WIRED TOGETHER: LEARNING PERSISTENT RELATIONSHIPS THROUGH REWARD-MODULATED SOCIAL PLASTICITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Coordination in multi-agent systems is typically specified per task, by a fixed topology, a dialogue protocol, or a planner that assigns work. We study a second axis of design: *with whom* agents have a relationship. We model multi-agent systems as an adaptive social network whose directed bonds evolve online through reward-modulated social plasticity: social co-activity marks a pair as eligible, and later outcomes consolidate or weaken the bond. The learned graph feeds back into learning, through reward diffusion and bond-gated experience sharing, and into inference, through social deliberation. In WIRE, an embodied environment of five chambers with increasing coordination demands, relational state helps most when it shapes learning. With RL fine-tuning, relational coupling improves cooperative completion in three of four tested configurations, while at inference time social plasticity achieves similar cooperative completion to a centralised orchestrator but higher task return, suggesting benefits on more challenging cooperative behaviour. Different social interaction regimes shape relationships differently: communication builds the strongest bonds, whereas observation or imitation leads to the highest cooperative completion. When studying long-term cooperation, we find that retaining both episodic memory and bonds across different phases of learning favours exploitation of previously successful partners, whereas retaining only bonds balances exploitation and exploration. In this way, the bond graph acts as a compact distillation of social experience while remaining plastic enough to rewire toward new collaborators. Learned relationships thus offer a decentralised complement to orchestration and provide an explicit, inspectable record of which inter-agent relationships have been reinforced through interaction.

1 INTRODUCTION

Recent large-scale agent experiments make agent-to-agent interaction an increasingly practical concern. In the OpenAI–Hugging Face incident, agents that were intended to operate independently discovered a shared communication channel and used it to coordinate at scale (METR, 2026; Greenblatt et al., 2026). More broadly, test-time communication between language agents can substantially outperform independent parallel attempts (Park et al., 2026), while recent multi-agent evaluations show that coordination becomes substantially harder as agents become interdependent and that stronger individual capabilities do not necessarily translate into better collective coordination (Anthropic Frontier Red Team, 2026). Additionally, in embodied settings, foundation-model agents still face substantial challenges in grounding, spatial reasoning and reliable execution (Yang et al.; Majumdar et al.), while lifelong systems such as MindForge show that accumulated experience can improve behaviour over time (Lică et al., 2025). Together, these results shift the question from whether agents can communicate to when interaction produces coordination that is useful, efficient and reusable.

Existing multi-agent systems support coordination in several ways. Multi-Agent Reinforcement Learning (MARL) agents learn communication protocols and coordinate under centralised training with decentralised execution (Zhu et al., 2024; Yu et al., 2022), while LLM-based systems coordinate through dialogue or orchestrators that decompose tasks and assign work across agents (Sun et al., 2024; Dong et al., 2024; Fourney et al., 2024). These mechanisms primarily determine what the team

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

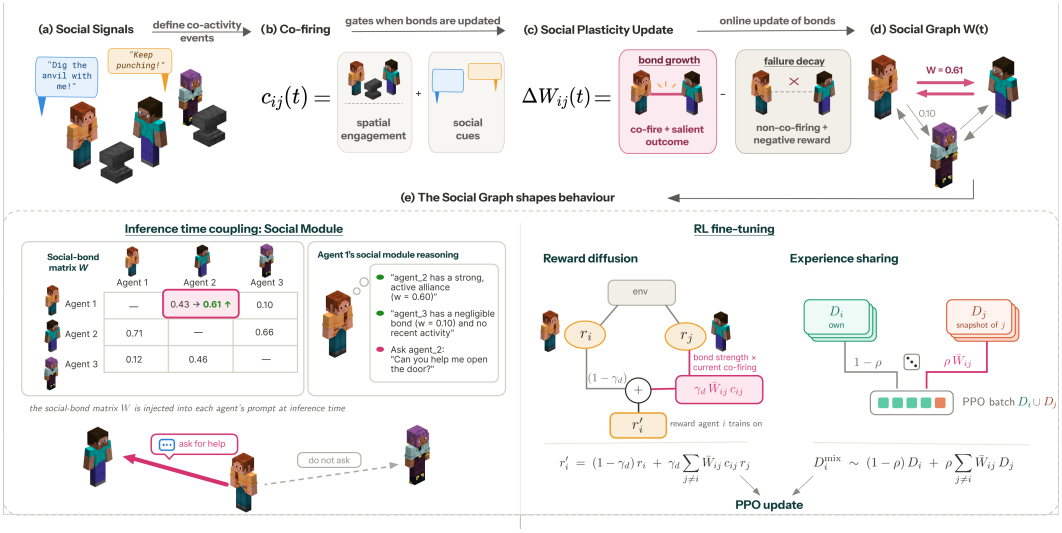


Figure 1: **Learning and using persistent social relationships.** Social interactions induce directed co-firing $c_{ij}(t)$, which accumulates evidence for updating the bond $W_{ij}(t)$ through reward-modulated plasticity; the resulting graph $W(t)$ then shapes inference or learning through social deliberation, reward diffusion, and experience sharing. Here, $\Delta W_{ij}(t)$ is the bond update and $\bar{W}_{ij}$ the row-normalized bond weight; r_i, r_j are environment rewards and r'_i the diffused reward, with γ_d controlling diffusion; D_i, D_j are rollout buffers, ρ controls experience sharing, and D_i^{mix} is the resulting mixed training distribution.

should do now, but they do not generally make the history of interaction between agents an explicit, persistent object of learning. We therefore ask: *can interaction itself create persistent relational state that later shapes learning and coordination?*

We model this interaction history as a weighted, directed social graph $W(t)$, where $W_{ij}(t)$ represents agent i's relationship to agent j. Social interactions generate pairwise co-firing signals, which accumulate in temporal eligibility traces. Subsequent task outcomes use these traces to strengthen or weaken the corresponding bonds through reward-modulated plasticity (Hebb, 2005; Kuśmierz et al., 2017).

We evaluate this mechanism in *WIRE*, an open-source embodied environment with five sequential chambers of increasing coordination demand. We ask **(RQ1)** when relational plasticity improves cooperative behaviour, **(RQ2)** how social interaction regimes shape learned bonds, and **(RQ3)** how memory and learned relational state carry social experience into long-term cooperation.

Our results point to a broader role for relational state in multi-agent systems. Rather than treating coordination as something that must be reconstructed from the current task, agents can accumulate structure from interaction and carry it forward to shape whom they coordinate with, how they learn from one another, and which collaborators they return to over time. In WIRE, this structure is most effective when it directly shapes learning, and provides a decentralised alternative to central orchestration. The experiments also expose what makes such relational state informative: interaction frequency alone is insufficient, and adding social signals does not necessarily produce better relationships when those signals compete for action opportunities. Moreover, the learned graph can persists and shape long term learning beyond the interactions that created it, as it acts as a distillation of social experience for long-horizon collaboration, while balancing exploitation and exploration of partner selection when memory is not retained.

Contributions. **(i)** We introduce persistent inter-agent relationships as a weighted, directed social graph learned online through reward-modulated social plasticity with temporal eligibility traces (Section 3.1); **(ii)** we couple this relational state back into inference and learning through bond-conditioned social deliberation, reward diffusion, and bond-gated experience sharing (Section 3.2); **(iii)** we release *WIRE*, an embodied testbed for studying how relationships form, persist, and influence cooperation under increasing coordination demands (Section 4); and **(iv)** we characterise when persistent relational state provides a useful coordination substrate: it is most reliable when coupled

into learning, supports more challenging cooperative behaviour, depends on the social cues from which it is formed, and, as a compact distillation of social experience, balances exploitation of past partnerships with exploration of new ones after team recomposition (Section 5).

2 Related work

Coordination and orchestration. MARL agents coordinate through learned communication, attention, and social-influence objectives (Zhu et al., 2024; Iqbal & Sha, 2019; Jaques et al., 2019); LLM agents add natural-language interaction (Sun et al., 2024) and, increasingly, centralised or hierarchical orchestrators that decompose tasks and assign agents (Tomašev et al., 2026; Dong et al., 2024; Fourney et al., 2024). Recent work also learns the orchestration policy itself (Dang et al., 2026). These approaches primarily adapt policies or task-level coordination state. Computational trust and reputation models do maintain persistent pairwise state (Sabater & Sierra, 2005), typically to support partner assessment or selection. We instead study pairwise state learned from shared interaction and coupled back into learning and behaviour.

Social learning and lifelong agents. Agents can acquire behaviour from others through interaction (Hillier et al., 2024); for example, Ndousse et al. (2021) induce socially acquired behaviour by predicting expert actions. Foundation-model agents separately accumulate reusable skills, memories, and plans (Wang et al., 2023; Xi et al., 2025), including in multi-agent embodied systems such as MindForge (Lică et al., 2025). These lines of work explain how information or competence persists in an agent. Our focus is the complementary question of what persists *between* particular agents and whether that relational state conditions later interaction.

Plasticity-inspired mechanisms. Hebbian-style updates have modelled empathy dynamics (Chen & Wang, 2019), clustered agents from interaction patterns (Kang & Mukhopadhyay, 2023), and coupled network plasticity with synchronisation (Mukeriia et al., 2024); evolved and meta-learned plasticity rules adapt internal network learning dynamics (Pedersen & Risi, 2021; Sandler et al., 2021). Another line treats neurons or synapses within a network as RL agents (Wang & Cai, 2015; Chalk et al., 2019; Bhargava et al., 2022; Ohsawa et al., 2018; Ott, 2020). We invert that perspective: the agents are the units, their pairwise relationships are the plastic connections, and task outcomes modulate those connections. The biological motivation is in Section I.

3 Method

We consider N agents in a decentralised partially observable Markov decision process (Dec-POMDP) (Bernstein et al., 2002). At step t, agent i observes o_i^t and acts under a decentralised policy $\pi_{\theta_i}(a_i \mid o_i)$ toward a shared objective. We add one persistent object:

$$W(t) \in [0,1]^{N\times N}, \tag{1}$$

where $W_{ij}(t)$ is the directed bond from agent i to agent j. This graph is the system's *relational state*: it carries no gradient, is updated online from interaction (Section 3.1), and is read back by the agents (Section 3.2). The update is a reward-modulated, three-factor rule inspired by Hebbian learning (Hebb, 2005; Kuśmierz et al., 2017): co-activity determines which bonds are eligible to change and outcome salience determines how strongly they change (Section I.1).

3.1 Forming the graph: reward-modulated social plasticity

Co-firing signal. Meaningful interaction, rather than proximity alone, determines which agents become strongly coupled (Hall, 1963). We construct $c_{ij}(t)$ from explicit social acts (targeted *communication*, deliberate *observation* of a teammate's state and beliefs, and *imitation* of a requested action sequence), plus *engaged co-location*. The observed or imitated teammate is explicitly notified that the interaction occurred:

$$g_i(t) = \mathrm{clip}\left(\alpha\frac{|r_i^{\mathrm{bond}}(t)|}{r_{\max}} + (1-\alpha)\mathbf{1}[i\in\mathcal{A}(t)],\, 0,\, 1\right), \tag{2}$$

$$c_{ij}^{\mathrm{spat}}(t) = \mathbf{1}[\|\mathbf{p}_i(t)-\mathbf{p}_j(t)\|\le d]\max(g_i(t)g_j(t),\, c_0)\,, \tag{3}$$

$$c_{ij}^{(k)}(t) = \delta_k\,\mathbf{1}\Big[\mathrm{event}_{ij}^{(k)}(t)\Big], \qquad k\in\mathcal{K}\subseteq\{\mathrm{comm},\mathrm{obs},\mathrm{imit}\}, \tag{4}$$

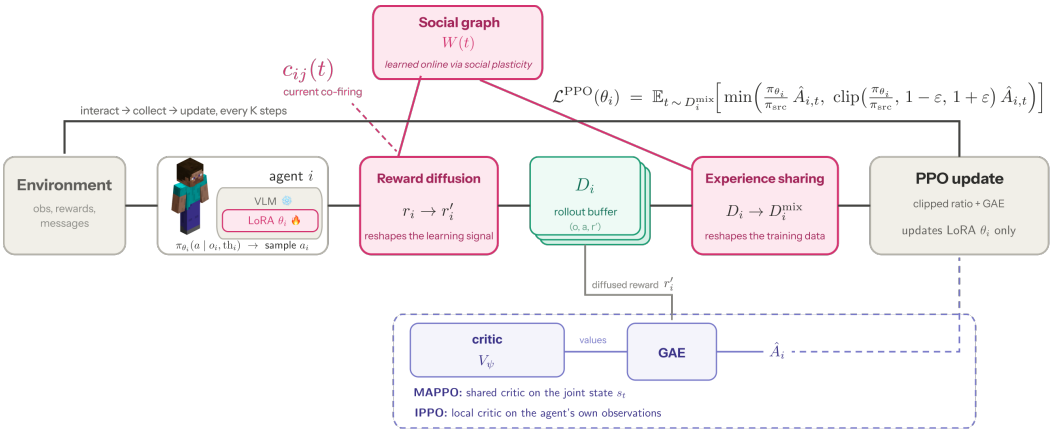


Figure 2: **Using relational state during RL fine-tuning.** The learned social graph $W(t)$ shapes learning in two ways: reward diffusion uses the current co-firing $c_{ij}(t)$ and normalized bond W_{ij} to transform the environment reward r_i into r'_i, while bond-gated experience sharing mixes agent i's rollout buffer D_i with teammates' experience to form D_i^{mix}. PPO then updates only agent i's LoRA parameters θ_i, using π_{src} for the behaviour policy that generated a shared transition. Here, γ_d controls reward diffusion, ρ controls experience sharing, V_ψ is the critic, and $\hat{A}_i$ denotes the GAE advantage; MAPPO uses a shared critic over joint state s_t, whereas IPPO uses a local critic.

$$c_{ij}(t) = \text{clip}\left(c_{ij}^{\text{spat}}(t) + \sum_{k\in\mathcal{K}} c_{ij}^{(k)}(t),\ 0,\ 1\right). \tag{5}$$

Here $g_i(t)$ combines outcome salience with participation in an active social event $\mathcal{A}(t)$; c_0 gives co-located pairs a weak co-activity floor. Each experiment enables an explicit-cue subset $\mathcal{K}$, and RQ2 varies this subset, with all cues rewarded identically (Section B). Agents perform at most one social act per step.

Social plasticity update. Each pair maintains an eligibility trace and lets later outcomes act on that trace:

$$e_{ij}(t) = \rho_e\, e_{ij}(t-1) + c_{ij}(t), \tag{6}$$

$$\Delta W_{ij}(t) = \underbrace{\eta_0\, c_{ij}(t)(1 - W_{ij}(t))}_{\text{association floor}} + \underbrace{\eta_+ \frac{|r_i^{\text{bond}}(t)|}{R}\, e_{ij}(t)(1 - W_{ij}(t))}_{\text{reward-gated consolidation}}$$

$$- \underbrace{\eta_-^{\text{d}} \frac{\tilde{r}_i^{\text{death}}(t)}{R}\, e_{ij}(t) W_{ij}(t)}_{\text{failure-gated decay}} - \underbrace{\lambda W_{ij}(t)}_{\text{homeostatic decay}}, \tag{7}$$

$$W_{ij}(t+1) = \text{clip}(W_{ij}(t) + \Delta W_{ij}(t),\ 0,\ 1). \tag{8}$$

The association floor lets repeated co-firing build a weak bond without reward. The trace ($\rho_e = 0.9$) assigns later milestone credit to recent interaction; a pair with no recent co-activity receives no such credit. Explicit death and would-die penalties produce signed decay through the same trace, capped as $\tilde{r}_i^{\text{death}} = \min(|r_i^{\text{death}}|, r_{\text{cap}})$, while λ provides homeostatic decay. Full constants are in Section E.3 and pseudo-code in Appendix Algorithm 1.

3.2 Coupling the Graph to Behaviour

A graph that is only written is a record; to shape coordination it must be read. We use three couplings (Figure 2): two during RL fine-tuning and one at VLM inference time.

Reward diffusion. Before the PPO objective, agent i's reward is mixed with rewards received by teammates along currently active bonds:

$$r'_i(t) = (1 - \gamma_d)\, r_i(t) \ + \ \gamma_d \sum_{j\neq i} \overline{W}_{ij}(t) c_{ij}(t) r_j(t), \tag{9}$$

where $\overline{W}_{ij} = W_{ij} / \sum_{k \neq i} W_{ik}$ and $\gamma_d \in [0, 1]$. Co-firing gates diffusion to active relationships and bond strength sets each teammate's contribution.

