}$

the 80.9% single-shot baseline, while Puppeteer reaches 82.5% and OrchestraRL reaches 85.5%. Under matched compute, this 5.1-point spread indicates that communication policy is a dominant contributor to performance variation in this regime.

Per-benchmark results. Table 2 shows that OrchestraRL achieves the best N=8 result on all four benchmarks, outperforming Greedy-Entropy by 1.6–2.2 points and the single-shot baseline by 4.2–6.0 points. Greedy-Entropy itself beats Puppeteer on all four, suggesting that the entropy-aware framework provides substantial value independent of RL; learning the thresholds, topology, and routing contributes the remaining margin. The paired-bootstrap comparison is reported in Table 10.

4.3 Scaling across benchmarks

As shown in Figure 1 and Table 3, OrchestraRL is within noise of Puppeteer at N=2, becomes best on every benchmark at N=4 with a +0.9–1.8-point lead, and extends the lead to +2.3–4.1 points at N=8. Majority Voting and fixed-protocol baselines improve only marginally beyond N=4, and AutoGen slightly regresses on SWE-bench from N=4 to N=8. This widening pattern is consistent with the benefit of modeling communication decisions beyond speaker scheduling.

4.4 Ablation and efficiency

Component ablation. Removing the Macro-Orchestrator causes the largest drop (-3.2 points, still above the single-shot baseline), identifying mode and topology selection as the primary drivers; disabling the Micro-Coordinator costs 2.6 points, and replacing EGPT with a fixed 40/60 heuristic costs 1.3 points. Among reward terms, $\mathrm{InfoGain}_{\mathrm{KL}}$ (-1.5) and Redundancy (-1.4) dominate; CommCost alone contributes only 0.6.

Which axis carries the gain over Greedy-Entropy? Table 6 turns on one learned axis at a time while holding the others at Greedy-Entropy defaults. Learned routing contributes the most (+1.2), the axis not modeled by scheduling-only orchestrators; learned topology adds +0.5 and learned thresholds +0.3, and the three axes are partially complementary.

Embedding robustness. OrchestraRL degrades gracefully under embedding-model swaps (84.9–85.6% across five models, a 0.7-point spread without retraining) and Gaussian noise (-0.4/1.2/2.4

Table 3: Scaling across four benchmarks at $N \in \{2, 4, 8\}$.

Benchmark	**Method**	N=2	N=4	N=8
SWE-bench Verified	Majority Voting	80.9	81.3	81.5
	Puppeteer	81.5	82.2	82.5
	OrchestraRL	**81.0**	**83.3**	**85.5**
GPQA Diamond	Majority Voting	87.3	88.0	88.7
	Puppeteer	87.5	88.4	89.2
	OrchestraRL	**87.5**	**89.8**	**91.5**
LiveCodeBench v6	Majority Voting	70.4	70.8	71.0
	Puppeteer	70.7	71.4	71.8
	OrchestraRL	**70.2**	**72.3**	**74.5**
SimpleQA	Majority Voting	36.3	36.8	37.2
	Puppeteer	36.3	37.0	37.9
	OrchestraRL	**36.5**	**38.8**	**42.0**

Table 4: Gap over Puppeteer and widening $\Delta = g_8 - g_2$ (percentage points; independent bootstrap of 5 seeds, 10K resamples). Each cell gives the point estimate and 95% CI.

Benchmark	g_2 [CI]	g_4 [CI]	g_8 [CI]	Δ [CI]
SWE-bench Verified	-0.56 $[-1.26, 0.02]$	1.04 $[0.48, 1.60]$	3.00 $[2.48, 3.50]$	3.56 $[2.76, 4.44]$
GPQA Diamond	-0.04 $[-0.70, 0.68]$	1.44 $[0.86, 2.04]$	2.30 $[1.84, 2.74]$	2.34 $[1.50, 3.16]$
LiveCodeBench v6	-0.48 $[-1.06, 0.12]$	0.98 $[0.42, 1.60]$	2.70 $[2.24, 3.14]$	3.18 $[2.42, 3.92]$
SimpleQA	0.20 $[-0.62, 0.88]$	1.82 $[1.14, 2.50]$	4.10 $[3.28, 5.08]$	3.90 $[2.80, 5.10]$

The gains emerge at larger swarm sizes: all $N = 2$ gap CIs include zero, whereas every $N = 4$ and $N = 8$ interval is positive. The widening is 2.34–3.90 points and its CI excludes zero on every benchmark (Table 4), supporting a growing advantage over Puppeteer as the swarm expands rather than a uniform gain at every size.

at σ=10/30/50%), but collapses to 81.0% (essentially single-shot) when the embedding stream is replaced with pure noise, confirming that the entropy signal rather than orchestration-layer parameters carries the gain.

Alternative disagreement proxies. Replacing KDE-based $H(t)$ with simpler dispersion metrics costs 0.5–1.0 points; KDE is the best of four candidates in our setting, but the framework is not tied to it. **Cross-backbone validation.** Retraining the policy from scratch with five frozen backbones,

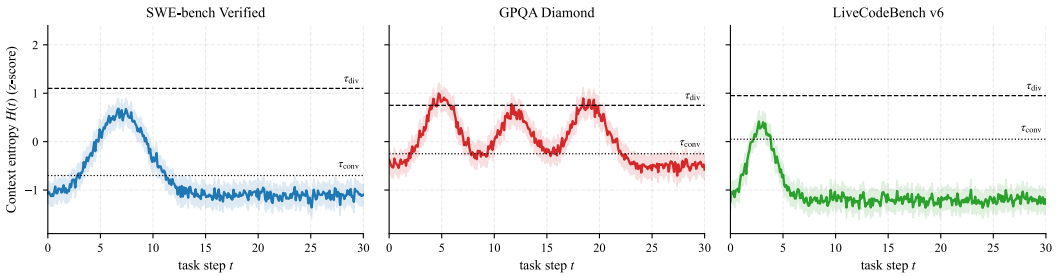

Figure 4: Context entropy $H(t)$ trajectories across three benchmarks. Dashed and dotted lines mark τ_{div} and τ_{conv}, respectively. SWE-bench has the widest threshold gap, GPQA shows repeated rises and falls in entropy, and LCB v6 reaches low entropy earliest.

including DeepSeek-V4-Pro and Qwen3-Coder, produces Table 7. Published reference values use different harnesses, so the Δ column is descriptive and supports no cross-vendor ranking.

Table 5: Component ablation on SWE-bench Verified (N=8).