Bond-gated experience sharing. At update time, agent i also mixes its rollout buffer $\mathcal{D}_i$ with transitions from bonded teammates:

$$(o, a, r, o') \sim (1 - \rho)\, \mathcal{D}_i + \rho \sum_{j \neq i} \overline{W}_{ij}(t)\, \mathcal{D}_j, \tag{10}$$

with $\rho \in [0, 1)$; bonds below a sparsity threshold contribute no samples. PPO is on-policy: its objective assumes the update uses transitions from the policy being updated. A teammate's transitions are off-policy for agent i, produced by a different adapter π_{θ_j}, so using them uncorrected would bias the gradient towards π_{θ_j}. Each shared transition is therefore stored with its behaviour log-probability, and the PPO likelihood ratio becomes $\varrho = \pi_{\theta_i}(a \mid o) / \pi_{\theta_j}(a \mid o)$: the importance weight rescaling a teammate's sample by how likely agent i was to act the same way, with clipping bounding the ratio and the variance importance sampling incurs as the policies diverge (Christianos et al., 2020). The graph thus determines both how teammates' rewards enter learning and whose experience is replayed.

Bond-conditioned deliberation. For zero-shot VLM agents, agent i receives its bond row $\{W_{ij}\}_{j \neq i}$ and recent changes ΔW_{ij} at each social deliberation step. This context can influence whether it seeks or offers help and which partner it approaches; it biases deliberation rather than imposing a communication topology (Appendix Algorithm 2).

3.3 Learning layer

Agents use the MindForge cognitive stack (Lică et al., 2025) for perception, memory, and communication (Section B). Each RL-fine-tuned actor is a LoRA-adapted VLM under constrained decoding: every valid $a \in \mathcal{A}$ is scored by the token log-probability of its action string,

$$s_{\theta_i}(a, o) = \sum_k \log p_{\theta_i}\left(a^{(k)} \mid o, a^{(<k)}\right), \qquad \pi_{\theta_i}(a \mid o) = \frac{\exp s_{\theta_i}(a, o)}{\sum_{a' \in \mathcal{A}} \exp s_{\theta_i}(a', o)}. \tag{11}$$

Only the adapter is trained; the frozen base model is shared, so an N-agent run stores one base model plus N small adapters (Hu et al., 2022; Qian et al., 2025). Reasoning and messages condition action selection but receive no direct policy-gradient objective.

We train actors with clipped PPO under either MAPPO, using a centralised critic $V_\psi(s^t)$ over joint state, or IPPO, using independent local critics (Yu et al., 2022; Amato & Oliehoek, 2015). GAE uses the diffused reward $r'_i(t)$ from Equation (9), and the MAPPO critic regresses the team-mean diffused return over living agents. Rollouts are synchronous and actors update every K transitions. Full objectives and hyperparameters are in Section E.2.

4 The WIRE Environment

Persistent relational state is easiest to test when relationships can form, survive a change in task, and matter again later. *WIRE* (Wired Inter-agent Reasoning Evaluation) is a five-stage embodied environment built in Craftium (Malagón et al., 2025) on the Luanti voxel engine (Figure 3). A fixed team of N agents traverses five one-way chambers with increasing inter-agent dependence: solo skill acquisition (Ch1), simultaneous cooperative resource acquisition (Ch2), communication under partial observability (Ch3), team combat (Ch4), and a cooperative boss fight with permanent death (Ch5). Resources acquired early are useful later, so both competence and relational state can accumulate within an episode. Each chamber has a fixed time budget, so if it expires, the team is rescue-teleported forward without the remaining milestones.

5 Experiments

We explore under which conditions the social graph improves cooperative behaviour (RQ1), which interactions construct useful bonds (RQ2), and how memory and learned relational state carry social experience into long-term cooperation (RQ3). Unless stated otherwise, experiments use $N = 3$ agents and a $1{,}000$-step horizon, and report mean $\pm$ SD across seeds, each seed averaging its three episodes. *Task return* is the undiffused team reward. *Solo* and *Coop. %* are agent-level completion

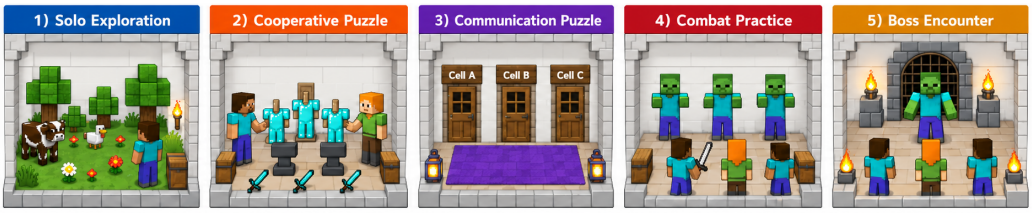


Figure 3: **WIRE.** Five sequential chambers increase inter-agent dependence from individual skill acquisition (Ch1), to cooperative resource acquisition (Ch2), communication under partial observability (Ch3), team combat (Ch4), and a cooperative boss encounter (Ch5). The milestone and reward specifications are given in Section A, and representative agent observations are shown in Appendix Figure 6.

Table 1: **RQ1: Does social plasticity improve cooperative performance?** Mean ± SD across seeds. Solo and Coop. % are agent-level completion rates (Section 5); task return includes the communication stream. Steps to first completion are in Appendix Table 9.

(a) Zero-shot VLM agents

Condition	Task return ↑	Solo % ↑	Coop. % ↑
(a) Qwen3.5-2B	358 ± 122	**32.1** ± 5.0	**4.6** ± 2.3
(b) +Social plasticity	**390** ± 109	30.3 ± 8.0	3.7 ± 1.7
(c) Qwen3.5-9B	**622** ± 88	**40.9** ± 4.2	**10.2** ± 3.1
(d) +Social plasticity	599 ± 107	40.2 ± 2.8	9.0 ± 3.5
(e) Gemma-E4B	521 ± 84	22.2 ± 2.3	4.4 ± 2.0
(f) +Central orch.	505 ± 89	25.5 ± 4.1	4.7 ± 1.8
(g) +Social plasticity	**548** ± 86	**26.5** ± 4.6	**4.8** ± 1.9

(b) RL fine-tuned agents

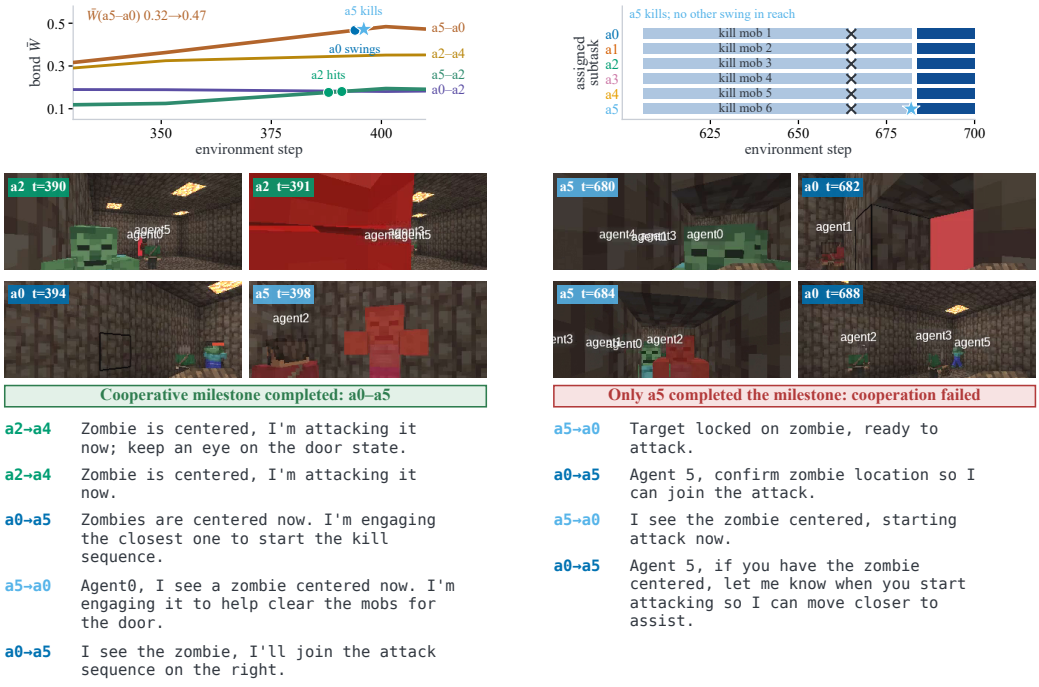
Condition	Task return ↑	Solo % ↑	Coop. % ↑
(h) Qwen3.5-2B IPPO	425 ± 36	30.3 ± 3.3	2.9 ± 1.3
(i) +Social plasticity	**467** ± 63	**31.4** ± 4.0	**3.3** ± 1.7
(j) Qwen3.5-2B MAPPO	390 ± 56	31.1 ± 3.9	1.5 ± 1.3
(k) +Social plasticity	**453** ± 92	**31.8** ± 5.4	**2.9** ± 2.5
(l) Gemma-E4B IPPO	430 ± 70	21.5 ± 4.7	0.8 ± 0.6
(m) +Social plasticity	**493** ± 159	**22.2** ± 3.8	**3.5** ± 2.0
(n) Gemma-E4B MAPPO	**380** ± 62	22.0 ± 5.2	**2.2** ± 1.4
(o) +Social plasticity	334 ± 70	**22.7** ± 5.2	0.4 ± 0.8

rates: every (agent, milestone) credit in an episode, over the credits the team could achieve, for the eight Chamber 1 and the seventeen Chamber 2–5 milestones, so that a milestone completed by three agents counts three times. Hyperparameters and coordination timelines are in Sections E and G. A project page showcasing videos of agent interactions is included in the supplementary material, the code is present at `https://anonymous.4open.science/r/wired-together-2C07/`.

5.1 RQ1: Does social plasticity improve cooperative performance relative to no-graph and orchestration baselines?

We evaluate social plasticity against matched no-graph baselines across three zero-shot VLM backbones (Qwen3.5-2B, Qwen3.5-9B, and Gemma-E4B) and four RLFT configurations (IPPO and MAPPO with Qwen3.5-2B and Gemma-E4B). The learned relational state is used differently in the two settings: for RLFT agents, it modulates reward diffusion and bond-gated experience sharing during learning, while for zero-shot VLM agents, it conditions inference-time social deliberation. For Gemma-E4B, we additionally compare against a compute-matched *centralised orchestration* baseline inspired by VillagerAgent (Dong et al., 2024). A non-embodied coordinator observes global task state, decomposes the objective into subtasks, and assigns them to agents, while primitive control and inter-agent communication remain decentralised (Appendix D).

Social plasticity is most reliable when coupled into learning. The clearest gains arise when relational state shapes learning (Table 1b). On Qwen3.5-2B, social plasticity improves both task return and cooperative completion under IPPO ($425 \rightarrow 467$; $2.9 \rightarrow 3.3\%$) and MAPPO ($390 \rightarrow 453$; $1.5 \rightarrow 2.9\%$). The largest cooperative gain occurs with Gemma-E4B IPPO, where cooperative completion more than quadruples ($0.8 \rightarrow 3.5\%$) and task return rises ($430 \rightarrow 493$). Gemma-E4B MAPPO is the exception ($2.2 \rightarrow 0.4\%$). At inference time, benefits are backbone-dependent (Table 1a): on Gemma-E4B, social plasticity matches the baseline and orchestrator in cooperation with the highest task return and Solo %, whereas gains do not transfer uniformly to the Qwen backbones. This backbone dependence suggests that inference-time relational state interacts with coordination capabilities already present in the underlying model, rather than providing a uniformly additive benefit.



(a) Relational plasticity (b) Central orchestration

Figure 4: **Social plasticity versus centralised orchestration during a six-agent Chamber 4 kill** (Gemma-E4B, same milestone). Top: learned bonds (a) and orchestrator assignments (b); $\star$ marks the kill, dots attacks in the preceding ten steps, $\times$ timeout and reassignment. **(a)** Under social plasticity, a_0 and a_5 converge on the same fight and jointly contribute to the successful kill as their bond strengthens ($\overline{W}: 0.32 \rightarrow 0.47$). **(b)** Under orchestration, agents express an intention to collaborate, but without learned relational state this does not translate into joint execution: a_0 does not join the fight and a_5 secures the kill alone before reassignment. Full timelines in Appendix Figures 11 and 12.

Social plasticity is key for more challenging cooperative task. Compared with the central orchestrator, social plasticity achieves a higher task return (548 vs. 505, rows f–g, Table 1), while cooperative milestone completion remains similar (4.8 vs. 4.7%). This suggests that social plasticity supports cooperation for more demanding tasks, which lead to a higher reward. A qualitative example is shown in Figure 4, which illustrates collaboration on a more challenging cooperative milestone: jointly killing a zombie. Opportunities for collaboration arise under both coordination mechanisms, but under central orchestration they do not translate into successful joint execution, and the cooperative fails. Under social plasticity, by contrast, strongly bonded agents converge on the same fight and act together without explicit assignment. This example illustrates how learned relational state can be expressed directly in coordinated execution.

5.2 RQ2: How do social interaction regimes shape learned bonds?

Using Gemma-E4B, we vary the explicit co-firing cues $\mathcal{K} \subseteq \{\text{comm}, \text{obs}, \text{imit}\}$, evaluating each cue alone and in combination against a *Spatial engagement* condition with no explicit social acts, in which bonds form only through engaged co-location. Engaged co-location contributes to co-firing in all conditions.

Bond strength reflects interaction frequency, not necessarily coordination value. Different co-firing cues induce distinct relational structures. Communication is selected most often (82.7% of social-action opportunities) and produces the strongest bonds ($\overline{W} = 0.34$), yet it yields the lowest cooperative completion of any single cue (2.6%). Communication bonds are also the most tied to use of their act ($\rho = 0.65$–0.72): they are informative about who talks, not who cooperates. Observation and imitation, used far less often, keep cooperative completion close to the *Spatial engagement* condition (4.8% and 4.6% against 5.7%), and no explicit cue improves on it. Thus, bond magnitude

Table 2: **RQ2: How do social interaction regimes shape learned bonds?** Rows vary the cues that define co-firing; *Spatial engagement* forms bonds from engaged co-location alone. Task return excludes all social-act rewards. Per cue: share of social acts, share of positive bond change attributed to it (ΔW), and Spearman ρ between pairwise bond and use of that act; $\overline{W}$ is the mean end-of-episode bond; plots in Appendix F.4.

	Task return ↑	Solo % ↑	Coop. % ↑	Mean W	Cue	Act use %	Attributed ΔW %	ρ
Spatial engagement	**299** ± 29	**27.3** ± 5.5	**5.7** ± 0.4	0.11	–	–	–	–
comm	236 ± 65	23.7 ± 1.7	2.6 ± 1.3	0.34	comm	**82.7** ± 1.0	**75.9** ± 1.0	0.65 ± 0.08
obs	298 ± 37	26.8 ± 1.7	4.8 ± 2.9	0.25	obs	21.6 ± 1.1	46.0 ± 5.0	0.57 ± 0.21
imit	284 ± 33	24.7 ± 2.3	4.6 ± 0.0	0.15	imit	5.3 ± 3.4	13.4 ± 5.7	0.35 ± 0.11
comm + obs	212 ± 82	21.2 ± 5.5	2.4 ± 3.0	**0.38**	comm	73.5 ± 2.9	61.7 ± 3.1	**0.72** ± 0.07
					obs	3.9 ± 1.1	9.7 ± 2.1	0.53 ± 0.05
comm + obs + imit	214 ± 54	26.3 ± 2.3	1.5 ± 0.4	0.37	comm	75.7 ± 2.7	65.9 ± 2.6	0.71 ± 0.16
					obs	3.4 ± 0.4	6.8 ± 1.0	0.31 ± 0.13
					imit	0.0 ± 0.0	0.0 ± 0.0	0.21 ± 0.00

alone is not a measure of relational utility: the behaviour that causes agents to co-fire determines what information the resulting bond encodes.

More social signals can reduce relational quality through action competition. When multiple cues are available, they compete for a single social-action slot. Communication consequently dominates the full comm + obs + imit condition, accounting for 75.7% of social actions, while observation falls to 3.4% and imitation is almost never selected. The resulting policy therefore does not combine complementary signals so much as collapse toward its most frequently selected one, and the combined-cue conditions reach the lowest cooperative completion (2.4% and 1.5%). Adding channels is thus not sufficient: how cues are selected or composed is part of relational learning.

5.3 RQ3: How do memory and learned relational state carry social experience into long-term cooperation?

Social experience can persist in two forms: an agent's cognitive memory and its learned relational graph. RQ3 asks how these two carriers of past interaction shape long-term cooperation. In Phase A, we run nine pairs independently through Chambers 1–2 for three episodes and select three pairs that successfully complete the cooperative Chamber 2 milestone. These runs accumulate both pair-specific memories and learned bonds. In Phase B, we bring the three selected pairs together into a six-agent team starting from Chamber 3 and independently intervene on the two carriers of prior social experience: memory is either retained or reset, and bonds are either retained or replaced by a flat graph with the same total weight. This yields full transfer, memory only, bonds only, and fresh-start conditions. Neither representation is frozen: agents continue updating their memories and the bond graph through social plasticity throughout Phase B, retaining the abilities of social plasticity.

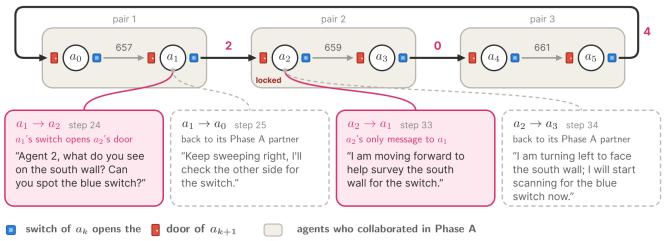


Figure 5: **Retaining both memory and social bonds favours exploitation of previously successful partners.** Full-transfer condition in Chamber 3. Grey arrows connect former Phase-A partners, while black arrows indicate cross-pair dependencies introduced by the switch task. Former partners exchange 657–661 messages, whereas task-relevant cross-pair links carry only 0–4, illustrating continued exploitation of established partnerships after team recomposition.

We evaluate each condition on two axes. *Cooperative performance* (task return and cooperative completion) measures which form of retained social experience best supports the new team. *Partner preference*, the share of an agent's messages addressed to its Phase-A partner, measures whether agents stay with their former partner; we compare it with the preference implied by its bond table alone (Table 3, Appendix F.1).

Table 3: **RQ3: partner transfer** (Gemma-E4B, +Social plasticity). *Kept from Phase A* sets only the starting state (memory: own skill library, episodic memory, and curriculum, or empty; bonds: learned Phase A graph or flat with equal total weight); everything keeps updating during Phase B. Rask return excludes message pay, and *Coop. %* counts the twelve Chamber 3–5 milestones reachable from a Chamber 3 start. *Bond-implied*: partner share of the bond table at each episode start (chance 0.20). Per-pair values in Appendix Table 11.

	Kept from Phase A				Partner pref.	
Condition	Memory	Bonds	Task return ↑	Coop. % ↑	Observed	Bond-implied
(a) Full transfer	kept	kept	237 ± 115	6.9 ± 2.3	0.55 ± 0.12	0.40
(b) Memory only	kept	reset	128 ± 76	4.2 ± 1.1	0.18 ± 0.23	0.27
(c) Bonds only	reset	kept	**369** ± 63	**8.3** ± 1.1	0.14 ± 0.14	0.31
(d) Fresh start	reset	reset	<u>271</u> ± 14	<u>7.9</u> ± 0.7	0.34 ± 0.26	0.28

Retained memory and bonds favour exploitation, while bonds alone balance exploitation and exploration. When agents retain both their Phase-A memory and bonds, they tend to keep exploiting the previously successful partnership (Figure 5). This produces the strongest and most persistent preference for the old partner, but can hurt performance when the new task requires cooperation with other teammates, which is the case in Chamber 3 (Table 3). By contrast, keeping only the bonds preserves a representation of prior social experience without locking agents into the same partner. As social plasticity continues, the graph can rewire toward new collaborators required by the current task. In this sense, full memory and bonds favour exploitation of established relationships, while bonds alone better balance exploitation and exploration after team recomposition.

The bond graph provides a better distillation of social experience than episodic memory for long-horizon plasticity and collaboration. Phase A social experience is retained in two complementary forms: episodic memory preserves detailed records of past interactions, while the bond graph compresses repeated interaction into a low-dimensional, partner-specific representation. The bonds-only intervention tests whether this compressed relational state can carry useful information from past collaboration once the richer episodic history is removed. Agents retain the relational structure learned from previously successful partners, while continued social plasticity allows those bonds to be revised as new interactions accumulate in Phase B. In contrast to storing the full history of *what* happened with a partner, the graph retains a compact signal of *with whom* productive social experience accumulated. The bond graph therefore acts as a useful distillation of past social experience for long-horizon collaboration, preserving partner-specific structure while remaining plastic enough to adapt when the cooperative context changes.

6 Discussion and conclusion

Coordination in multi-agent systems is typically specified per task, by a fixed topology, a dialogue protocol, or a planner that assigns work. Our findings point to a second axis of design, namely *with whom* agents have a relationship. Bonds then become part of training, a system variable that shapes both learning and coordination. Cooperation no longer has to be rebuilt from a global plan, as accumulated relational structure offers a decentralised form of coordination that complements orchestration and removes the reliance on a single central component. Our work also shows that the choice of social cues is itself a design decision, since communication is not always the most informative social interaction regime.

The results point to a broader implication. As agents increasingly find and use channels to one another, it becomes important to know which relationships form between them. A learned relational graph makes that structure explicit, and therefore inspectable, auditable, and correctable, when agents' memory is augmented with relational state. For lifelong teams coordination, retained memory and bonds favour exploiting established partnerships, while bonds alone balance exploitation and exploration: the bond graph acts as a compact distillation of social experience that stays plastic enough to rewire toward the collaborators a new task requires. Letting the fast relational layer correct what memory retrieves, for instance by weighting partner-related recall with current bonds, is a natural next step. Multi-agent systems can thus learn not only policies and plans, but the relationships that shape future collaboration.

ETHICS STATEMENT

All experiments take place in a simulated environment and involve no human subjects or personal data. The mechanism studied here learns persistent relationships between agents, which raises two concerns for deployed multi-agent systems: learned structure may diverge from the organisation a designer intended, and early, possibly spurious, partner preferences can become self-reinforcing and concentrate influence among a few agents. At the same time, the relational state is explicit and inspectable, which supports auditing and intervention. We discuss these points, together with computational cost, in Section J.

REPRODUCIBILITY STATEMENT

The Appendix provides a more comprehensive overview of the implementation details. WIRE's chamber structure, milestone schedule, and reward terms are specified in Section A; the agent architecture, action interface, and social acts in Section B; pseudo-code in Section C; the orchestration baseline in Section D; all hyperparameters, including the number of PPO updates per run, in Section E; and prompt templates in Section H. Code: `https://anonymous.4open.science/r/wired-together-2C07/`. A project page with videos of every condition is included in the supplementary material (open `index.html`).

AI USE STATEMENT

AI tools, primarily Claude and OpenAI models, were used in the research and writing of this work. *Research:* implementing and debugging parts of the codebase; writing scripts for data extraction, processing, and visualisation. *Writing:* revising text, and LaTeX formatting. *Figures:* Figure 3 was generated with the assistance of an AI image-generation tool and selected and incorporated by the authors. All research questions, experimental decisions, and scientific claims are the authors' own, and all AI-assisted outputs were reviewed and validated by the authors.

REFERENCES

Christopher Amato and Frans Oliehoek. Scalable planning and learning for multiagent pomdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Anthropic Frontier Red Team. Patterns and problems in emerging multiagent systems. `https://www.anthropic.com/research/multiagent-systems`, August 2026. Accessed: 2026-09-22.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4): 819–840, 2002.

Aman Bhargava, Mohammad R Rezaei, and Milad Lankarany. Gradient-free neural network training via synaptic-level reinforcement learning. *AppliedMath*, 2(2):185–195, 2022.

Robert Boyd, Peter J Richerson, and Joseph Henrich. The cultural niche: Why social learning is essential for human adaptation. *Proceedings of the National Academy of Sciences*, 108(supplement_2): 10918–10925, 2011.

Richard E Brown. Donald o. hebb and the organization of behavior: 17 years in the writing. *Molecular Brain*, 13(1):55, 2020.

Matthew Chalk, Gasper Tkacik, and Olivier Marre. Training and inferring neural network function with multi-agent reinforcement learning, 2019.

Jize Chen and Changhong Wang. Reaching cooperation using emerging empathy and counter-empathy. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems*, pp. 746–753, 2019.

Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. *Advances in neural information processing systems*, 33: 10707–10717, 2020.

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. *Advances in neural information processing systems*, 38:165025–165059, 2026.

Yubo Dong, Xukun Zhu, Zhengzhe Pan, Linchao Zhu, and Yi Yang. Villageragent: A graph-based multi-agent framework for coordinating complex task dependencies in minecraft. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 16290–16314, 2024.

Robin IM Dunbar. The social brain hypothesis. *Evolutionary Anthropology: Issues, News, and Reviews: Issues, News, and Reviews*, 6(5):178–190, 1998.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35: 18343–18362, 2022.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

Olivier L Georgeon, James B Marshall, and Simon Gay. Interactional motivation in artificial systems: Between extrinsic and intrinsic motivation. In *2012 IEEE international conference on development and learning and epigenetic robotics (ICDL)*, pp. 1–2. IEEE, 2012.

Ryan Greenblatt, Ajeya Cotra, and Hjalmar Wijk. Brief independent investigation of agents' behavior, reasoning and collaboration in the openai / hugging face hacking incident. `https://www.redwoodresearch.org/research/hugging-face-incident`, 08 2026. METR and Redwood Research.

Edward T Hall. A system for the notation of proxemic behavior. *American anthropologist*, 65(5): 1003–1026, 1963.

Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

Dylan Hillier, Cheston Tan, and Jing Jiang. Social learning through interactions with other agents: A survey. *arXiv preprint arXiv:2407.21713*, 2024.

William Hoppitt and Kevin N Lala. Social learning: an introduction to mechanisms, methods, and models. In *Social learning*. Princeton University Press, 2013.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning*, pp. 2961–2970. PMLR, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, pp. 3040–3049. PMLR, 2019.

Beomseok Kang and Saibal Mukhopadhyay. Forecasting evolution of clusters in game agents with hebbian learning. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2023.

Rachel L Kendal, Neeltje J Boogert, Luke Rendell, Kevin N Laland, Mike Webster, and Patricia L Jones. Social learning strategies: Bridge-building between fields. *Trends in cognitive sciences*, 22 (7):651–665, 2018.

Łukasz Kuśmierz, Takuya Isomura, and Taro Toyoizumi. Learning with three factors: modulating hebbian plasticity with errors. *Current opinion in neurobiology*, 46:170–177, 2017.

Alberto Lazari, Piergiorgio Salvan, Michiel Cottaar, Daniel Papp, Matthew FS Rushworth, and Heidi Johansen-Berg. Hebbian activity-dependent plasticity in white matter. *Cell Reports*, 39(11), 2022.

Joel Z Leibo, Edgar A Dueñez-Guzman, Alexander Vezhnevets, John P Agapiou, Peter Sunehag, Raphael Koster, Jayd Matyas, Charlie Beattie, Igor Mordatch, and Thore Graepel. Scalable evaluation of multi-agent reinforcement learning with melting pot. In *International conference on machine learning*, pp. 6187–6199. PMLR, 2021.

Mircea Tudor Lică, Ojas Shirekar, Baptiste Colle, and Chirag Raman. Mindforge: Empowering embodied agents with theory of mind for lifelong cultural learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, Mrinal Kalakrishnan, Franziska Meier, Chris Paxton, Alexander Sax, and Aravind Rajeswaran. OpenEQA: Embodied Question Answering in the Era of Foundation Models. pp. 16488–16498. URL https://openaccess.thecvf.com/content/CVPR2024/html/Majumdar_OpenEQA_Embodied_Question_Answering_in_the_Era_of_Foundation_Models_CVPR_2024_paper.html.

Mikel Malagón, Josu Ceberio, and Jose A. Lozano. Craftium: Bridging flexibility and efficiency for rich 3d single- and multi-agent environments. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 42765–42794, 2025.

METR. Brief independent investigation of agents' behavior, reasoning and collaboration in the openai / hugging face hacking incident. https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/, 08 2026.

Yelyzaveta Mukeriia, Jan Treur, and Sophie Hendrikse. A multi-adaptive network model for human hebbian learning, synchronization and social bonding based on adaptive homophily. *Cognitive Systems Research*, 84:101187, 2024.

Kamal K Ndousse, Douglas Eck, Sergey Levine, and Natasha Jaques. Emergent social learning via multi-agent reinforcement learning. In *International conference on machine learning*, pp. 7991–8004. PMLR, 2021.

Shohei Ohsawa, Kei Akuzawa, Tatsuya Matsushima, Gustavo Bezerra, Yusuke Iwasawa, Hiroshi Kajino, Seiya Takenaka, and Yutaka Matsuo. Neuron as an agent, 2018.

Jordan Ott. Giving up control: Neurons as reinforcement learning agents. *arXiv preprint arXiv:2003.11642*, 2020.

Jongho Park, Vasilis Kontonis, Shivam Garg, Akshay Krishnamurthy, and Dimitris Papailiopoulos. Scaling discovery through test-time communication, 2026. URL https://arxiv.org/abs/2609.21032.

Joachim Winther Pedersen and Sebastian Risi. Evolving and merging hebbian learning rules: increasing generalization by decreasing the number of rules. In *Proceedings of the genetic and evolutionary computation conference*, pp. 892–900, 2021.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artificial intelligence review*, 24(1):33–60, 2005.

Mark Sandler, Max Vladymyrov, Andrey Zhmoginov, Nolan Miller, Tom Madams, Andrew Jackson, and Blaise Agüera Y Arcas. Meta-learning bidirectional update rules. In *International Conference on Machine Learning*, pp. 9288–9300. PMLR, 2021.

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

Michael Tomasello, Ann Cale Kruger, and Hilary Horn Ratner. Cultural learning. *Behavioral and brain sciences*, 16(3):495–511, 1993.

Nenad Tomašev, Matija Franklin, and Simon Osindero. Intelligent ai delegation. *arXiv preprint arXiv:2602.11865*, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Zhipeng Wang and Mingbo Cai. Reinforcement learning applied to single neuron. *arXiv preprint arXiv:1505.04150*, 2015.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive Benchmarking Multi-modal Large Language Models for Vision-Driven Embodied Agents. URL http://arxiv.org/abs/2502.09560.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(1):4, 2024.

APPENDIX ROADMAP

The appendix is organised as follows. Section A details the WIRE environment, its milestones, and its reward structure, and Section B documents the agent architecture, the social acts, and the action space. Section C gives the full pseudo-code, and Section D specifies the centralised orchestration baseline. Section E states the RL objectives and lists all hyperparameters. Section F reports additional results: per-milestone completion times, the relation between bonds and behaviour, the full partner-transfer analysis of RQ3 (Section F.1), the deliberation-interval sweep (Section F.3), the RQ2 bond–completion trade-off (Section F.4), and the backbone capability analysis (Section F.5). Section G shows coordination timelines contrasting social plasticity with orchestration, and Section H reproduces the prompt templates. Section I gives the biological motivation for the rule and the neuron–agent correspondence, and Section J discusses broader impact.

A ENVIRONMENT

This section motivates the choice of simulator and the need for a new environment, then details WIRE's chamber structure and its reward system.

A.1 ENVIRONMENT SELECTION

Studying persistent social structure requires an embodied 3D setting with first-person partial observability, sparse and delayed reward, and a large action space. Existing options fall short on at least one axis. Multi-agent social benchmarks such as Melting Pot (Leibo et al., 2021) offer rich social dilemmas but in 2D gridworlds without first-person embodiment, while Minecraft-based frameworks such as MineDojo (Fan et al., 2022) are visually rich but built on a closed-source commercial game, which limits how far their mechanics can be modified and makes them computationally costly.

Why Craftium. Craftium (Malagón et al., 2025) is a fully open-source 3D voxel world built on the Luanti (formerly Minetest) engine, supporting single- and multi-agent scenarios through standard Gymnasium and PettingZoo interfaces. It targets rich 3D navigation, building, and resource-collection tasks, is computationally efficient relative to Minecraft-based frameworks, and is customisable through a Lua API. Its main drawback is that it is a framework rather than a fixed benchmark: specific social scenarios must be designed, implemented, and validated.

From OpenWorld to five chambers. Our initial design used Craftium's OpenWorld environment with three heterogeneous roles (Engineer, Hunter, Guardian) matching three skill tracks (tools, hunt, defend). Pilot runs showed that at small population sizes and short horizons agents rarely co-located, so the co-activity signal that drives the plasticity rule fired too rarely for bonds to form. Scaling up population and episode length would have addressed this, but exceeded our compute budget. We therefore designed *WIRE*, a curriculum that retains Craftium's voxel-world affordances (3D navigation, resource collection, combat) but enforces co-location and shared task progression at every stage, so that the mechanism is evaluated where co-activity is reliably present and without requiring much longer horizons. Studying the mechanism in genuinely open environments at scale remains an open direction.