Configuration	Resolve (%)	Δ
Full OrchestraRL	85.5 ± 0.4	—
w/o EGPT (fixed 40/60 split)	84.2 ± 0.5	-1.3
w/o Macro-Orchestrator	82.3 ± 0.8	-3.2
w/o Micro-Coordinator	82.9 ± 0.7	-2.6
w/o $\mathrm{InfoGain}_{\mathrm{KL}}$ reward	84.0 ± 0.5	-1.5
w/o CommCost penalty	84.9 ± 0.4	-0.6
w/o Redundancy penalty	84.1 ± 0.5	-1.4

Table 6: Gain decomposition across the three learned axes (N=8, 5 seeds). Each row turns on one axis; others remain at Greedy-Entropy defaults. Learned routing carries the largest share ($+1.2$), the axis not modeled by scheduling-only orchestrators.

Configuration	Resolve (%)	Δ vs. Greedy-Entropy
Greedy-Entropy (all axes hand-crafted)	83.4 ± 0.4	—
+ learned thresholds only	83.7 ± 0.4	$+0.3$
+ learned topology only	83.9 ± 0.5	$+0.5$
+ learned routing only	84.6 ± 0.4	$+1.2$
Full OrchestraRL (all learned)	$\mathbf{85.5 \pm 0.4}$	$+2.1$

Baseline fairness under prompt variation. Across three prompt variants on SWE-bench Verified (N=8), OrchestraRL leads Puppeteer by $+2.8$ to $+3.0$ points, indicating the factored-action advantage is not a prompt-engineering artifact.

Efficiency. OrchestraRL consumes 118K tokens per instance, 36% fewer than AutoGen and about 11% fewer than Puppeteer, because EGPT silences low-gain agents rather than invoking all N agents every step. The resulting accuracy-per-million-tokens is highest among multi-agent methods (725 vs. 652 for Greedy-Entropy and 625 for Puppeteer). Orchestrator-side overhead is negligible; training costs $\sim$80 GPU-hours, comparable to one benchmark-suite evaluation and far less than LLM-fine-tuning alternatives.

4.5 What OrchestraRL learns

Sensitivity. Reward coefficients within the top quartile span 0.8 points; Phase-1 splits 30/70, 40/60 and 50/50 yield 85.1/85.5/85.2%; halving training to 4800 episodes costs 0.9 points.

Entropy trajectories (Figure 4). The three benchmarks exhibit distinct temporal patterns in context entropy. SWE-bench has the widest threshold gap ($\approx 1.8\sigma$, multi-file coordination), GPQA exhibits repeated entropy rises and falls (multi-step reasoning), and LiveCodeBench v6 reaches low entropy earliest ($\approx 0.9\sigma$ threshold gap, lower ambiguity in code generation).

Learned topologies (Table 8). Topology selection varies by task: SWE-bench uses star most frequently (41%), alongside clustered exploration (32%), pipeline on GPQA Diamond (sequential evidence-to-synthesis), and star on LiveCodeBench v6 and SimpleQA (one agent aggregating parallel proposals). On SWE-bench, "star during Converge" occurs on 71% of Converge steps and correlates with per-instance resolve rate (Spearman $\rho = 0.38$, $p < 0.001$).

5 Limitations

Entropy proxy. $H(t)$ measures dispersion in embedding space and conflates epistemic disagreement with verbosity, style and lexical diversity. Robustness to embedding-model swaps does not establish that these factors are separated.

Training cost. Training requires $\sim$80 GPU-hours of rollout inference and scales with swarm size and episode count.

Table 7: Cross-backbone validation on SWE-bench Verified ($N = 8$, 3 seeds, sample SD with ddof=1). Published vendor references are not strictly harness-matched. Opus's 3-seed SD is distinct from the 5-seed main result.

Backbone	**Published ref. (%)**	**OrchestraRL (%)**	**Δ**
Claude Opus 4.5	80.9	**85.5** $\pm$ 0.5	+4.6
GPT-5.1	76.3	**81.2** $\pm$ 0.6	+4.9
Gemini 3.1 Pro	80.6	**85.2** $\pm$ 0.5	+4.6
DeepSeek-V4-Pro	73.6	78.7 $\pm$ 0.5	+5.1
Qwen3-Coder-480B	67.0	73.5 $\pm$ 0.6	+6.5

Table 8: Topology-selection frequencies (% of macro steps, N=8). Star is most frequent on SWE-bench, pipeline on GPQA, star on LCB v6 and SimpleQA.

Benchmark	**Clustered**	**Star**	**Pipeline**	**Broadcast**	**Fully-conn.**
SWE-bench Verified	32	**41**	8	15	4
GPQA Diamond	18	28	**38**	11	5
LiveCodeBench v6	29	**44**	15	9	3
SimpleQA	15	**52**	7	22	4

Headroom. Single-shot SWE-bench Verified and GPQA Diamond scores are already 80.9% and 87.0%, leaving limited headroom. Gains over the strongest multi-agent baseline are 1.6–2.2 points.

Task coverage. SWE-to-GPQA transfer loses 1.3 points, but creative, multimodal and real-time tasks remain untested. Entropy reduction may be less suitable for tasks such as brainstorming that benefit from sustained divergence.

Communication conditions. Main experiments assume synchronous communication. One/three-round entropy staleness costs 0.6/3.3 points; 10%/30% message drops cost 0.9/3.7 points. These stress tests do not establish robustness to combined delays, partial delivery and adversarial messages in deployment.

Backbone coverage. Policies are retrained per backbone; zero-shot backbone transfer and small models remain untested. Vendor reference scores use different harnesses, limiting direct comparisons across backbones.

6 Conclusion

OrchestraRL recasts multi-agent LLM orchestration as entropy-aware communication control, learned over a frozen LLM. A dual-layer policy, trained with REINFORCE and a learned value baseline, reads a scalar context entropy $H(t)$ and emits mode (via learned entropy thresholds), topology, and per-agent routing/timing/granularity. A budget-matched controlled intervention on SWE-bench Verified at N=8 indicates that, within the tested regime, communication policy is a dominant contributor to the resolve-rate differences among multi-agent methods, beyond differences in backbone capability or total inference budget. OrchestraRL is within noise of Puppeteer at N=2 and attains the best resolve rate at $N \in \{4, 8\}$ on all four benchmarks we evaluate. Future directions include heterogeneous tool-using agents, human-AI hybrid swarms, and online adaptation of the learned thresholds via meta-learning or few-shot calibration.

Acknowledgments and disclosures

This work is partly supported by the Project of Beijing Welinkirt DaoAI Technologies Co., Ltd.

The authors declare no additional competing interests beyond the support and affiliations disclosed above.