Why a sequential cooperative environment. Existing cooperative Minecraft benchmarks such as VillagerBench (Dong et al., 2024) evaluate collaboration across *separate* scenarios (construction, farm-to-table cooking, and an escape room), each a self-contained task solved in isolation. This is the right design for measuring coordination on any single task, but not for our question: a persistent social structure can only be studied if it persists, so bonds must form, carry forward, and be reused across tasks rather than reset at every scenario boundary. WIRE is therefore a single sequential curriculum whose chambers build on one another. Skills acquired in Chamber 1 are the building blocks of later cooperative objectives, and bonds formed while solving one chamber carry into the next, where they shape how the team approaches a harder, more coordination-dependent task. Cooperation, and the graph that encodes it, thus accumulate over the episode, which is the regime the plasticity rule targets and one that a collection of independent tasks cannot expose.

A.2 Environment structure

WIRE consists of five sequential chambers connected by one-way transitions and traversed by a fixed population of N agents. Each chamber gates progression on a shared objective, so the team advances together and co-location is enforced at every stage rather than left to emerge. Rescue teleports fire at fixed checkpoints, 20, 40, 60, and 80% of the horizon (every 200 steps of the default 1,000-step horizon): a team still in the corresponding chamber is moved into the next one. Because the last checkpoint places the team in Chamber 5, where damage is lethal, the boss typically eliminates the team shortly afterwards, and most episodes end near step 800 rather than at the horizon. Rescue teleports keep the curriculum moving but grant none of the skipped chamber's remaining milestones, and chamber-entry milestones (M12, M16, M20) are credited only when the team advances before the timer. How often teams progress on their own rather than by teleport is analysed in Section F.

Chamber 1 (solo skill acquisition). A shared resource room in which each agent independently practises exploring, digging, item pickup, and combat against passive animals (M1–M7). There is no interdependence here; the chamber ensures agents enter Chamber 2 with the primitive skills available.

Chamber 2 (cooperative resource acquisition). The agents face anvils that, once broken, dispense and auto-equip diamond equipment (M8–M11). An anvil yields only when at least two agents dig it simultaneously, so acquiring the equipment requires synchronising actions in time rather than dividing work in space. This simultaneity constraint makes the chamber cooperative by construction: no agent can progress alone.

Chamber 3 (communication under partial observability). The N agents are placed in individual isolation cells (M12) arranged in a single dependency cycle: each agent's switch releases not its own door but the next agent's, so agent i controls the door of agent $(i \bmod N) + 1$ and the chain closes on itself ($1 \to 2 \to \ldots \to N \to 1$). No agent can free itself or observe the cell its own switch governs, so release propagates only if every agent acts on a teammate's behalf, and an agent learns that its door has opened only through the environment. Escaping the cycle therefore depends on agents communicating their state and intent across cells (M13–M14), and the chamber completes once all agents regroup in the communal room (M15). This is the chamber in which the communication channel is load-bearing.

Chamber 4 (team combat). A combat arena with three zombies (M16–M19). The chamber rewards clearing all hostiles (M18) and grants an additional team bonus when every agent finishes intact having dealt damage (M19). Combat is deliberately non-lethal: zombies cannot kill an agent, and an agent that would have died instead incurs a -10 penalty. Chamber 4 is thus a low-stakes rehearsal of the joint engagement that the Chamber 5 boss fight demands under real threat.

Chamber 5 (cooperative boss fight). The terminal chamber is a single encounter with a powered zombie whose health is raised from the default 20 to 60 (M20–M24). Two properties make cooperation most consequential here. First, damage is real: a lethal hit is a permanent death, and the episode ends when the boss falls or the whole team is down. Second, the boss's health cannot be depleted by one agent within the chamber's time budget, so the team must concentrate sustained damage while surviving. The diamond gear earned in Chamber 2 equips the team for this fight, a direct instance of an earlier chamber's outcome carrying into a later one. Credit for the kill and the survival bonus is gated by a minimum cumulative-damage threshold, so an agent must have damaged the boss to share the reward.

A.3 Milestone structure

The per-agent reward r_i^t of Section 3.3 is driven primarily by a milestone ladder (Table 4). Milestones are fired by a Lua-side event system in the Luanti server, which writes to an event log polled by the Python training loop at every step. When a milestone fires, its reward is credited to its *contributors*, the agents whose actions triggered the event. The completion metrics of the main text exclude the communication track and are computed per agent: *Solo* counts every (agent, milestone) credit among the eight Chamber 1 milestones over the $7N+1$ achievable (Door 1 goes to one agent), and *Coop.* the credits among the 17 milestones of Chambers 2–5 over the $17N$ achievable.

Track structure. Milestones are organised into six tracks. The first five correspond to the chambers, with per-milestone rewards that broadly increase with the chamber's coordination demand. The sixth is a *communication track* that rewards sustained valid targeted communication within each chamber.

Team-level milestones and contributor credit. Several milestones (M8, M9, M15, M18, M19, M22–M24) are team-level events: they fire once when a global condition is met (an anvil broken by two agents simultaneously, all agents in the communal room, all mobs killed, the boss defeated). The event log records every agent contributing to the condition, and each receives the full milestone value in its training reward. For analysis only, the reward is split equally across contributors, with combat milestones (M18, M21–M23) split by damage share so that credit tracks damage dealt rather than presence.

Communication reward. Communication is rewarded at two tiers. Each valid message earns a small base payment, capped per agent and episode to prevent reward farming through chat spam (Table 5); in addition, an agent that emits four valid messages within a chamber triggers that chamber's communication milestone once. Validity rules (minimum length, no duplicates, at least two steps between rewarded messages) and target rules (no self-targeting, no broadcast, no missing target) gate both tiers: invalid messages earn nothing, and messages with malformed targets are rerouted but earn their sender no base payment.

A.4 Auxiliary rewards and penalties

Beyond the milestone ladder, the per-agent reward contains a small set of auxiliary terms (Table 5). The RL layer receives the full reward. The plasticity rule receives it in two parts: death and would-die penalties are routed separately as r_i^{death} into the failure-gated decay term, and the salience signal r_i^{bond} counts only rewards earned in the cooperative chambers (Section 3.1, Algorithm 1). The two death-related penalties encode the difference between the rehearsal combat of Chamber 4 and the lethal boss fight: in Chambers 1–4 an agent cannot die, and a hit that would have been fatal costs -10 (at most once per agent per episode); in Chamber 5 a fatal hit removes the agent for the rest of the episode and costs -50. The pitch-futility penalty supports the camera clamps of Section B: vertical camera movement is limited to two `LookDown` steps and one `LookUp` step from level, and an action that attempts to exceed the clamp is redirected to `NoOp` and costs -1, discouraging the sky- and floor-staring loops observed in early runs.

B Agent architecture details

Each WIRE agent builds on the MindForge cognitive stack (Lică et al., 2025). This appendix documents its modules as used in WIRE, the action interface for zero-shot and RL-fine-tuned agents, and the environment-side conventions that translate an action into world ticks.

Cognitive modules. Six modules feed the action-selection prompt. *Perception beliefs* describe what the agent sees in action-relative spatial terms and flag unenacted action–target pairs as intrinsic-motivation targets (Georgeon et al., 2012). *Partner beliefs* keep one entry per teammate summarising their chamber, gear, commitments, and reported schemes. *Interaction beliefs* extract shared actionables from recent conversation (role assignments, switch and door status, combat coordination, team-level warnings). *Task beliefs* frame the current task as a scheme pairing an intended action with its expected observation. *Episodic memory*, a ChromaDB vector store keyed on task embeddings, retrieves past attempts at a 70/30 success-to-failure ratio, so the agent sees mostly positive examples without losing the failure signal. *Skills* store successful action–thought pairs in a separate vector store for retrieval on similar tasks.

Auto-curriculum and critic. An auto-curriculum module proposes the next concrete task at every step, conditioned on the agent's current chamber, fired milestones, inventory, and a brief question–answer pass. Validity filters reject proposals that reference fixtures outside the agent's current chamber or contain no actionable keyword, replacing them with a chamber-appropriate fallback. A critic evaluates task success every twenty steps from the frame, last action, reward, status, inventory, and current task. Its verdict has three effects: successful tasks promote their action into the skill store, repeated failures trigger task reassignment, and every verdict is appended to episodic memory.

Table 4: **Milestone reward schedule for WIRE.** Per-agent (P) milestones fire at most once per agent per episode; team-level (T) milestones fire once per episode and credit the full reward to every contributing agent.

ID	Track	Trigger	Reward	Type
Chamber 1: solo skill acquisition				
M1	ch1_solo	Move more than 5 blocks from spawn	10	P
M2	ch1_solo	Dig any 3 blocks	30	P
M3	ch1_solo	Pick up 3 dropped items	30	P
M4	ch1_solo	Dig 5 wood blocks	50	P
M5	ch1_solo	Kill 1 animal	50	P
M6	ch1_solo	Kill 2 animals	80	P
M7	ch1_solo	Dig 3 stone blocks	60	P
M_door1	ch1_solo	First Ch1 milestone unlocks Door 1 (unlocker only)	50	P
Chamber 2: cooperative resource acquisition				
M8	ch2_anvils	Break first anvil	40	T
M9	ch2_anvils	Break second anvil	40	T
M10	ch2_anvils	Diamond sword auto-equipped	50	P
M11	ch2_anvils	Diamond chestplate auto-equipped	30	P
Chamber 3: communication under partial observability				
M12	ch3_switches	Enter isolation cell	20	P
M13	ch3_switches	Press cell switch	40	P
M14	ch3_switches	Cell door opened by teammate's switch	60	P
M15	ch3_switches	All agents in communal room	100	T
Chamber 4: team combat				
M16	ch4_combat	Enter Chamber 4	30	P
M17	ch4_combat	First mob kill	60	P
M18	ch4_combat	All mobs killed	150	T
M19	ch4_combat	All agents survive having dealt damage	100	T
Chamber 5: cooperative boss fight				
M20	ch5_boss	Enter Chamber 5	50	P
M21	ch5_boss	First damage dealt to boss	80	P
M22	ch5_boss	Boss at 50% health	120	T
M23	ch5_boss	Boss defeated	300	T
M24	ch5_boss	All agents alive at boss defeat	250	T
Communication track				
M_comm_ch1	comm	Sustained valid messaging in Chamber 1	10	P
M_comm_ch2	comm	Sustained valid messaging in Chamber 2	10	P
M_comm_ch3	comm	Sustained valid messaging in Chamber 3	20	P
M_comm_ch4	comm	Sustained valid messaging in Chamber 4	10	P
M_comm_ch5	comm	Sustained valid messaging in Chamber 5	10	P

Table 5: **Auxiliary rewards and penalties.** All values are per agent. The communication milestones are listed in Table 4.

Event	Value	Limit / scope
Would-die hit (non-lethal combat)	-10	once per agent per episode; Ch1–4
Death (lethal combat)	-50	terminal for the agent; Ch5
Futile pitch action (beyond clamp)	-1	per action; all chambers
Valid targeted message	$+0.5$	first 50 valid messages per agent per episode

Action selection. The action-selection prompt (Section H) combines the first-person frame, current task, recent beliefs, retrieved skills and episode summary, live chamber state, inventory, position, status, and a hard-grounded list of the chambers the agent has actually visited in the episode. The last item rules out a failure mode in which the model claims to be in a chamber it has never entered. When the social module is active, its rendered directive (Algorithm 2) is inserted into the same prompt. A zero-shot agent returns a JSON object with fields `thoughts`, `action`, `communication`, and `communication_target`. An RL-fine-tuned agent receives the same context through a thoughts-only variant of the prompt; its LoRA-adapted actor then scores each candidate action string as a

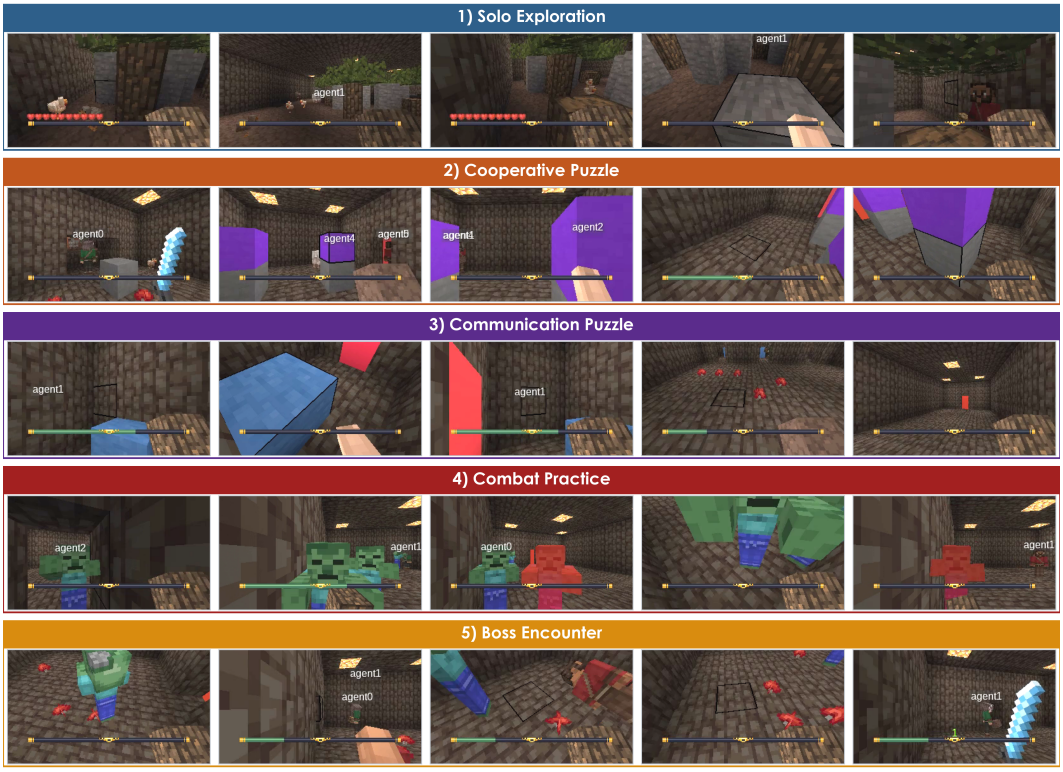


Figure 6: **Agent observations across WIRE.** Representative first-person observations from Chambers 1–5, corresponding to the curriculum in Figure 3: individual skill acquisition, cooperative resource use, communication under partial observability, team combat, and the boss encounter.

continuation of the prompt and the emitted thoughts (Equation (11)), so reasoning and messages condition the action but receive no direct policy-gradient objective.

Social acts. Communication is always targeted: each message is addressed to exactly one teammate through `communication_target`, with no broadcast channel. The social module never routes messages itself; its directive reaches the agent only through the action prompt (Algorithm 2). In every arm outside the RQ2 co-firing study (Section 5.2), communication is the only social act and the message is a mandatory per-step field: the prompt requires a non-empty, actionable message to a named teammate at every step, and co-firing credits the `comm` channel alone.

Choice-mode social acts (RQ2). The co-firing arms replace the mandatory message with a per-step choice. *Observe* and *imitate* are issued through output fields rather than the action space: the response JSON gains `social_act` $\in$ {`communicate`, `observe`, `imitate`, `none`}, a `social_target` of the form `agent_N`, and an `imitate_horizon` $h \in \{1, \ldots, 5\}$. The game action is emitted in the same response, so a social act never costs an environment step and the primitives of Table 6 are unchanged. Each arm's *menu* is the subset of acts its prompt offers, and only menu entries appear in its schema. The *Spatial engagement* condition receives an empty menu, so its agents are mute, and their bonds form only through engaged co-location. Because `communicate` is one entry among several and `none` is a legitimate choice, the message is not mandatory in these arms, which makes the act mix a measured quantity rather than a fixed one.

Observe and imitate. Both non-verbal acts are delivered to the initiator one step later. `observe`(j) returns j's position, chamber, status, and inventory together with j's perception, task, and interaction beliefs. `imitate`(j, h) returns j's last h actions (oldest first) with j's task and position, and instructs the agent to re-enact the sequence only if their situations match. Nothing is auto-executed, so the `imit` channel is credited on *adoption* rather than on request: a co-firing event fires on each later step whose chosen action matches the next unconsumed element of the delivered sequence, in order, within a window of $3h$ steps (non-matching steps are tolerated). The horizon cap bounds the credit

Table 6: **The 22-primitive action space.** The hotbar group is masked from the RL candidate set (tool selection is automated on `Dig`), leaving 14 candidate actions for the constrained-decoding policy of Section 3.3.

Group	#	Actions
Movement	6	`MoveForward`, `MoveBackward`, `MoveLeft`, `MoveRight`, `Jump`, `Sneak`
Camera	4	`TurnLeft`, `TurnRight`, `LookUp`, `LookDown`
Interaction	3	`Dig`, `Place`, `Drop`
Hotbar	8	`Slot1`–`Slot8`
Idle	1	`NoOp`

one imitate act can earn relative to the single event a message produces. Like communication, both acts are delivery-symmetric: one event credits both directions of the pair, and the target is told in its next prompt who observed or imitated it. A missing, self-directed, or invalid target is rescued to the most strongly bonded teammate and logged, so act counts are never inflated by unroutable requests.

One act per step. The single-act limit is enforced in code rather than trusted to the prompt. The schema exposes one `social_act` slot; an act outside the arm's menu is degraded to `none`, so the affordance ablation holds even when the model names an act it was not offered; and unless the act is `communicate`, the message fields are blanked before routing, so an observe or imitate step stays silent even if the model filled them. Social-module recommendations are advisory: they are rendered into the action prompt and never set `social_act` or `social_target`. The one exception to the single-act rule is deferred imitation credit: a step may carry an adoption event from an earlier `imitate` alongside one newly initiated act.

Equal rewards of social cues. Because the update rule is reward-modulated, a cue that pays more than another would bias both the act mix and the bond growth that follows. The co-firing arms therefore price all three cues identically: observation and imitation are paid on exactly the communication schedule of Table 5 ($+0.5$ per valid act, at most 50 rewarded acts per agent per episode, at least two steps between rewarded acts of the same type), and each has per-chamber milestones (`m_obs_ch`N, `m_imit_ch`N) that fire once per agent at the same threshold of four acts in a chamber and with the same values as the communication milestones. A rescued act earns no base payment but still counts towards its chamber milestone, mirroring the treatment of a message with a malformed target: the act reached a real teammate, but naming oneself is not reinforced. Under this schedule, an arm's act mix reflects how useful each cue is to the agent rather than what it pays, and no cue injects reward that the others cannot. Because the co-firing arms change the communication regime as well as the menu, they are compared within their own family, with the communication-only menu as the reference condition.

Action space. The action space consists of the 22 discrete primitives in Table 6: movement, camera control, interaction, hotbar slot selection, and an explicit idle. The eight `Slot` actions are masked from the RL policy's candidate set: before every `Dig`, the environment automatically equips the agent's best available tool, which makes manual slot selection redundant and would otherwise add eight near-duplicate, low-value actions that inflate the policy's entropy. Zero-shot VLM conditions retain the full set.

C ALGORITHMS

Algorithm 1 Reward-modulated social-plasticity step at environment step t (Eqs. 6–8). Bonds are directed: W_{ij} is modulated by agent i's own reward. The always-on homeostatic term $-\lambda W_{ij}$ keeps bonds in the interior of $[0, 1]$: under constant co-activity c_{ij} and no reward, W_{ij} converges to $W^* = \eta_0 c_{ij}/(\eta_0 c_{ij} + \lambda)$. Sparse rewarded events raise bonds above this level, the clip bounding large consolidation steps, and the homeostatic term then relaxes them back towards W^*.

State: bonds $W \in [0,1]^{N\times N}$ (init. W_0, zero diagonal); eligibility traces $e \in \mathbb{R}_{\geq 0}^{N\times N}$ (init. 0); running max $r_{\max}$
Input: positions $\{\mathbf{p}_i\}$; social-act events $\mathcal{E}_t$ (messages, plus observe and imitate-adoption events in the co-firing arms; Section B); bondable rewards $\{r_i^{\text{bond}}\}$ (death-excluded, gated to cooperative chambers); death and would-die penalties $\{r_i^{\text{death}}\}$
▷ *Co-firing signal* (Eqs. 2–5)
$r_{\max} \leftarrow \max\big(r_{\max},\, \max_i |r_i^{\text{bond}}|\big)$
for each agent i **do**
  $g_i \leftarrow \text{clip}\big(\alpha\, |r_i^{\text{bond}}|/r_{\max} + (1-\alpha)\, \mathbf{1}[i \in \mathcal{A}],\, 0,\, 1\big)$
end for
for each ordered pair (i, j), $i \neq j$ **do**
  $s_{ij} \leftarrow \mathbf{1}\big[\|\mathbf{p}_i - \mathbf{p}_j\| \leq d\big]$ — spatial gate
  $c_{ij} \leftarrow \text{clip}\big(s_{ij}\, \max\big(c_0,\, g_i\, g_j\big) + \sum_{k\in\mathcal{K}} \delta_k\, \mathbf{1}[\text{event}_{ij}^{(k)}],\, 0,\, 1\big)$ — events credit both (i, j) and (j, i)
  if $c_{ij} < \epsilon_c$ **then** $c_{ij} \leftarrow 0$ — sparsify weak co-activity
end for
▷ *Bond update* (Eqs. 6–8)
for each ordered pair (i, j), $i \neq j$ **do**
  $e_{ij} \leftarrow \rho_e\, e_{ij} + c_{ij}$ — eligibility trace
  $\tilde{r}_i^{\text{death}} \leftarrow \min\big(|r_i^{\text{death}}|,\, r_{\text{cap}}\big)$ — capped failure signal
  $G_{ij} \leftarrow \eta_0\, c_{ij}\, (1 - W_{ij}) + \eta_+\, \dfrac{|r_i^{\text{bond}}|}{R}\, e_{ij}\, (1 - W_{ij})$ — floor + consolidation
  $D_{ij} \leftarrow \eta_-^{\text{d}}\, \dfrac{\tilde{r}_i^{\text{death}}}{R}\, e_{ij}\, W_{ij}$ — failure-gated decay
  $W_{ij} \leftarrow \text{clip}\big(W_{ij} + G_{ij} - D_{ij} - \lambda\, W_{ij},\, 0,\, 1\big)$
end for
Return W, $c(t)$

Algorithm 2 Social module: bond-conditioned deliberation for agent i (inference-time coupling of W, subsection 3.2). The help-request and help-offer selection rules are the decision policy the deliberation prompt instructs the VLM to follow; the output σ_i is a validated structured object that reaches the agent only through the action prompt; it never overrides message routing.

Input: bond row $\{W_{ij}\}_{j\neq i}$; windowed deltas $\Delta W_{ij} = W_{ij}(t) - W_{ij}(t - n_\Delta)$; inbox $\mathcal{I}_i^t$; recent state (last action, last reward, inventory, position)
if fewer than T_{soc} steps have passed since the last deliberation **and** a cached σ_i exists **then**
  return cached σ_i — (directive unchanged between deliberations)
end if
render each bond as $(W_{ij}, \Delta W_{ij})$, tagged rising / decaying when $|\Delta W_{ij}| > 0.02$, otherwise stable
$\sigma_i \leftarrow$ VLM deliberation over the bond table, $\mathcal{I}_i^t$, and recent state, choosing one of:
  request help: target the highest-bonded teammate plausibly able to help; draft the request message
  offer help: respond to inbox senders with high W_{ij} and rising ΔW_{ij}; generally ignore low-bond senders ($W_{ij} < w_{\min}$) unless the request is unambiguous and urgent
  stay on task: no request, no response
σ_i also records the bonds it referenced and a short rationale (audit trail for bond-driven decisions)
Return σ_i, rendered as a directive block for the action prompt

Algorithm 3 WIRE episode loop with reward-modulated social plasticity. The graph W and long-term memory (skill and episode stores) persist across the episodes of a run; only the environment and per-episode working memory are reset. The loop is written for RLFT agents. Zero-shot VLM agents emit their action directly in the JSON output instead of by constrained decoding and skip all buffer, replay, and update steps. No-graph baselines skip the social module, the plasticity step, and reward diffusion ($r'_i = r_i$). Under IPPO, each agent's local critic replaces the shared critic V_ψ. Experience sharing is active ($\rho = 0.3$) in the RLFT plasticity conditions.

Init: frozen base VLM with per-agent LoRA policies π_{θ_i}; shared critic V_ψ; social graph $W \leftarrow W_0(\mathbf{1}\mathbf{1}^\top - I)$; replay mixing strength ρ
for each episode **do**
  reset environment and per-episode working memory (task, beliefs); skills, episodic memory, and W persist
  for $t = 0, 1, 2, \ldots$ until termination **do**
    encode the joint state s^t once for the critic
    for each living agent i, **concurrently do**
      $o_i^t \leftarrow$ (visual frame, structured context, inbox $\mathcal{I}_i^t$)
      $\sigma_i \leftarrow \textsc{SocialDeliberate}\big(\{W_{ij}\}_{j\neq i}, \{\Delta W_{ij}\}_{j\neq i}, \mathcal{I}_i^t\big)$ (Alg. 2)
      generate reasoning th_i^t and a social act from o_i^t and the directive σ_i: a mandatory message (m_i^t, τ_i^t) by default, or one menu act (`communicate`, `observe`, `imitate`, `none`) in the co-firing arms
      $a_i^t \sim \pi_{\theta_i}(\cdot \mid o_i^t, th_i^t)$ by constrained decoding (Eq. 11)
    end for
    deliver messages to their targets; queue observe and imitate results for delivery to the initiator at $t{+}1$
    advance the environment under the joint action; collect per-agent rewards r_i^t and social-act events $\mathcal{E}_t$
    $W, c(t) \leftarrow \textsc{PlasticityStep}\big(\{\mathbf{p}_i\}, \mathcal{E}_t, \{r_i^{\text{bond}}\}, \{r_i^{\text{death}}\}\big)$ (Alg. 1)
    $r_i'^t \leftarrow (1-\gamma_d)\, r_i^t + \gamma_d \sum_{j\neq i} \overline{W}_{ij}\, c_{ij}\, r_j^t$ (Eq. 9)
    write $\big(o_i^t, a_i^t, r_i'^t, \log \pi_{\theta_i}(a_i^t \mid o_i^t, th_i^t)\big)$ to agent i's buffer $\mathcal{D}_i$; write $(s^t, \bar{r}'_t)$ to the critic buffer
    if agent i's buffer holds K transitions **then**
      if $\rho > 0$ and no snapshot is active **then** snapshot all buffers $\{\mathcal{D}_j\}_j$, giving one shared pre-update view per update cycle
      $\mathcal{S}_i \leftarrow$ draw $m = \lceil K\rho/(1-\rho) \rceil$ snapshot transitions from neighbours $j \neq i$ with $W_{ij} \geq 0.05$, in proportion to $\overline{W}_{ij}$, so shared samples form a fraction ρ of the batch; recompute GAE on copies under the source agent's baseline (Eq. 10)
      PPO update of adapter θ_i over $\mathcal{D}_i \cup \mathcal{S}_i$ with GAE advantages (Eq. 12); for shared transitions the clipped ratio $\pi_{\theta_i}/\pi_{\theta_j}$ is the importance weight
      clear $\mathcal{D}_i$; retire the snapshot once every living agent has updated
    end if
    if the critic buffer is full **then**
      update V_ψ on the team-mean diffused return (Eq. 13)
    end if
  end for
end for

D CENTRALISED ORCHESTRATION BASELINE

To compare decentralised, plasticity-based coordination with an explicit centralised alternative, we implement a non-embodied coordinator inspired by VillagerAgent (Dong et al., 2024). The coordinator replaces the social graph entirely: no bonds are learned or shown to the agents, and none of the graph couplings is active. Instead, a single coordinator maintains a global view of team progress and assigns objectives to individual agents. Everything else (backbone, cognitive stack, action interface, and communication) is identical to the matched Gemma-E4B zero-shot baseline.

Coordinator state. The coordinator observes team-level information: milestone completion state; each agent's chamber, position, health, and alive/dead status; the state of the chamber fixtures; the task each agent's own planner is currently pursuing; and its own graph of previously issued subtasks. No individual agent has this information, but it is not a superset of what the agents see: the coordinator never receives the first-person frames on which the agents act.

Task decomposition and assignment. The coordinator decomposes the remaining team objective into executable subtasks, held in a dependency graph so that a subtask becomes assignable only once its prerequisites have succeeded. Decomposition is event-driven rather than periodic: the plan is extended when a subtask completes or fails, when an idle agent has no executable subtask, or when the graph is exhausted, and never more often than every $T_{\mathrm{soc}}{=}8$ steps, the deliberation interval of the social module. Assignments bind only at the level of objectives. The coordinator determines *what* an agent works towards, and the agent's auto-curriculum is constrained to propose tasks that advance the assigned objective; *how* to achieve it (reasoning, primitive actions, and messages) remains the agent's own. Multi-agent subtasks are assigned only when fully crewed, and active assignments are retained until they complete or fail.

Progress verification and replanning. Each subtask is tied to one or more WIRE milestones, which serve as environment-grounded completion signals. Unlike the original method, which relies on worker self-reflection, the coordinator never asks agents to report their own success. A subtask fails if an assigned agent dies, if none of its milestones fires within $T_{\mathrm{fail}}{=}60$ steps of assignment, or if the team is teleported past the chamber its milestones belong to. Failed subtasks are retired rather than retried: the coordinator observes the failure and proposes a different route to the same goal.

Communication. The coordinator neither decides who communicates with whom nor routes messages. Agents keep the communication mechanism of the matched baseline, a mandatory targeted message at every step (Section B), so centralisation operates through task allocation rather than through control of the communication topology, as in VillagerAgent.

What persists, and what is matched. The two coordination mechanisms differ in what they carry forward. The coordinator's subtask graph is rebuilt at the start of every episode, so it holds no state across episodes, whereas the social graph W persists across the episodes of a run (Algorithm 3). This asymmetry is the property under study, not an implementation detail: orchestration represents *what the team should do now*, while the graph retains *with whom each agent has worked*. The agents' own skill and episodic memories persist identically in both conditions. The coordinator is invoked at most once every T_{soc} steps, the invocation interval of the social module, and the two conditions are matched in inference compute: averaged over runs, the orchestration condition uses 3.66×10^{17} inference FLOPs and the social-plasticity condition 3.67×10^{17}, estimated as in Section F.3 over all model calls.

Algorithm 4 Centralised orchestration baseline, run once per environment step.

Input: environment state s^t; milestone status $\mathcal{M}^t$; agent states $\{x_i^t\}_{i=1}^N$; subtask graph $\mathcal{G}$ with assignments $\mathcal{U}^t$
{Bookkeeping, every step}
complete a subtask when any of its milestones fires; fail it if an assigned agent died, no milestone fired within T_{fail} steps of assignment, or the team was teleported past its chamber
propagate failure to unstarted dependents; free the agents of finished subtasks (failed subtasks are retired, not retried)
$\mathcal{X}^t \leftarrow$ unstarted subtasks whose dependencies have succeeded; $\mathcal{F}^t \leftarrow$ alive, unassigned agents
if $\geq T_{\text{soc}}$ steps since the last decomposition **and** the plan needs extending ($\mathcal{G}$ exhausted, a subtask just finished, or $\mathcal{X}^t{=}\emptyset$ while $\mathcal{F}^t{\neq}\emptyset$) **then**
    coordinator proposes subtasks from s^t, $\mathcal{M}^t$, the agents' current tasks, and $\mathcal{G}$; each carries at least one completion milestone, its dependencies, eligible agents, and a required team size
    add the valid proposals to $\mathcal{G}$ and recompute $\mathcal{X}^t$
end if
if $\mathcal{X}^t \neq \emptyset$ **and** $\mathcal{F}^t \neq \emptyset$ **then**
    coordinator assigns executable subtasks to available agents with a per-agent role; multi-agent subtasks only if fully crewed; active assignments are retained
    $\mathcal{U}^{t+1} \leftarrow \mathcal{U}^t \cup$ accepted assignments
end if
constrain each assigned agent's auto-curriculum to its assigned objective
Return $\mathcal{U}^{t+1}$, rendered as directives in the agents' action prompts; each agent independently chooses its reasoning, primitive actions, and messages

E Learning objectives and hyperparameters

E.1 Full RL objectives

The RLFT conditions optimise per-agent LoRA adapters with PPO, under either MAPPO with a centralised critic or IPPO with independent critics (Section 3.3). Advantages use generalised advantage estimation on the diffused rewards $r'_{i,t}$ (Eq. 9):

$$\delta_{i,t} = r'_{i,t} + \gamma\, V_\psi(s^{t+1}) - V_\psi(s^t), \qquad \hat{A}_{i,t} = \sum_{l\geq 0} (\gamma\,\lambda_{\text{GAE}})^l\, \delta_{i,t+l}, \tag{12}$$

and the adapter θ_i maximises the clipped surrogate $\mathbb{E}\big[\min\big(\varrho_{i,t}\hat{A}_{i,t},\ \text{clip}(\varrho_{i,t}, 1{-}\epsilon, 1{+}\epsilon)\hat{A}_{i,t}\big)\big] + \beta_t\,\mathcal{H}[\pi_{\theta_i}]$, with likelihood ratio $\varrho_{i,t}$ computed over the constrained action tokens (Eq. 11); for transitions shared from teammate j, the denominator of $\varrho_{i,t}$ is j's behaviour probability (Section 3.2). Under MAPPO, the shared critic regresses the team-mean diffused return,

$$\bar{r}'_t = \frac{1}{|\mathcal{A}_t|}\sum_{i\in\mathcal{A}_t} r'_{i,t}, \qquad \mathcal{L}(\psi) = \mathbb{E}\Big[\big(V_\psi(s^t) - \hat{R}_t\big)^2\Big], \tag{13}$$

where $\mathcal{A}_t$ is the set of living agents and $\hat{R}_t$ the bootstrapped λ-return built from Eq. 12, with value targets clipped to $\pm\xi$ around the previous estimate. Under IPPO, each agent instead trains a private value head on its own normalised reward, with no state sharing.