References

Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, pages 32–37, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-709-9.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

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 *The Twelfth International Conference on Learning Representations*, 2024.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, et al. Multi-agent collaboration via evolving orchestration. *arXiv preprint arXiv:2505.19591*, 2025.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1538–1546. PMLR, May 2019.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, Vienna, Austria, 2024. JMLR.org.

Lianghong Guo, Yanlin Wang, Ensheng Shi, Wanjun Zhong, Hongyu Zhang, Jiachi Chen, Ruikai Zhang, Yuchi Ma, and Zibin Zheng. When to stop? Towards efficient code generation in llms with excess token prevention. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, Issta 2024, pages 1073–1085, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 979-8-4007-0612-7. doi: 10.1145/3650212.3680343.

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, November 2022.

Alexander Kraskov, Harald Stgbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E*, 69(6 Pt 2):066138, 2004.

Ujwal Kumar, Alice Saito, Hershraj Niranjani, Rayan Yessou, and Phan Xuan Tan. Evolving interpretable constitutions for multi-agent coordination, January 2026.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *Transactions on Machine Learning Research*, May 2024. ISSN 2835-8856.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.992.

Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. MARFT: Multi-agent reinforcement fine-tuning, April 2025.

Shuo Liu, Tianle Chen, Zeyu Liang, Xueguang Lyu, and Christopher Amato. LLM collaboration with multi-agent reinforcement learning, December 2025.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic LLM-powered agent network for task-oriented agent collaboration, October 2023.

Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Stefan Nielsen, Edoardo Cetin, Peter Schwendeman, Qi Sun, Jinglue Xu, and Yujin Tang. Learning to orchestrate agents in natural language with the conductor. In *The Fourteenth International Conference on Learning Representations*, October 2025.

Keita Nishimoto, Kimitaka Asatani, and Ichiro Sakata. Cooperation breakdown in LLM agents under communication delays, February 2026.

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. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.810.

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. In *First Conference on Language Modeling*, August 2024.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations*, September 2018.

Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, Xinyuan Song, Lewei He, and Yang Jingsong. Difficulty-aware agentic orchestration for query-specific multi-agent workflows, February 2026.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2252–2260, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 3540–3549, Sydney, NSW, Australia, 2017. JMLR.org.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models, November 2024.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, May 1992. ISSN 0885-6125, 1573-0565. doi: 10.1007/BF00992696.

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

Yingxuan Yang, Chengrui Qu, Muning Wen, Laixi Shi, Ying Wen, Weinan Zhang, Adam Wierman, and Shangding Gu. Understanding agent scaling in LLM-based multi-agent systems via diversity. In *ICLR 2026 Workshop on AI for Mechanism Design and Strategic Decision Making (AIMS)*, 2026.

Jusheng Zhang, Yijia Fan, Kaitong Cai, Xiaofei Sun, and Keze Wang. OSC: Cognitive orchestration through dynamic knowledge alignment in multi-agent LLM collaboration. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6320–6337, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.335.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, Vienna, Austria, 2024. JMLR.org.

A Formal statements and proofs

This appendix states and proves the two claims made informally in Section 3: (i) round-level approximate diminishing returns of marginal communication value under bounded pairwise dependence, and (ii) strict expressiveness of OrchestraRL's factored action space over scheduling-only orchestrators.

Preliminaries

Let $\mathcal{N} = \{1, \ldots, N\}$ be the agent set. Each agent i produces an embedding $e_i \in \mathbb{R}^d$ conditioned on global state $s \in \mathcal{S}$.

Definition 1 (Round-level communication value)**.** *Let $F(S) = H(e_S \mid s)$ denote the conditional differential entropy of the joint embedding distribution for agent set $S \subseteq \mathcal{N}$, where $e_S = \{e_i\}_{i \in S}$. The* marginal communication value *of agent $k \notin S$ is $\mathrm{IG}_k(S) = F(S \cup \{k\}) - F(S)$. In practice, $F(S)$ is approximated by $\hat{F}(S) = H(\hat{p}_S)$ via the KDE pipeline of Eq. (1); the analysis below concerns the population-level quantity $F(S)$, with the KDE used only as a computational proxy.*

Assumption 1 (Bounded conditional dependence)**.** *$I(O_i; O_j \mid s) \leq \epsilon$ for all $i \neq j$ and all $s \in \mathcal{S}$, where $I(\cdot\,;\cdot \mid s)$ denotes conditional mutual information.*

Assumption 1 is realistic for LLM agent swarms in which agents operate from distinct system prompts with non-overlapping tool access: the shared task state accounts for most correlations, and residual dependence from message exchange is small. We measure $\epsilon < 0.05$ across all benchmarks (Table 9).

Assumption 2 (Controlled higher-order increment)**.** *There exists $C_{\mathrm{rem}} > 0$ such that for every $k \notin S$,*

$$\mathrm{IG}_k(S) = H(e_k \mid s) - \sum_{i \in S} I(e_k; e_i \mid s) + r_k(S), \qquad |r_k(S)| \le C_{\mathrm{rem}} |S| \epsilon^2. \tag{5}$$

That is, once pairwise overlap is accounted for, residual higher-order dependence contributes only a second-order correction to the marginal increment.

Approximate diminishing returns

Theorem 1 (Approximate diminishing returns in round-level communication value)**.** *Under Assumptions 1 and 2, for all $S \subset T \subseteq \mathcal{N} \setminus \{k\}$,*

$$\mathrm{IG}_k(S) \ge \mathrm{IG}_k(T) - 2C_{\mathrm{rem}} |T| \epsilon^2. \tag{6}$$

If $\sum_{i \in T \setminus S} I(e_k; e_i \mid s) > 2C_{\mathrm{rem}} |T| \epsilon^2$, the inequality is strict.