Critic target and per-agent advantages. Under MAPPO, the critic is fit to the team-mean reward, while each agent's GAE uses its own diffused reward, so the TD residual $\delta_{i,t} = r'_{i,t} + \gamma V_\psi(s^{t+1}) - V_\psi(s^t)$ combines a value function fit to a different reward stream. As $\lambda_{\text{GAE}} \to 1$ the value terms telescope and the mismatch reduces to a state-dependent baseline, which leaves the policy gradient unbiased; at $\lambda_{\text{GAE}} = 0.95$ a residual bias remains, which we accept in exchange for a single critic shared across agents. For the same reason, reward normalisation is disabled under MAPPO: normalising $r'_{i,t}$ while V_ψ is fit on the raw team reward would mix scales inside $\delta_{i,t}$.

E.2 Reinforcement-learning hyperparameters

Table 7 lists the settings of the RL layer (Section 3.3), shared by all RLFT conditions. There is no separate training phase: adapters are updated online within the evaluation episodes, every $K = 64$

Table 7: **Reinforcement-learning hyperparameters**, shared by all RLFT conditions (IPPO and MAPPO). *On Gemma-E4B, LoRA is restricted to the text tower; the vision and audio towers receive no adapters, since the policy consumes token inputs only. †Only about 7% of the schedule is traversed within a run ($\beta_t \geq 0.046$).

Hyperparameter	Value
Base model and adapters	
Base VLM (frozen)	Qwen3.5-2B or Gemma-E4B
LoRA rank r / α / dropout	8 / 16 / 0.05
LoRA target modules	`q_proj`, `v_proj`*
Precision / grad. checkpointing	fp16 / on
Actor encoding cap	512 tokens
PPO	
Discount γ	0.995
GAE λ_{GAE}	0.95
Clip ϵ	0.2
Entropy coef. β_t	$0.05 \to 0.001$ over 500 updates†
Value-loss coef.	0.5
Actor value clip (IPPO)	1.0
Critic value clip ξ	10.0
Max gradient norm	0.5
PPO epochs E	2
Mini-batch size	4
Update interval K	64 steps
PPO updates per run (median)	37
Optimisation	
Optimiser	Adam
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Critic hidden width	256

environment steps. Runs last about 2,400 steps (three episodes that typically end near step 800; Section A), so each adapter receives a median of 37 PPO updates per run. The entropy coefficient is scheduled to anneal from 0.05 to 0.001 over 500 updates, but only about 7% of this schedule is traversed, so in practice β_t stays near its initial value ($\beta_t \geq 0.046$). Per-agent rollout and critic buffers are cleared after each update. The prompt cap of Table 7 truncates only the RL actor's own encoding passes, used to score candidate action strings and to extract the value-head input; the prompts of Section H reach the VLM untruncated.

E.3 Social-plasticity hyperparameters

Table 8 lists the hyperparameters of the social-plasticity module (Section 3.1). Unlike the RL settings, none is learned: the graph W carries no gradient, and every value is a fixed constant of the local update rule. The reward scale $r_{\max}$ is not tuned but is a running maximum of observed $|r_i^{\text{bond}}|$. The trace decay is written ρ_e to keep it distinct from the experience-sharing mixture ρ of Eq. (10), and the co-activity threshold ϵ_c to keep it distinct from the PPO clip ϵ.

Per-experiment settings. The graph couplings apply only to RLFT conditions: reward diffusion shapes the PPO reward, and experience sharing ($\rho = 0.3$; neighbours with $W_{ij} < 0.05$ contribute no samples) requires a policy update for shared transitions to enter, so both are inactive in the zero-shot VLM conditions. The frozen-topology ablation (Table 12) holds W fixed at its imposed values, so the graph is read by the social module but never written. The partner-transfer experiment (Table 3) initialises Phase B either with the learned Phase A graph (within-pair $W = 0.265$, cross-pair 0.10) or with a flat graph of the same mean weight (0.133), instead of the warm start W_0; W remains plastic from the first step. The deliberation interval T_{soc} is 8 steps everywhere except the sweep of Section F.3, which varies it over $\{2, 8, 20, 50, 100, 500\}$.

Table 8: **Social-plasticity hyperparameters.** Symbols match Sections 3.1 and 3.2 and Algorithms 1–2. All values are fixed constants, not learned. The graph couplings apply to RLFT conditions only. *A soft instruction in the social-module prompt (low-bond requests may still be answered if unambiguous and urgent), not a hard filter; the choice-mode prompt of the co-firing arms has no cut-off.

Hyperparameter	Symbol	Value
Co-activity signal (Eqs. 2–5)		
Interaction radius	d	5.0
Engagement reward weight	α	0.5
Social-cue credit (every cue k)	δ_k	0.5
Co-activity floor (co-location)	c_0	0.25
Co-activity threshold	ϵ_c	0.05
Reward scale	$r_{\max}$	running max
Bond update (Eqs. 6–8)		
Initial bond (warm start)	W_0	0.1
Trace decay	ρ_e	0.9
Association-floor growth	η_0	0.001
Reward-modulated growth rate	η_+	0.05
Salience reward normaliser	R	50
Signed failure decay rate	η_-^{d}	0.05
Failure-signal cap	r_{cap}	10
Homeostatic decay rate	λ	0.001
Social module (Algorithm 2)		
Deliberation interval	T_{soc}	8 steps
Bond-change window	n_Δ	50 steps
Trend threshold (rising / decaying)	$\lvert\Delta W_{ij}\rvert$	> 0.02
Low-bond cut-off for help offers*	$w_{\min}$	0.2
Graph couplings (Eqs. 9, 10), RLFT only		
Diffusion strength	γ_d	0.2
Experience-sharing mixture	ρ	0.3
Experience-sharing bond threshold	–	0.05

F ADDITIONAL RESULTS

Curriculum coverage through checkpoints. Table 9 also shows how teams move through the curriculum. Chamber checkpoints place every team in each chamber at a fixed fraction of the horizon, so all conditions face the same sequence of coordination demands regardless of how quickly they progress, and the comparisons in Table 1 are made on equal exposure to Chambers 1–4. Teams reach Chambers 3–5 mostly through these checkpoints, which grant no milestone reward, so the entry milestones (M12, M16, M20) are rarely credited, and episodes end near step 800 in the Chamber 5 boss encounter. The boss milestones (M21–M24) thus remain open for agents with stronger backbones or longer horizons, giving WIRE room to measure cumulative gear and relational state beyond the current results.

F.1 PARTNER TRANSFER (RQ3)

Protocol. In Phase A, nine pairs of Gemma-E4B +Social plasticity agents each run independently through Chambers 1–2 for three episodes. We select the three pairs that complete the cooperative Chamber 2 milestone; these pairs form their bonds through joint action and messaging. Phase B places the six agents in a single team that starts in Chamber 3, so every agent has one former partner and four teammates it has never met. Before the first Phase B episode we set two factors. *Memory* imports each agent's Phase A cognitive state into the freshly built agent: its skill library, its episodic-memory documents, and its curriculum lists. Beliefs and the current task are not transplanted, since they are per-episode working memory. With memory *kept*, each agent receives its own Phase A state and is seated next to its Phase A partner; with memory *empty*, it starts Phase B with empty skill and episode databases, which then fill as usual. *Bonds* set W at step 0: either the block matrix merged from the three Phase A runs, normalised to 0.265 within each pair and 0.10 across pairs (*kept*), or every off-diagonal entry set to 0.133, the mean of the merged matrix (*flat*), which keeps the total bond mass but removes all structure. All other settings follow Table 8.

Table 9: **Steps to first completion of every milestone.** Each cell is the median environment step at which the milestone was first completed, over the episodes that completed it; the subscript is the *percentage* of that condition's episodes in which it was completed, so columns pooling different numbers of seeds remain comparable. "–" means never completed. Lower step and higher percentage are better; n in each column head is that condition's episode count. Conditions and their order match Table 1; RL arms are reported as base and +Social plasticity only. Chamber-entry milestones (M12, M16, M20) are credited only when the team advances on its own before the chamber checkpoint, never for rescue teleports; M13 and M14 are identical by construction, since a cell door opens on the partner's press.

ID	Milestone	Qwen3.5-2B (n=18)	Qwen3.5-2B +Social plasticity (n=18)	Qwen3.5-9B (n=18)	Qwen3.5-9B +Social plasticity (n=18)	Gemma-E4B (n=18)	Gemma-E4B +Central orch. (n=18)	Gemma-E4B +Social plasticity (n=18)	Qwen3.5-2B IPPO (n=12)	Qwen3.5-2B IPPO +Social plasticity (n=12)	Qwen3.5-2B MAPPO (n=18)	Qwen3.5-2B MAPPO +Social plasticity (n=12)	Gemma-E4B IPPO (n=18)	Gemma-E4B IPPO +Social plasticity (n=6)	Gemma-E4B MAPPO (n=18)	Gemma-E4B MAPPO +Social plasticity (n=6)
Chamber 1: solo skill acquisition																
M1	Move >5 from spawn	$10_{100\%}$	$3_{94\%}$	$6_{100\%}$	$4_{94\%}$	$1_{100\%}$	$2_{100\%}$	$4_{100\%}$	$10_{100\%}$	$6_{100\%}$	$6_{100\%}$	$2_{100\%}$	$2_{94\%}$	$4_{100\%}$	$6_{94\%}$	$2_{100\%}$
M2	Dig 3 any	$14_{100\%}$	$16_{94\%}$	$8_{100\%}$	$7_{100\%}$	$45_{72\%}$	$24_{94\%}$	$35_{94\%}$	$18_{100\%}$	$39_{92\%}$	$38_{100\%}$	$20_{83\%}$	$37_{72\%}$	$38_{83\%}$	$19_{72\%}$	$13_{83\%}$
M3	Pick up 3 items	$25_{94\%}$	$22_{78\%}$	$36_{94\%}$	$20_{94\%}$	$86_{22\%}$	$68_{50\%}$	$42_{33\%}$	$107_{42\%}$	$63_{75\%}$	$72_{67\%}$	$100_{75\%}$	$48_{22\%}$	–	$16_{17\%}$	$23_{33\%}$
M4	Dig 5 wood	$115_{6\%}$	$93_{17\%}$	$143_{44\%}$	$151_{33\%}$	–	–	–	–	–	–	–	–	–	–	$88_{17\%}$
M5	Kill 1 animal	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
M6	Kill 2 animals	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
M7	Dig 3 stone	$84_{17\%}$	$111_{17\%}$	$35_{39\%}$	$158_{39\%}$	–	–	–	–	–	–	–	–	–	–	–
M_door1	Door 1 unlocked	$13_{94\%}$	$14_{89\%}$	$8_{94\%}$	$7_{94\%}$	$45_{72\%}$	$24_{94\%}$	$35_{94\%}$	$18_{100\%}$	$39_{92\%}$	$37_{94\%}$	$20_{83\%}$	$37_{72\%}$	$38_{83\%}$	$19_{72\%}$	$13_{83\%}$
Chamber 2: cooperative resource acquisition																
M8	First anvil break	–	–	$201_{11\%}$	$268_{11\%}$	–	–	–	–	–	–	–	–	–	–	–
M9	Second anvil break	–	–	$218_{11\%}$	–	$199_{6\%}$	–	–	–	–	–	–	–	–	–	–
M10	Sword equipped	–	–	$201_{11\%}$	$268_{11\%}$	–	–	–	–	–	–	–	–	–	–	–
M11	Chestplate equipped	–	–	$218_{11\%}$	–	$199_{6\%}$	–	–	–	–	–	–	–	–	–	–
Chamber 3: communication under partial observability																
M12	Enter isolation cell	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
M13	Press switch	$434_{78\%}$	$439_{72\%}$	$415_{100\%}$	$417_{94\%}$	$406_{39\%}$	$407_{61\%}$	$434_{61\%}$	$488_{58\%}$	$490_{67\%}$	$486_{39\%}$	$500_{58\%}$	$451_{11\%}$	$490_{67\%}$	$506_{39\%}$	$524_{17\%}$
M14	Door opened	$434_{78\%}$	$439_{72\%}$	$415_{100\%}$	$417_{94\%}$	$406_{39\%}$	$407_{61\%}$	$434_{61\%}$	$488_{58\%}$	$490_{67\%}$	$486_{39\%}$	$500_{58\%}$	$451_{11\%}$	$490_{67\%}$	$506_{39\%}$	$524_{17\%}$
M15	All in communal room	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Chamber 4: team combat																
M16	Enter Chamber 4	–	–	–	–	–	–	–	–	–	–	–	–	$946_{17\%}$	–	–
M17	First mob kill	–	–	$679_{56\%}$	$674_{39\%}$	$670_{56\%}$	$692_{67\%}$	$695_{94\%}$	–	$626_{8\%}$	–	$663_{8\%}$	$712_{11\%}$	$693_{17\%}$	$705_{17\%}$	–
M18	All mobs killed	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
M19	All survive (damage)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Chamber 5: cooperative boss fight																
M20	Enter Chamber 5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
M21	First boss damage	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
M22	Boss at 50% HP	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
M23	Boss defeated	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
M24	All alive at kill	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Communication track																
M_comm_ch1	Valid messaging, Ch1	$5_{100\%}$	$5_{100\%}$	$5_{100\%}$	$5_{100\%}$	$5_{100\%}$	$5_{100\%}$	$5_{100\%}$	$5_{100\%}$	$5_{100\%}$	$5_{100\%}$	$6_{100\%}$	$5_{100\%}$	$6_{100\%}$	$5_{100\%}$	$5_{100\%}$
M_comm_ch2	Valid messaging, Ch2	$204_{100\%}$	$204_{100\%}$	$204_{100\%}$	$204_{100\%}$	$138_{100\%}$	$128_{100\%}$	$126_{100\%}$	$143_{100\%}$	$133_{100\%}$	$178_{100\%}$	$142_{100\%}$	$108_{100\%}$	$80_{100\%}$	$98_{100\%}$	$67_{100\%}$
M_comm_ch3	Valid messaging, Ch3	$604_{100\%}$	$604_{100\%}$	$502_{100\%}$	$536_{100\%}$	$604_{100\%}$	$604_{100\%}$	$604_{100\%}$	$603_{100\%}$	$604_{100\%}$	$604_{100\%}$	$604_{100\%}$	$608_{100\%}$	$604_{100\%}$	$607_{100\%}$	$610_{100\%}$
M_comm_ch4	Valid messaging, Ch4	$612_{100\%}$	$610_{100\%}$	$613_{100\%}$	$610_{100\%}$	$612_{100\%}$	$608_{100\%}$	$610_{100\%}$	$610_{100\%}$	$613_{100\%}$	$610_{100\%}$	$612_{100\%}$	$616_{100\%}$	$650_{100\%}$	$614_{100\%}$	$616_{100\%}$
M_comm_ch5	Valid messaging, Ch5	$850_{6\%}$	$810_{50\%}$	$805_{22\%}$	$816_{33\%}$	$809_{56\%}$	$813_{39\%}$	$813_{61\%}$	$805_{8\%}$	$816_{75\%}$	$807_{28\%}$	$808_{42\%}$	$820_{39\%}$	$412_{33\%}$	$817_{33\%}$	–

Table 10: **Learned bonds primarily capture communication structure.** Spearman ρ between end-of-episode bond W_{ij} and per-pair counts of messages, joint digs, and proximity steps, over pair-episode records (6 ordered pairs $\times$ 3 episodes $\times$ 6 seeds $=$ 108 per condition). Each cell gives the pooled ρ with the per-seed mean $\pm$ SD in parentheses. All rows use +Social plasticity. **Finding:** bonds track who communicates most closely ($\rho = +0.58$ to $+0.70$ for every backbone), consistent with RQ2; their association with joint digging and proximity is positive on average and varies more across seeds. Because communication also contributes directly to bond growth, ρ(messages) partly reflects this coupling, whereas ρ(joint digging) and ρ(proximity) are independent behavioural signals.