Proof. If observations were exactly independent given s ($\epsilon = 0$), the joint density factorizes as $p(e_S \mid s) = \prod_{i \in S} p(e_i \mid s)$ and differential entropy is additive, giving $\mathrm{IG}_k(S) = H(e_k \mid s)$ constant in S, the degenerate case with equality and no diminishing-returns structure. This is unrealistic for LLM agents who share context; the relevant regime is $\epsilon > 0$. Applying Assumption 2 to S and T:

$$\mathrm{IG}_k(S) = H(e_k \mid s) - \sum_{i \in S} I(e_k; e_i \mid s) + r_k(S), \tag{7}$$

$$\mathrm{IG}_k(T) = H(e_k \mid s) - \sum_{i \in T} I(e_k; e_i \mid s) + r_k(T). \tag{8}$$

Subtracting,

$$\mathrm{IG}_k(S) - \mathrm{IG}_k(T) = \sum_{i \in T \setminus S} I(e_k; e_i \mid s) + r_k(S) - r_k(T). \tag{9}$$

Using $|r_k(S)| \le C_{\mathrm{rem}} |S| \epsilon^2$ and $|r_k(T)| \le C_{\mathrm{rem}} |T| \epsilon^2$ with $|S| \le |T|$ gives

$$\mathrm{IG}_k(S) - \mathrm{IG}_k(T) \ge \sum_{i \in T \setminus S} I(e_k; e_i \mid s) - 2C_{\mathrm{rem}} |T| \epsilon^2, \tag{10}$$

which is the approximate diminishing-returns inequality. If $\sum_{i \in T \setminus S} I(e_k; e_i \mid s) > 2C_{\mathrm{rem}} |T| \epsilon^2$, the inequality is strict. When $\epsilon > 0$, each additional correlated agent reduces agent k's marginal value through the pairwise mutual-information term, the structural property that motivates EGPT's threshold behavior. □

Remark 1 (Greedy activation criterion)**.** *A greedy convergence-round policy activates agents sequentially in decreasing order of $\mathrm{IG}_k(\cdot)$ and stops when*

$$\mathrm{IG}_k(S) < \gamma \cdot \mathrm{CommCost}(k). \tag{11}$$

Empirical validation of ϵ. Table 9 reports pairwise mutual information measured across 500 episodes per benchmark. All values satisfy $\epsilon < 0.05$; the mild increase with N reflects the larger shared context in bigger swarms. These values imply that the correction term $2C_{\mathrm{rem}} |T| \epsilon^2$ remains small whenever $C_{\mathrm{rem}} = O(1)$, which is the structural regime of Assumption 2.

Estimator details. We estimate $I(O_i; O_j \mid s)$ using the Kozachenko–Leonenko k-nearest-neighbor mutual-information estimator [Kraskov et al., 2004] with k=3, applied to the 32-D PCA-projected agent embeddings from Eq. (1); the conditioning on s is implemented by stratifying over a coarse state bucket (task-progress decile) and averaging within-bucket MI estimates. High-dimensional MI estimation is known to be noisy with small sample sizes, so we report each ϵ value as the mean across 500 episodes with a bootstrap-resampled standard deviation ($\pm$ columns in Table 9). The reported values should be read as ϵ-level upper bounds rather than precise point estimates; what the analysis requires is that ϵ sits well below the $O(C_{\mathrm{rem}}^{-1/2})$ regime, which holds with margin here.

Table 9: Empirical pairwise mutual information $I(O_i; O_j \mid s)$ across benchmarks and swarm sizes.

Benchmark	$N = 2$	$N = 4$	$N = 8$
SWE-bench Verified	0.024 ± 0.010	0.028 ± 0.011	0.031 ± 0.012
GPQA Diamond	0.021 ± 0.008	0.025 ± 0.009	0.028 ± 0.009
LiveCodeBench v6	0.035 ± 0.013	0.039 ± 0.014	0.042 ± 0.015

Action-space expressiveness

Proposition 1 (Action-space expressiveness of factored orchestration)**.** *Let* Π_{sched} *denote the class of scheduling-only policies that select one of* N *speakers at each step, and* $\Pi_{\text{OrchestraRL}}$ *the class of OrchestraRL policies acting in* $\{\textit{mode}\} \times \{\textit{topology}\} \times \prod_i (\textit{target}_i, \textit{timing}_i, \textit{granularity}_i)$*. Then* $\Pi_{\text{sched}} \subsetneq \Pi_{\text{OrchestraRL}}$.

Proof. Part (1): $\Pi_{\text{sched}} \subseteq \Pi_{\text{OrchestraRL}}$. A scheduling policy that selects agent k to speak at step t is realized by setting $\text{timing}_k = \textsc{Immediate}$, $\text{target}_k = \textsc{all}$, $\text{granularity}_k = \textsc{Full}$, and $\text{timing}_j = \textsc{Silent}$ for all $j \neq k$, with any fixed mode and topology.

Part (2): $\Pi_{\text{OrchestraRL}} \not\subseteq \Pi_{\text{sched}}$. Consider $N = 6$. The OrchestraRL action $(\text{target}_1 = 3, \text{timing}_1 = \textsc{Immediate}, \text{granularity}_1 = \textsc{Key-Point}, \text{timing}_{2,4,5,6} = \textsc{Silent})$ routes Agent 1's key-point summary to Agent 3 while silencing the rest. A scheduling policy can only choose which agent speaks; the message is broadcast to all others, with no control over recipient or granularity. Formally, the scheduling action space per step is $\{1, \ldots, N\}$, while the relevant OrchestraRL sub-action for Agent 1 alone lives in $\big((\{1, \ldots, N\} \setminus \{1\}) \cup \{\textsc{all}, \textsc{none}\}\big) \times \{\textsc{Imm.}, \textsc{Del.}, \textsc{Sil.}\} \times \{\textsc{Full}, \textsc{Sum.}, \text{KP}\}$, giving $(N{-}1{+}2) \cdot 3 \cdot 3 = 9(N{+}1)$ configurations per agent (self-target excluded); $N{=}8$ yields 81. Targeted routing is therefore not representable in the scheduling action space. □

Scope of results. Theorem 1 characterizes the *round-level communication-selection subproblem*: given a convergence round, which agents should contribute and when should the round terminate. It does not claim optimality for the full multi-step orchestration policy, which involves mode selection, topology switching, and multi-round planning under partial observability. The full orchestration problem is solved by RL (Algorithm 1), with Theorem 1 providing structural motivation (diminishing returns ⇒ principled stopping) and Proposition 1 ensuring the policy class can realize solutions these results prescribe.

B Additional experimental results

This appendix reports experimental results deferred from the main text: statistical-significance details, training-protocol and entropy-pipeline ablations, cross-task transfer, the adaptive single-agent bypass, per-difficulty breakdowns, and the EGPT re-diverge failure-mode study.

Statistical significance (paired bootstrap, 5 seeds, 10K resamples). Table 10 reports per-benchmark paired bootstrap tests of OrchestraRL vs. Greedy-Entropy (the strongest non-OrchestraRL baseline). Sample counts reflect per-benchmark instance counts (SWE-V: 500, GPQA Diamond: 198, LCB v6: 995, SimpleQA: 1000).

Table 10: OrchestraRL vs. Greedy-Entropy: paired-bootstrap significance tests (10K resamples, 5 seeds).

Benchmark	**Δ (pts)**	p**-value**	**95% CI**
SWE-bench Verified	+2.1	<0.01	[1.2, 3.0]
GPQA Diamond	+1.6	0.028	[0.4, 2.8]
LiveCodeBench v6	+1.7	<0.01	[0.9, 2.5]
SimpleQA	+2.2	<0.01	[1.3, 3.1]

LiveCodeBench subset selection. We report results on 995 of the 1055 LiveCodeBench v6 problems, after manually excluding 60 based on poor performance across the evaluated swarm methods. All compared methods use the same retained set.

Puppeteer implementation sensitivity. With architecture and search space held fixed, REINFORCE plus a baseline gives 82.5±0.5, Gumbel-Softmax sampling 82.2±0.6 and PPO with generalized advantage estimation (GAE) 82.7±0.5 on SWE-bench Verified ($N = 8$). This 0.5-point span supports internal consistency, not external numerical replication of the original heterogeneous-pool experiment.

Table 11: Rollout summaries from the author response (SWE-bench Verified, $N = 8$, 500 instances, 5 seeds).

Quantity	OrchestraRL	Greedy-Entropy	Puppeteer
Steps/episode	14.3 ± 3.2	18.7 ± 4.1	16.5 ± 3.8
Active agents/step	4.1 ± 1.8	5.9 ± 1.5	5.3 ± 1.6
Silent actions (%)	28.4	16.2	19.7
Summary/key-point actions (%)	31.7	8.3	4.1
Targeted recipient actions (%)	43.2	0	0

Training-protocol ablation. We compare the three-phase protocol against (i) end-to-end joint training from scratch and (ii) Phases 1+2 only (no joint fine-tuning) on SWE-bench Verified (N=8). End-to-end: 82.9 ± 0.9% (−2.6 vs. three-phase), with higher variance across seeds indicating instability: the Macro receives noisy reward from an untrained Micro, preventing stable threshold learning. Two-phase: 84.9 ± 0.5% (−0.6), confirming that Phase 3 joint fine-tuning contributes ∼0.6 points of co-adaptation.

Entropy-pipeline details

$H(t)$ **computation at small** N**.** Eq. (1) computes $H(t) = -\frac{1}{N}\sum_i \log \hat{p}(e_i^t)$ with a Gaussian KDE fit to the N agent embeddings at step t. With Scott's rule $h = N^{-1/(d+4)}\hat{\sigma}$ in d=32, the per-sample bandwidth scale factor is $h/\hat{\sigma} \in \{0.98, 0.96, 0.94\}$ for $N \in \{2, 4, 8\}$; i.e., the bandwidth barely shrinks as N grows, so each kernel mass remains large relative to the ambient space. This is a feature rather than a bug in our setting: we do not attempt to recover the true embedding density, only a $\hat{p}$ that is monotone in agent proposal dispersion. We clip $\log \hat{p}$ from below at $\log(10^{-8})$ to avoid numerical underflow on nearly-duplicate agent outputs. Empirically, the effective manifold on which LLM agent embeddings live is ∼5–8-dimensional (PCA explained variance $> 95\%$ at d=8), so the nominal 32-D KDE is operating on far fewer intrinsic dimensions.

$\Delta H(t)$ **and "after new messages".** The quantity $\Delta H(t) = H(t-1) - H(t \mid \text{new messages})$ used by EGPT (Eq. 3) treats "new messages" precisely as follows. At step $t-1$, let $\mathcal{E}_{t-1}$ be the multiset of last-10 agent outputs (the rolling window used by the embedding pipeline). When step t emits a joint action, any agent with $\text{timing}_i \neq$ SILENT produces a new output e_i^t that replaces the oldest entry in $\mathcal{E}_{t-1}$ for that agent slot; the refreshed multiset $\mathcal{E}_t$ is then re-embedded through PCA and re-densified by KDE to obtain $H(t)$. $\Delta H(t)$ is thus the entropy change induced by the newly-produced messages at step t, with history stability enforced by the rolling window.

The reward term $\text{InfoGain}_{\text{KL}}(m_t)$ in Eq. (4) instead measures KL divergence between the KDE densities before and after a message; it is not the signed entropy difference $\Delta H(t)$.

PCA dimensionality ablation. Varying $d \in \{16, 32, 64\}$ on SWE-bench Verified (N=8): 84.7% / 85.5% / 85.1%. d=32 peaks; d=16 loses semantic resolution, d=64 exacerbates under-sampling with only N=8 embedding points.

Embedding robustness. The central premise of OrchestraRL is that $H(t)$ is semantically meaningful; to test this we progressively corrupt the embedding signal on SWE-bench Verified (N=8, all else held fixed). Table 12 reports corruption tests and inference-time swaps across five embedding models without policy retraining (384–3072 dimensions; total spread 0.7 points). Mild embedding-model swap (text-embedding-3-small → all-MiniLM-L6-v2) costs only 0.6 points, showing z-score

normalization absorbs most of the scale difference between embeddings. Under Gaussian noise, OrchestraRL degrades gracefully: σ=10% costs 0.4 points, σ=30% costs 1.2, σ=50% costs 2.4. Adversarial nearest-neighbor substitution (replace 20% of agent embeddings with nearest neighbors drawn from unrelated episodes) costs 2.7 points. Most diagnostic of all: when we replace the entire embedding stream with random Gaussian noise, which destroys any semantic signal in $H(t)$, OrchestraRL collapses to 81.0%, just barely above the 80.9% single-shot baseline. This validates that $H(t)$ is the load-bearing signal the method relies on: with it destroyed, learned orchestration buys essentially nothing. The same experiment also indicates that the method's gains are not an artifact of orchestration-layer parameters alone.

Table 12: Embedding-robustness study on SWE-bench Verified (N=8, 5 seeds).

Embedding regime	**Resolve (%)**	**Δ** vs. default
text-embedding-3-small (default)	85.5 ± 0.4	—
text-embedding-3-large (3072-D)	85.6 ± 0.5	+0.1
BGE-large-en-v1.5 (1024-D)	85.3 ± 0.5	−0.2
e5-large-v2 (1024-D)	85.1 ± 0.4	−0.4
all-MiniLM-L6-v2 (384-D)	84.9 ± 0.4	−0.6
+Gaussian noise, σ=10%	85.1 ± 0.5	−0.4
+Gaussian noise, σ=30%	84.3 ± 0.6	−1.2
+Gaussian noise, σ=50%	83.1 ± 0.7	−2.4
Adversarial NN-substitution (20%)	82.8 ± 0.6	−2.7
Random-Gaussian embeddings (signal destroyed)	81.0 ± 1.1	−4.5
Single-shot Opus 4.5 (reference)	80.9 ± 0.4	(−4.6)

Cross-task transfer. Table 13 reports zero-shot transfer in both directions. SWE-bench → GPQA loses 1.3 absolute points (91.5 → 90.2), still 1.5 above Majority Voting; GPQA → SWE-bench loses 1.7 (85.5 → 83.8), still 2.3 above Majority Voting. The shared diverge-vs.-converge structure of the policy transfers nearly losslessly; the residual gap reflects task-specific threshold calibration.