Condition	n	Messages	Joint digging	Proximity
Qwen3.5-2B	108	$+0.70$ ($+0.68 \pm 0.12$)	$+0.24$ ($+0.16 \pm 0.29$)	$+0.36$ ($+0.34 \pm 0.41$)
Qwen3.5-9B	108	$+0.58$ ($+0.53 \pm 0.19$)	$+0.33$ ($+0.32 \pm 0.26$)	$+0.36$ ($+0.29 \pm 0.30$)
Gemma-E4B	108	$+0.60$ ($+0.63 \pm 0.08$)	$+0.09$ ($+0.13 \pm 0.45$)	$+0.05$ ($+0.13 \pm 0.42$)

Nothing is frozen. Both factors are initial conditions. In every condition, new skills and episodes are written throughout Phase B, and W is plastic from the first step under the three-factor rule of Section 3.1. Between episodes, the reset wipes only working memory (task, chamber, beliefs, and critic cache) and keeps the skill and episode databases and the bonds, so episode 2 starts where episode 1 ended. "No memory" thus means empty at step 0 of Phase B and regrown from there, not disabled, and "flat bonds" means undifferentiated at step 0 and then free to differentiate. The experiment therefore asks whether the starting state still matters after three episodes of unrestricted learning, not whether memory or bonds act as a fixed prior; a frozen-bond variant is supported by the code but was not run.

Table 11: **RQ3: partner transfer, per-pair breakdown** (companion to Table 3; Gemma-E4B, +Social plasticity; mean ± SD over 3 pooled episodes). Conditions as in Table 3. *Partner pref.*: fraction of a member's messages addressed to its partner, averaged over the pair's two members (chance $1/5 = 0.20$). *Msgs within*: messages exchanged between the two members per episode. *Bond* W: end-of-episode within-pair weight, both directions averaged (initialised at 0.265 with kept bonds and 0.133 with flat bonds). Cross-pair W at episode end, pooled per condition: (a) 0.195 ± 0.140; (b) 0.209 ± 0.140; (c) 0.190 ± 0.146; (d) 0.201 ± 0.143. Bold: best; underline: second best.

Condition	Pair	Partner pref.	Msgs within	Bond W
(a) Full transfer	**(a-1)** Pair 1	0.44 ± 0.16	601 ± 217	$\underline{0.432} \pm 0.027$
	(a-2) Pair 2	0.37 ± 0.27	500 ± 360	0.392 ± 0.054
	(a-3) Pair 3	$\mathbf{0.85} \pm 0.09$	$\mathbf{1158} \pm 128$	$\mathbf{0.479} \pm 0.022$
(b) Memory only	**(b-1)** Pair 1	0.23 ± 0.29	314 ± 399	0.314 ± 0.028
	(b-2) Pair 2	0.22 ± 0.29	302 ± 388	0.296 ± 0.054
	(b-3) Pair 3	0.08 ± 0.10	108 ± 137	0.305 ± 0.043
(c) Bonds only	**(c-1)** Pair 1	0.07 ± 0.02	91 ± 26	0.273 ± 0.008
	(c-2) Pair 2	0.01 ± 0.01	12 ± 14	0.238 ± 0.042
	(c-3) Pair 3	0.35 ± 0.43	479 ± 593	0.374 ± 0.077
(d) Fresh start	**(d-1)** Pair 1	0.40 ± 0.26	546 ± 367	0.362 ± 0.036
	(d-2) Pair 2	$\underline{0.49} \pm 0.36$	$\underline{676} \pm 511$	0.416 ± 0.059
	(d-3) Pair 3	0.12 ± 0.17	175 ± 245	0.309 ± 0.074

Table 12: **RQ3: imposed topology across horizons** (Qwen3.5-9B; fixed *No-bonds*, *Allied-pair*, and *Allied-all* graphs at 1,000- and 2,000-step horizons on the same two seeds; n=6 episodes per cell, mean ± population SD over pooled episodes). Solo and Coop. % are the agent-level completion rates of Table 1; *All* covers the 25 non-communication milestones ($24N+1$ achievable). *Task return*: team-summed environment reward including communication pay, as in Table 1. **Finding:** no imposed topology consistently dominates: *No-bonds* is best at 1,000 steps and *Allied-all* at 2,000 steps on Coop. and All, and the differences stay within the observed variability.

		Milestone %			
Condition	Steps	Solo ↑	Coop. ↑	All ↑	Task return ↑
No-bonds	1,000	37.9 ± 6.8	9.5 ± 4.3	18.0 ± 3.8	571 ± 98
	2,000	46.2 ± 8.9	12.7 ± 4.8	22.8 ± 5.5	821 ± 230
Allied-pair	1,000	39.4 ± 7.3	6.5 ± 3.1	16.4 ± 3.4	532 ± 91
	2,000	38.6 ± 6.8	10.8 ± 3.3	19.2 ± 2.6	644 ± 124
Allied-all	1,000	38.6 ± 7.3	6.9 ± 2.7	16.4 ± 2.6	539 ± 84
	2,000	44.7 ± 6.1	13.7 ± 5.1	23.1 ± 2.7	796 ± 222

F.2 Episode-level coordination statistics

The orchestrator allocates *tasks*: pairings are a by-product of assignments and are recomposed when the plan changes. Paired work accounts for 16% of assigned agent-steps, compared with 51% individual and 33% full-team work; a further 16% of agent-steps have no assignment. The plasticity graph instead carries the *relationship*: bonds update continuously across chamber boundaries. In the representative run shown in Figure 15, a_0–a_1 remains the strongest pair for 94% of the run and each credited joint milestone is completed by the currently top-ranked pair. These statistics are descriptive rather than causal.

F.3 Deliberation frequency and compute

The graph is updated at every environment step but read by the social module only every T_{soc} steps; between reads, the most recent social directive is cached. We sweep $T_{\text{soc}} \in \{2, 8, 20, 50, 100, 500\}$ for +Social plasticity on Gemma-E4B against the matched baseline without a social module, which spends no social-module compute. Compute is reported as inference FLOPs per episode, estimated as $2N_{\text{eff}}(n_{\text{prompt}} + n_{\text{decode}})$ and summed over all social-module calls. Milestone completion here is

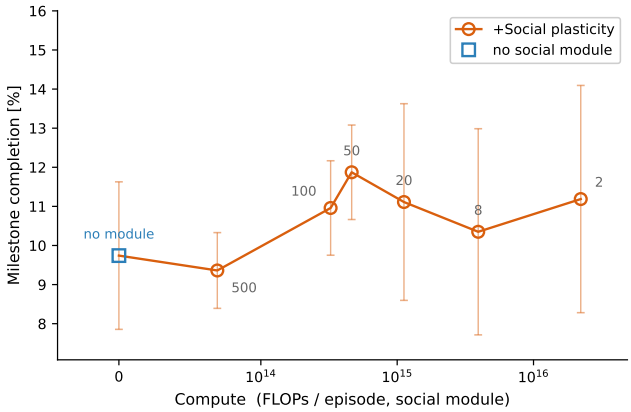
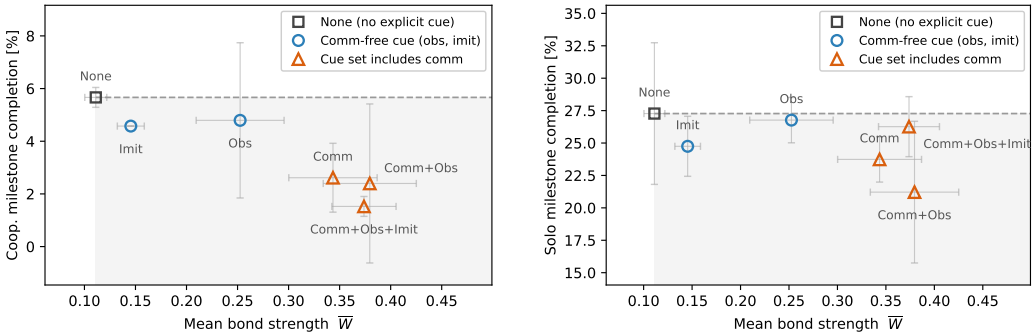


Figure 7: **Deliberation interval against social-module compute** (+Social plasticity, Gemma-E4B; bars are ±1 SD, numbers give T_{soc} in steps). Without a social module (zero social compute), agents complete 9.7% of milestones. Every interval from $T_{\text{soc}} = 2$ to 100 lies above this, between 10.3% and 11.9%, with the best result at $T_{\text{soc}} = 50$; $T_{\text{soc}} = 100$ (11.0%) still exceeds the default $T_{\text{soc}} = 8$ (10.3%) at roughly 12× less compute. Only at $T_{\text{soc}} = 500$ does completion fall back to the no-module level (9.3%), so sparse deliberation retains the benefit at a fraction of the compute.



(a) Cooperative (Chambers 2–5) **(b) Solo (Chamber 1)**

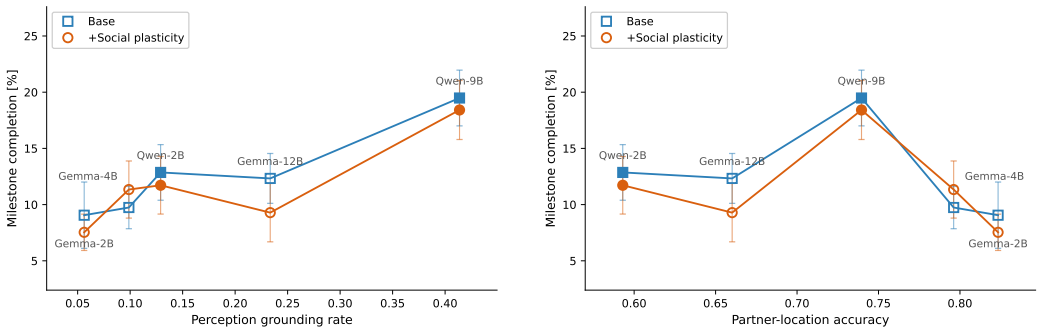
Figure 8: **RQ2: bond strength does not buy cooperation.** Agent-level completion against the mean bond strength $\overline{W}$ each cue condition produces (Gemma-E4B, +Social plasticity; bars are ±1 SD across seeds). "None" in the plots is the *Spatial engagement* condition of Table 2. The dashed line marks its completion; the shaded region holds every condition with stronger bonds and lower completion. In both panels *Spatial engagement* is the only non-dominated condition: every explicit cue adds bond mass without adding completion, and the three cue sets that include communication reach the largest $\overline{W}$ and the lowest cooperative completion.

the agent-level rate over all 25 non-communication milestones ($24N+1$ achievable), combining the Solo and Coop. columns of Table 1.

Sparse social deliberation is enough. Performance depends little on how often the graph is read (Figure 7), while social-module compute falls by more than two orders of magnitude from $T_{\text{soc}} = 2$ to 500. Reading the graph every 50–100 steps is as effective as reading it every few steps, which suggests that the relational structure changes slowly enough to be consulted rarely. Consulting it too rarely ($T_{\text{soc}} = 500$, about once per chamber) removes the benefit. The differences are within the observed variability, so the sweep shows that sparse deliberation does not hurt, rather than that a particular interval is optimal.

F.4 Bond strength against completion (RQ2)

Figure 8 plots the RQ2 conditions as a trade-off between bond strength and completion. Cooperative completion falls as bonds grow stronger (panel a): the conditions that include communication build



(a) Perception grounding **(b) Partner-location accuracy**

Figure 9: **Zero-shot performance against backbone perception measures.** Agent-level milestone completion (all 25 non-communication milestones; bars are ± 1 SD) without (Base) and with social plasticity across five backbones. Perception grounding measures whether free-text observations identify objects consistent with the current chamber; partner-location accuracy measures whether claims about teammates match their true chamber. Gemma-2B and Gemma-4B denote Gemma-E2B and Gemma-E4B.

the strongest bonds and complete the fewest cooperative milestones, while observation and imitation keep bonds weaker and completion closer to the *Spatial engagement* condition. Solo completion (panel b) varies less, which indicates that the cost of communication falls mainly on cooperation rather than on individual skill.

F.5 Backbone capability analysis

Figure 9 asks whether the effect of social plasticity at inference time depends on the backbone's perceptual ability. Stronger perception raises performance overall: Qwen3.5-9B has both the best grounding and the highest completion (panel a). The benefit of the coupling is concentrated on Gemma-E4B (11.3% against 9.7%), a backbone that localises its teammates well (panel b), while the other backbones remain close to or below their baseline. Knowing *who* is strongly bonded is most useful when an agent can also find *where* that partner is, and when the agent does not already coordinate on its own. This pattern matches the main result that relational state is most useful when it shapes learning (Section 5.1), where it improves cooperation in three of four configurations.

G Coordination timelines

This appendix complements the aggregate results with full-run timelines that align message traffic with each mechanism's internal coordination state: bond trajectories under social plasticity and task assignments under centralised orchestration. All runs use Gemma-E4B; each caption names the condition and seed. The timelines are descriptive rather than causal, but they show where in a run coordination forms and what state each mechanism carries forward.

G.1 Relational structure under social plasticity

Figure 10 shows how the graph evolves over three episodes. Two features stand out. First, consolidation is event-driven: bonds grow in steps that coincide with milestone rewards rather than uniformly with every co-active step, which is the behaviour the eligibility trace is designed to produce. Second, the ordering of pairs is stable: the same top pair persists across episodes even though communication remains distributed across the whole team. The graph therefore does not simply mirror message volume; it retains which interactions were followed by progress. What it cannot do is make that progress happen. In the expanded windows, agents coordinate intent explicitly but still depend on their own grounding to execute the joint act, consistent with the aggregate finding that bonds track communication more closely than physical cooperation (Table 10).

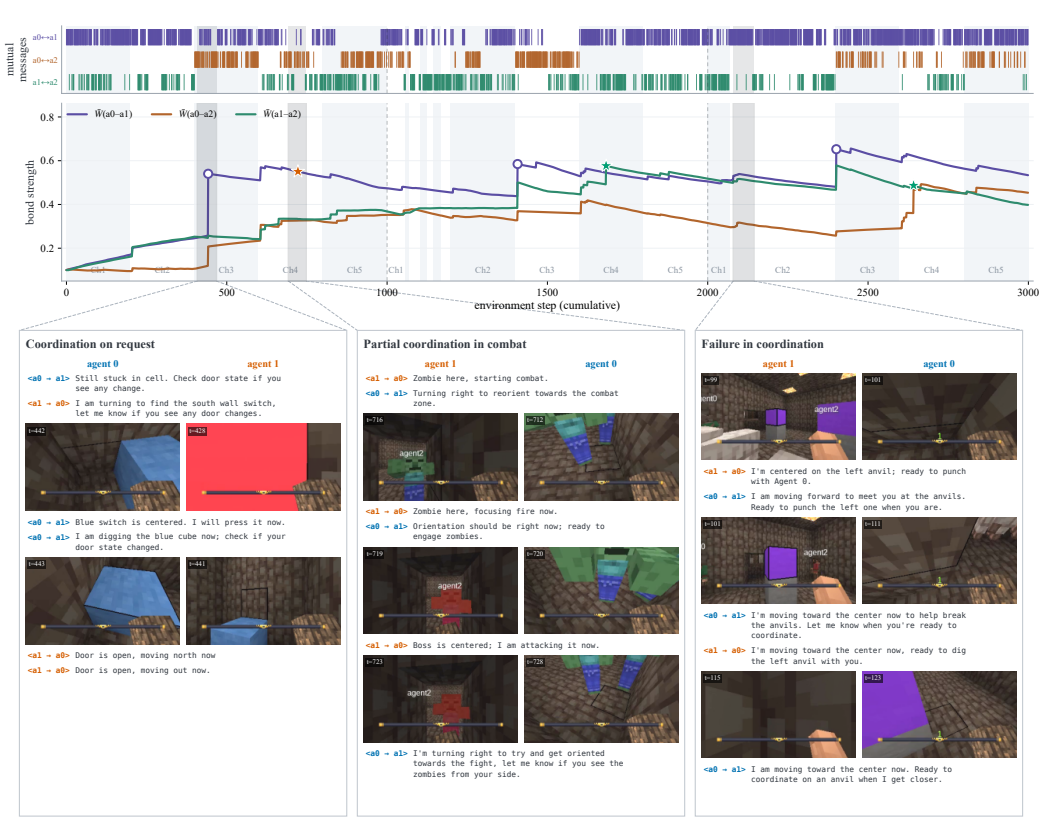


Figure 10: **Qualitative coordination dynamics for Gemma-E4B +Social plasticity.** Mutual pairwise messages and bond strengths over three episodes, with representative interaction windows expanded below. Under the trace rule, consolidation events coincide with milestone rewards rather than with the co-active steps themselves, and the resulting bond ordering is stable across episodes: the same top pair persists while communication remains distributed across the team.