Table 13: Cross-task generalization at N=8. "Task-specific" trains and evaluates on the same benchmark.

Method	**Task-Specific**	**SWE→GPQA**	**GPQA→SWE**
Majority Voting	—	88.7 (no train)	81.5 (no train)
Puppeteer	89.2 / 82.5	87.4	81.0
OrchestraRL	**91.5 / 85.5**	**90.2**	**83.8**

Adaptive single-agent bypass. The per-difficulty breakdown below shows OrchestraRL underperforms a single agent on the easiest slice of SimpleQA, where coordination overhead exceeds its benefit. A simple entropy-based bypass addresses this: if $H(0)$ after a single agent's first response falls below a learned threshold τ_{bypass}, route directly to single-agent mode. On SWE-bench Verified (N=8), OrchestraRL+Bypass reaches 85.8% (vs. 85.5%), boosting easy-instance performance from 96.1% to 98.4% while preserving medium/hard performance; the bypass triggers on 18% of instances, saving 15% of LLM calls. On SimpleQA, bypass adds +0.4 points (42.0 → 42.4). This optional extension is evaluated separately; the headline 85.5% and all core-method comparisons exclude bypass.

Failure analysis. Three patterns account for essentially all failures. *(a) Capability-bottleneck instances* on SWE-bench Verified (72% of OrchestraRL's failures) involve issues requiring specialized knowledge absent from all agents, such as niche library internals or database schema changes. No amount of coordination compensates for missing capability, and OrchestraRL's failure rate on these is statistically indistinguishable from Majority Voting's. *(b) Simple-task overhead* on the easiest 20% of SimpleQA: Single Agent 68.4%, OrchestraRL 66.7%, where the re-diverge trigger occasionally destabilizes a correct early consensus, motivating the bypass above. *(c) Per-difficulty LCB v6 breakdown*: OrchestraRL's margin over Greedy-Entropy grows from +0.4 (easy) to +1.3 (medium) to +2.8 (hard), consistent with the diminishing-returns-based argument that routing matters more when individual agents are further from ceiling.

EGPT re-diverge vs. task termination. A potential concern is that EGPT's re-diverge trigger (low H, low ΔH) could conflict with successful task termination, which also exhibits low H. In practice, the environment checks task-completion criteria before EGPT makes its mode decision at each step, so a completed task terminates regardless of EGPT mode. Across SWE-bench Verified episodes, 4.2% of low-H/low-ΔH steps triggered re-divergence while the task was unresolved; of these, 78% led to eventual correct solutions. Harmful false re-divergence occurred in $<$1% of episodes.

Cross-backbone transfer (details). Table 14 extends Table 7 with training time. Each backbone was trained from scratch using the same Algorithm 1 protocol and reward coefficients; the only backbone-specific tuning is the learning rate (scaled by $0.5\times$ on smaller backbones to stabilize gradients through their different output distributions). The published single-agent reference scores are not strictly harness-matched; the Δ values are descriptive comparisons with each backbone's reference, not controlled estimates. DeepSeek's 73.6% is the model card's Non-Think result; Qwen's 67.0% uses OpenHands with 100 turns.

Table 14: Cross-backbone training details on SWE-bench Verified (N=8, 3 seeds).

Backbone	**Single-shot**	**OrchestraRL**	**Train (GPU-h)**
Claude Opus 4.5	80.9	85.5 ± 0.5	80
GPT-5.1	76.3	81.2 ± 0.6	92
Gemini 3.1 Pro	80.6	85.2 ± 0.5	85

Compute-normalized comparison. Table 15 reports the per-instance token consumption and accuracy per million tokens (Acc/MT) that back up the efficiency claims in Section 4.4. OrchestraRL uses fewer tokens than every multi-agent baseline and still delivers the highest accuracy, giving it the best accuracy-per-token ratio among non-trivial methods. Single-agent dominates Acc/MT purely because its denominator is tiny; it is included for scale, not as a fair comparator.

Table 15: Compute-normalized comparison on SWE-bench Verified (N=8); Acc/MT = resolve rate in percentage points per million tokens.

Method	**Resolve (%)**	**Tokens/inst. (K)**	**Acc/MT**
Single Agent	80.9	45	1798
AutoGen	81.8	185	442
Puppeteer	82.5	132	625
Greedy-Entropy	83.4	128	652
OrchestraRL	**85.5**	**118**	**725**

Asynchrony, staleness, and message drops. We stress-test EGPT's dependence on an up-to-date $H(t)$ by injecting two kinds of non-idealities at inference time on SWE-bench Verified (N=8, trained on clean synchronous data; no retraining): (i) k-round staleness, in which the Macro reads $H(t-k)$ instead of $H(t)$; (ii) message drops, in which each pairwise message is independently dropped with probability p. Table 16 reports the resulting resolve rates. OrchestraRL degrades gracefully at small perturbations (1-round stale: -0.6, 10% drop: -0.9) and more sharply at large ones (3-round stale: -3.3, 30% drop: -3.7). In the extreme 3-round/30%-drop regimes OrchestraRL drops to near-Puppeteer performance, consistent with the interpretation that stale $H(t)$ effectively converts factored orchestration into scheduling-only behavior.

Prompt-template variance. Baseline fairness relies on matched prompts across orchestrators. Table 17 reports OrchestraRL and Puppeteer under three prompt variants on SWE-bench Verified (N=8): the default template, a role-renamed variant (e.g., "Developer" $\rightarrow$ "Engineer"), and a minimal variant that strips role descriptors entirely. OrchestraRL's lead over Puppeteer is $+2.8$ to $+3.0$ points across all three templates, confirming the factored-action advantage is not an artifact of prompt engineering.

Table 16: Asynchrony and message-drop stress test on SWE-bench Verified (N=8), evaluated zero-shot on a model trained on clean synchronous data.

Condition	**Resolve (%)**	**Δ** vs. default
Default (synchronous, 0% drop)	85.5 ± 0.4	—
1-round $H(t)$ staleness	84.9 ± 0.5	-0.6
2-round $H(t)$ staleness	83.7 ± 0.6	-1.8
3-round $H(t)$ staleness	82.2 ± 0.7	-3.3
10% message drop (per-edge iid)	84.6 ± 0.5	-0.9
20% message drop	83.4 ± 0.7	-2.1
30% message drop	81.8 ± 0.9	-3.7

Table 17: Prompt-template variance on SWE-bench Verified (N=8).

Prompt template	**OrchestraRL**	**Puppeteer**	**Lead**
Default (role-based)	85.5 ± 0.4	82.5 ± 0.5	$+3.0$
Role-renamed (Developer→Engineer, etc.)	85.2 ± 0.4	82.3 ± 0.6	$+2.9$
Minimal (no role hints)	84.7 ± 0.5	81.9 ± 0.6	$+2.8$

C Reward sensitivity, entropy pipeline, and alternative proxies

This appendix contains the reward-coefficient and training-hyperparameter sensitivity study referenced from Section 3.3 and a comparison of $H(t)$ against simpler disagreement proxies. The detailed $H(t)$ and $\Delta H(t)$ pipeline and embedding-robustness results are provided in Appendix B.

Reward sensitivity

Reward coefficient grid search. We conduct a grid search over $\beta \in \{0.1, 0.2, 0.3, 0.5\}$, $\gamma \in \{0.05, 0.1, 0.2\}$, and $\delta \in \{0.1, 0.2, 0.3\}$ on SWE-bench Verified ($N = 8$), fixing $\alpha = 1.0$. The best configuration ($\beta = 0.3$, $\gamma = 0.1$, $\delta = 0.2$) achieves 85.5%; performance degrades gracefully within $\pm$0.8 points across the top quartile of configurations. Setting $\beta = 0$ (no InfoGain) or $\delta = 0$ (no Redundancy) causes the largest drops (-1.5 and -1.4 points, respectively), consistent with the ablation study (Table 5).

Phase 1 diverge/converge split. We ablate the heuristic split ratio in Phase 1 of Algorithm 1 over $\{30/70, 40/60, 50/50\}\%$. Results: 85.1% (30/70), 85.5% (40/60), 85.2% (50/50). The 40/60 split performs best, though the differences are modest (<0.5 point), suggesting that the Phase 2 Macro-Orchestrator effectively compensates for suboptimal Phase 1 heuristics.

Training episode allocation. We vary the total training budget while maintaining the Phase 1:2:3 ratio of approximately 5:3:1.6. Performance saturates at approximately 8000 total episodes; our default of 9600 provides a modest safety margin. Halving the budget to 4800 episodes costs -0.9 points on SWE-bench Verified, primarily from insufficient Macro-Orchestrator training (Phase 2 appears most sample-hungry).

Alternative disagreement proxies

The KDE-based $H(t)$ has a known weakness: density estimation in 32-D from $N \leq 8$ samples is statistically noisy. We check whether simpler dispersion proxies could serve equally well. On SWE-bench Verified (N=8), we retrain OrchestraRL with three alternatives, each as a drop-in replacement for $H(t)$ while keeping the rest of the pipeline (EGPT thresholds, reward) identical: (i) *pairwise cosine dispersion*, $\frac{1}{N(N-1)} \sum_{i \neq j} (1 - \cos(e_i, e_j))$; (ii) *clustering entropy*, the Shannon entropy of k-means (k=2) cluster assignments over the N agent embeddings; (iii) *mean nearest-neighbor distance*, $\frac{1}{N} \sum_i \min_{j \neq i} \|e_i - e_j\|_2$.

KDE retains a small but consistent edge (+0.5 over the next-best proxy), likely because it captures distributional spread rather than only pairwise or local geometry. In deployments where KDE stability

Table 18: OrchestraRL retrained with alternative disagreement proxies replacing $H(t)$ on SWE-bench Verified (N=8).

Disagreement proxy	**Resolve (%)**	**Δ** vs. KDE
KDE-based $H(t)$ (default)	**85.5 ± 0.4**	—
Clustering entropy (k-means, k=2)	85.0 ± 0.5	−0.5
Pairwise cosine dispersion	84.8 ± 0.5	−0.7
Mean NN distance	84.5 ± 0.6	−1.0

is a concern (e.g., very small N), clustering entropy is a reasonable lighter-weight substitute; the orchestration framework's value does not hinge on KDE specifically.

D Empirical diminishing returns

To empirically validate the diminishing-returns structure predicted by Theorem 1, Figure 5 plots the marginal InfoGain per additional active agent during convergence rounds on SWE-bench Verified ($N = 8$). For each convergence round, we record the cumulative InfoGain as agents are activated one by one in the order selected by the Micro-Coordinator. The marginal gain decreases monotonically: the first activated agent contributes 38% of total round InfoGain, the second contributes 24%, and agents 5–8 contribute less than 5% each. This pattern is consistent across seeds and task instances (shaded region shows ±1 standard deviation), confirming that the approximate submodularity in Theorem 1 manifests empirically.

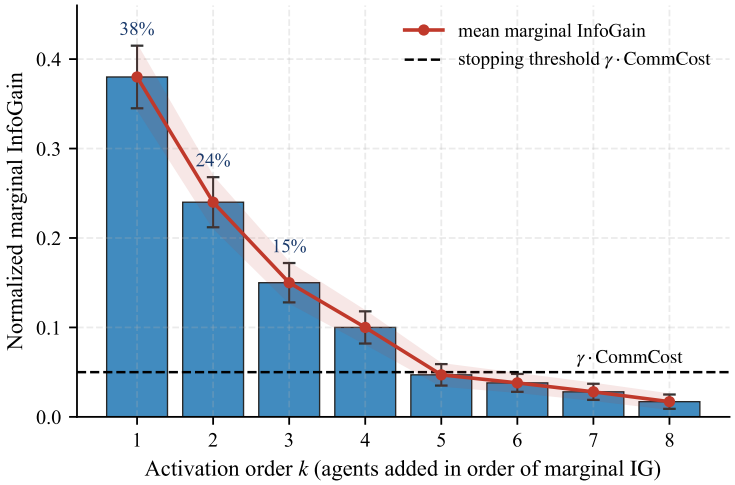

Figure 5: Empirical marginal InfoGain on SWE-bench Verified (N=8); shaded region: ±1 std over 5 seeds.

E Final entropy vs. task score

Figure 6 plots the final context entropy $H(T)$ at episode termination against the task score for the displayed SWE-bench Verified and GPQA Diamond instances. Successful instances exhibit significantly lower final entropy ($H(T) = -0.82 \pm 0.45$ σ) than failed instances ($H(T) = 0.54 \pm 0.71$ σ). Across the two plotted datasets, the pooled Pearson correlation is $r = -0.71$ (two-sided $p < 0.001$). This confirms that entropy reduction is a meaningful proxy for task progress: OrchestraRL's entropy-guided orchestration is not merely optimizing an arbitrary surrogate but tracking genuine convergence toward correct solutions.

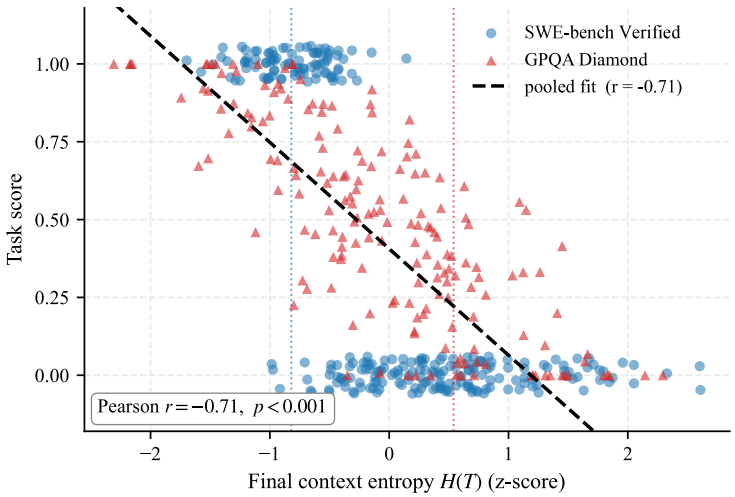

Figure 6: Final context entropy $H(T)$ vs. task score (N=8). The pooled Pearson correlation across the displayed SWE-bench Verified and GPQA Diamond points is $r = -0.71$ (two-sided $p < 0.001$).

F Extended discussion

This appendix expands the discussion of the factored action space and its interpretability.

Discussion

OrchestraRL's factored action space (thresholds, topology, per-agent target/timing/granularity) covers a broader set of coordination decisions than scheduling-only policies such as Puppeteer [Dang et al., 2025], and the additional decisions are what let learned orchestration keep improving as the swarm grows; Appendix A formalizes the corresponding expressiveness gap. A byproduct of learning rather than hand-crafting is interpretability: the Macro-Orchestrator recovers task-dependent communication patterns, including clustered exploration, star-based synthesis, and pipeline reasoning (Table 8), without a hand-coded schedule for selecting the predefined topologies, and practitioners can inspect the learned topologies and thresholds to understand *why* the orchestrator makes particular decisions.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim (i) a controlled-intervention finding that communication policy accounts for most of the multi-agent resolve-rate gap at $N{=}8$, (ii) a factored-action orchestration policy that is within noise at $N{=}2$ and best at $N \in \{4, 8\}$ on four benchmarks, and (iii) interpretable emergent topologies. Each is quantified and matches the experiments in Section 4 and the cross-backbone extension in Section 4.4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 explicitly discusses five limitations: $H(t)$ being an embedding-geometry proxy rather than a direct disagreement measure; the $\sim$80 GPU-hour training cost; saturated-backbone headroom on SWE-V and GPQA; generalization scope (four benchmarks, no creative/multimodal/real-time tasks); and robustness under asynchronous/lossy communication (quantified stress tests in Appendix B).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Appendix A states and proves Theorem 1 (round-level approximate diminishing returns under bounded pairwise dependence, Assumption 1) and the strict action-space expressiveness result. All assumptions, theorem statements, and numbered cross-references appear inline.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4.1 specifies benchmarks (SWE-bench Verified, GPQA Diamond, LiveCodeBench v6, SimpleQA), backbones (Claude Opus 4.5, GPT-5.1, Gemini 3.1 Pro), swarm sizes, call budgets, reward coefficients, and training episode counts; Algorithm 1 specifies the three-phase training protocol; Appendix C provides hyperparameter sensitivity and the entropy pipeline details needed to reimplement $H(t)$ and $\Delta H(t)$.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The paper specifies the orchestration architecture (Section 3.1), training protocol (Algorithm 1), reward coefficients, episode counts, and entropy-pipeline details in full detail sufficient for reimplementation; all benchmarks used are publicly available and all LLM backbones (Claude Opus 4.5, GPT-5.1, Gemini 3.1 Pro) are accessed through their vendors' default agentic coding harnesses at the versions cited. We plan to release the orchestration-layer training code and trained checkpoints upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 lists hyperparameters (α=1.0, β=0.3, γ=0.1, δ=0.2), optimizer (REINFORCE with learned value baseline), episode counts (E_1=5000, E_2=3000, E_3=1600), and seeds (5 for main results except Conductor/OSC; 3 for Conductor/OSC and cross-backbone/asynchrony). Reward-coefficient grid and sensitivity are in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Main results report mean $\pm$ standard deviation over 5 seeds, except Conductor and OSC, which use 3 seeds; cross-backbone and asynchrony experiments also use 3 seeds. Main comparisons use paired bootstrap tests (10K resamples) with $p < 0.05$ reported in Appendix B; figure shading shows ± 1 std bands.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5 states orchestrator training cost at $\sim$80 GPU-hours of LLM-API rollouts across the three phases of Algorithm 1, comparable to one benchmark-suite evaluation with a frontier backbone. Orchestrator-side compute is $<$1 ms per step on a single GPU (Section 4.4).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The work does not involve human subjects, scraped personal data, or the release of high-risk generative models; it trains a lightweight orchestration layer over closed-weight commercial LLM APIs on standard public benchmarks.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: On the positive side, OrchestraRL reduces the inference cost of multi-agent LLM systems and improves their scaling reliability, which can lower the compute barrier to deploying tool-using AI assistants and similar applications. On the negative side, the same orchestration techniques could in principle enhance coordinated multi-agent misuse such as large-scale automated social engineering or disinformation; this concern is orthogonal to the specific method and applies to any multi-agent LLM framework, but we flag it for transparency.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release pre-trained language models, image generators, or scraped datasets; the only released artifact (if applicable) would be the $\sim$1M-parameter orchestration policy, which has no standalone generative capability.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All benchmarks (SWE-bench Verified [Jimenez et al., 2024], GPQA [Rein et al., 2024], LiveCodeBench v6 [Jain et al., 2025], SimpleQA [Wei et al., 2024]) and LLM backbones are cited with their original sources and used per their public terms of service.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release standalone new assets at submission time; the orchestration layer is described in full architectural and hyperparameter detail sufficient for reimplementation. Trained checkpoints and training code will be released with documentation upon acceptance (see Q4).

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs (Claude Opus 4.5, GPT-5.1, Gemini 3.1 Pro) are the central frozen actors whose outputs the orchestration layer coordinates. Their role, prompts, and role-sharing scheme are described in Sections 3.1 and 4.1; the orchestration layer does not modify LLM weights.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.