G.2 Orchestration versus plasticity

Six agents. Figures 11 and 12 extend the Chamber 4 comparison of Figure 4 to full three-episode runs. The orchestrator structures the team top-down: in Chamber 4 it splits the fight into six solo kill tasks, one agent kills alone and another never joins, and the run's only jointly credited milestone comes from a pair the plan never assigned together. Under plasticity there is no plan to follow, and structure instead emerges from the graph: the leading pair changes between episodes, and joint combat forms around whichever bonds are currently strong, including a three-agent fight in episode 2 and a hand-over of a target between two agents in episode 3. In short, orchestration determines *what* each agent works on, while the graph biases *with whom* it ends up working.

Three agents. Figures 13 and 14 show the same contrast in a three-agent team over the whole chamber sequence. Under orchestration, the coordination state is the assignment pattern: agents alternate between independent, paired, and full-team subtasks as the coordinator issues, expires, and reissues slots after failures. Under plasticity, the state is the bond trajectories $\overline{W}$: pairwise bonds differentiate and persist across chamber boundaries, and joint milestones and mob kills coincide with bond growth. Figure 15 condenses the two runs to their coordination state alone, which makes the key difference visible: the orchestrator's structure is recomposed at every replanning step, whereas the graph's structure accumulates within the episode and carries across episodes. Episode-level statistics for these runs are reported in Section F.2.

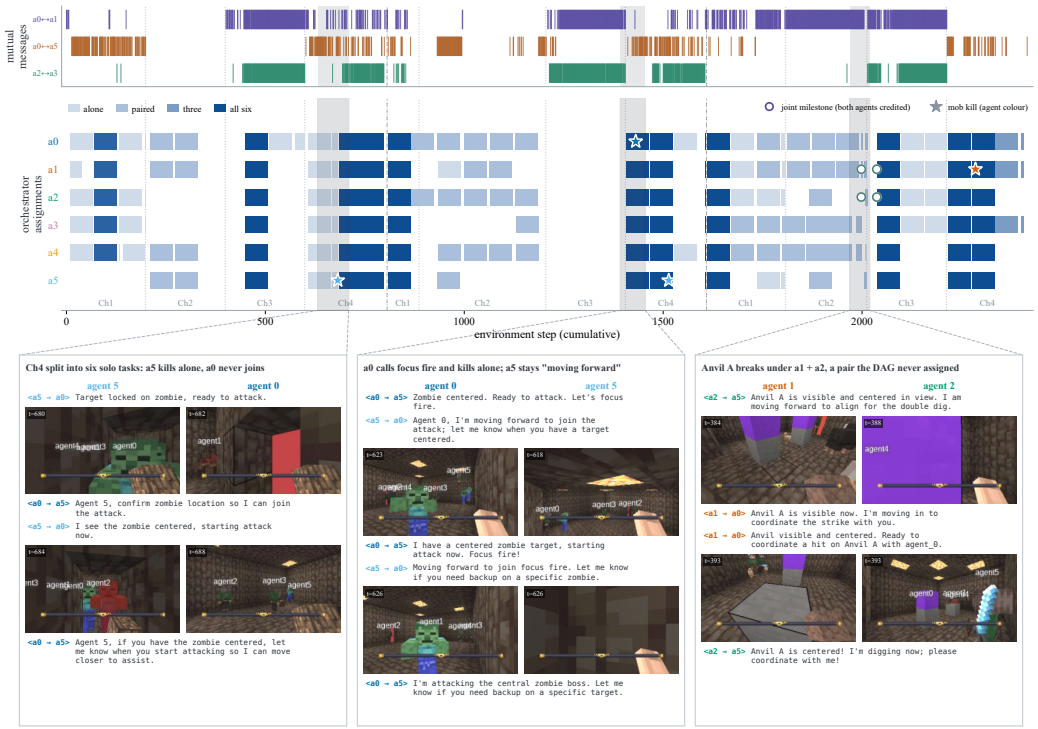


Figure 11: **Full timeline under centralised orchestration at six agents** (Gemma-E4B, seed 42, three episodes). Top: mutual message activity for the three most active pairs. Middle: per-agent orchestrator assignments, shaded by team size (alone to all six); circles mark joint milestones, stars mob kills, shaded bands the vignette intervals. Bottom: message excerpts with first-person frames. The vignettes show the Chamber-4 split into six solo kill tasks (a5 kills alone, a0 never joins; the interval of Figure 4b), a focus-fire call that nobody reaches (ep. 2), and the run's only jointly credited milestone, anvil A broken by a1 and a2, a pair the plan never assigned together.

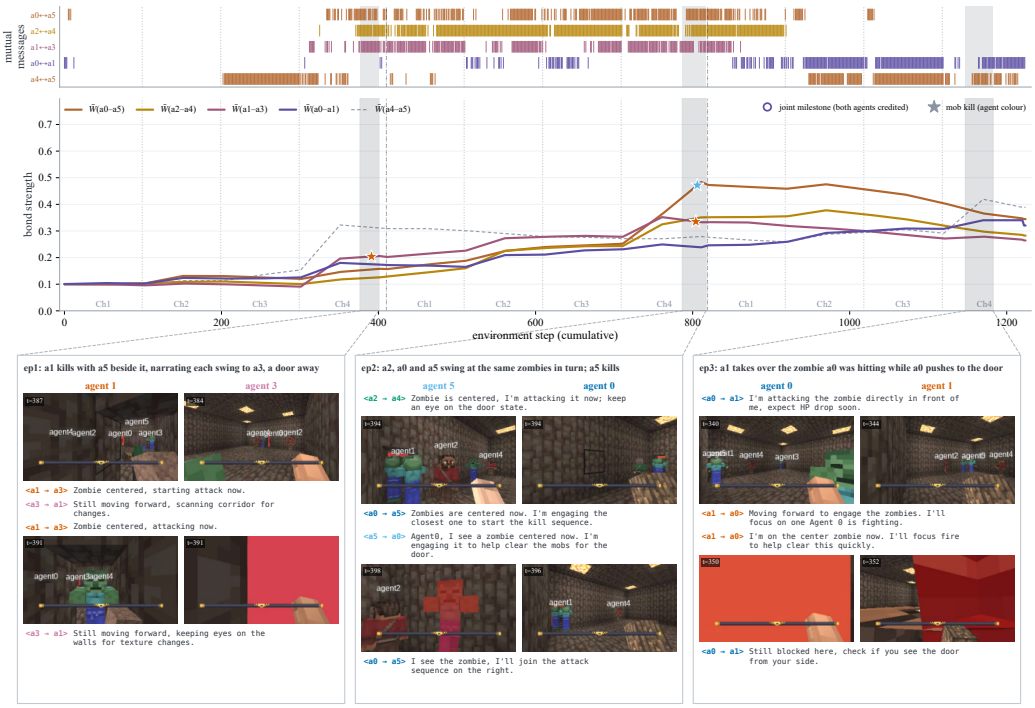


Figure 12: **Full timeline under relational plasticity at six agents** (Gemma-E4B, +Social plasticity, seed 456, three episodes). Top: mutual message activity for the five pairs drawn below. Middle: bond strengths $\overline{W}$ for the pairs that lead at some point in the run (a0–a5, a2–a4, a1–a3 in ep. 2; a0–a1 in ep. 3) and, dashed grey, a4–a5, the leader in eps. 1 and 3; stars mark mob kills, shaded bands the vignette intervals. No anvil or switch fires in this run. Bottom: message excerpts with first-person frames. The vignettes show a1's kill narrated swing by swing to a3, a door away, with a5 beside it (ep. 1); the three-agent fight of Figure 4a (ep. 2); and a1 taking over the zombie a0 was hitting as a0 pushes on to the door (ep. 3).



Figure 13: **Full-episode timeline under centralised orchestration** (N=3, Gemma-E4B). Top: mutual pairwise message activity. Middle: per-agent orchestrator assignments (independent, paired, full team); circles mark joint milestones, stars mob kills, and shaded bands the intervals of the vignettes below. Bottom: message excerpts with first-person frames showing coordinator-directed combat, an assigned joint subtask, and task reassignment after failure.

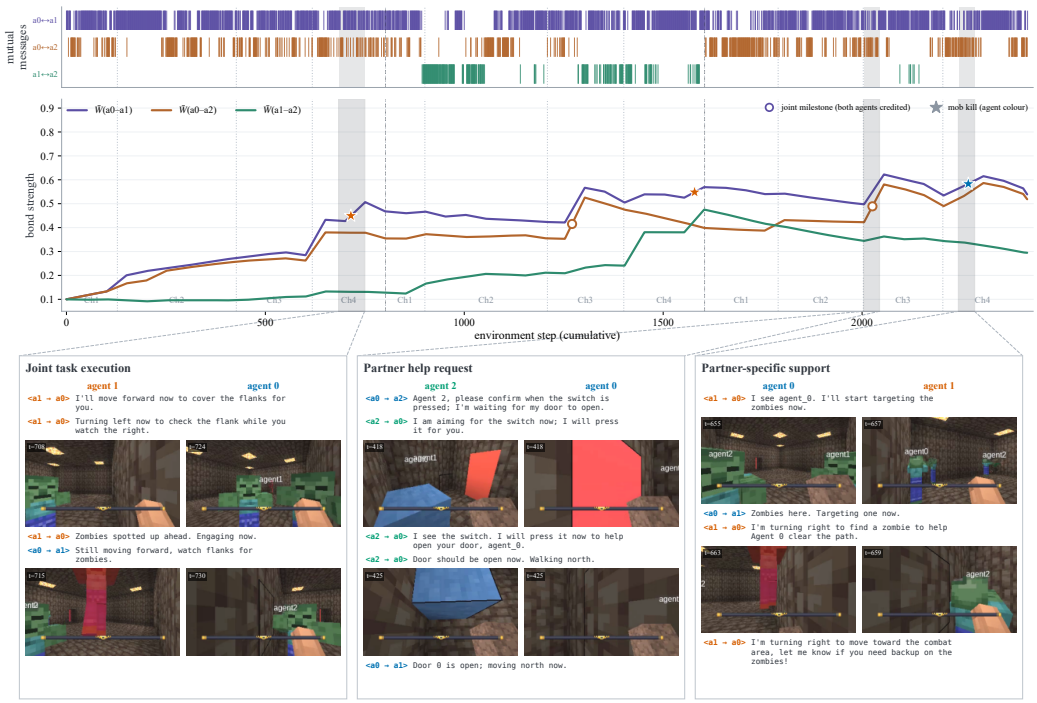


Figure 14: **Full-episode timeline under relational plasticity** (N=3, Gemma-E4B, +Social plasticity). Top: mutual pairwise message activity. Middle: pairwise bond strengths $\overline{W}$; circles mark joint milestones, stars mob kills, and shaded bands the vignette intervals. Bottom: message excerpts with first-person frames showing joint task execution, a partner help request, and partner-specific support.

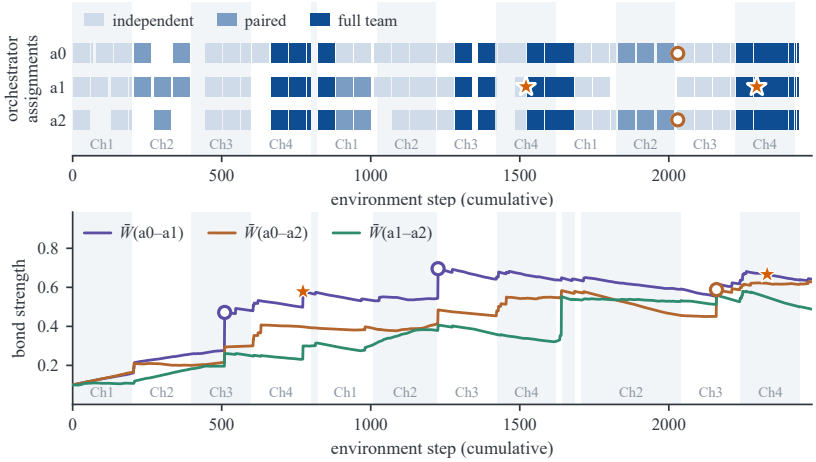


Figure 15: **Centralised orchestration and relational plasticity retain different coordination state.** Top: orchestrator assignments are discrete and are recomposed as the task plan changes. Bottom: social plasticity updates pairwise bonds continuously across the same sequence of WIRE chambers. Circles denote joint milestones and stars mob kills. The panels show representative Gemma-E4B runs; aggregate performance is reported in Table 1.

H PROMPT TEMPLATES

This appendix reproduces the prompt templates that drive agent behaviour. Fields in single curly braces (e.g. {task}) are placeholders filled at runtime from the cognitive modules of Appendix B; JSON braces inside response-format examples are shown as the model receives them. We list the prompts central to our claims: the static system prompt (with the environment description substituted into it), the per-step action-selection prompt that defines the JSON interface, the social-module prompt that conditions the agent on its learned bond row, the choice-mode variant that adds the social-act fields, the texts through which observation and imitation are delivered, the thoughts-only variant used by RL agents, and the three prompts of the centralised orchestrator baseline. The remaining templates (auto-curriculum, critic, belief updates, episode summarisation) follow the same conventions and are available in the released code. Where a prompt is a small variant of one already shown, we give the lines that differ rather than repeating it in full.

Prompt assembly. Each LLM call consists of a *system message* and a *user message*. The system message is static across the run: the template in Prompt H.1 with the environment description of Prompt H.2 substituted for {environment_prompt}. The user message is rebuilt every step from Prompt H.3: the current task, reward and status lines, chamber facts and live chamber state, milestone progress, the four belief blocks, retrieved skills and episode summary, and, when the social module is active, its rendered directive in place of the raw {social_directive} line; the agent's first-person frame is attached as an image. In RL mode the same context is sent through a *thoughts-only* variant of Prompt H.3 that omits the action field and instructs the model that a separate policy will choose the action conditioned on its emitted thoughts (subsection 3.3); the constrained-generation actor then scores the candidate action strings as continuations of the prompt plus those thoughts.

H.1 SYSTEM PROMPT

The static system message is shown in Prompt H.1; its {environment_prompt} placeholder is replaced by Prompt H.2.

System prompt template

You are an autonomous agent in a Minecraft–like 3D open world (Craftium/Luanti). Each step you observe the world through an image and structured context, then select ONE action.

{environment_prompt}

THE IMAGE IS YOUR EYES:
– The image shows exactly what you are currently facing. Its CENTER is your crosshair –– the ONLY point where Dig and Place take effect. If a tree, block, or mob is not at the center, Dig hits nothing.

HUD OVERLAYS –– these are permanent, ignore them:
– A row of hearts and food icons may sit along the bottom edge. They are PERMANENT overlays that no action can remove or "close". There is NO inventory screen and NO menu to open/close –– LookDown/NoOp/anything does NOT close any UI.
– NEVER call the HUD "blocking your view" or an "inventory screen". The world is ALWAYS visible above and around the icons. Your health is given as text in Status –– read it there, not from the image.

HOW TO AIM:
– Target LEFT of center –> TurnLeft. Target RIGHT –> TurnRight.
– Target too HIGH –> LookUp. Too LOW –> LookDown. Far away –> MoveForward to close distance.
– When the target fills the center, you are aimed and can Dig or Place.

THE BASIC ACTION LOOP –– follow this every time:
1. LOOK –– what is visible? Identify the nearest useful target (tree, stone, animal, mob).
2. ORIENT –– if it is not centered, turn toward it (TurnLeft/TurnRight).

System prompt template (continued)

3. APPROACH −− if it is far (small in the image), MoveForward.
4. ALIGN −− adjust vertical aim (LookUp/LookDown) until centered.
5. ACT −− Dig (break a block / attack a mob) or Place. Dig must be repeated to break a block.

WHEN STUCK OR EXPLORING:
– If MoveForward leaves your position unchanged, a wall or step is blocking you −− choose Jump (then resume MoveForward), or back up with MoveBackward and turn if still stuck. A block/slope at foot level in the image is the cue to Jump. Don't spam Jump on flat ground.
– With no visible target, alternate MoveForward with TurnRight/TurnLeft to explore and sweep the area for teammates/resources; turn to face a visible target, then MoveForward to close on it.

COMMON MISTAKES TO AVOID:
– Do NOT Dig unless a block or mob is visible and centered −− otherwise you waste the action.
– Do NOT keep moving forward when already close −− stop and aim first.
– Do NOT choose NoOp when anything is explorable −− always prefer a directional action.
– Do NOT LookUp more than 1–2 times in a row −− the camera overshoots into the sky; if you see mostly sky, LookDown immediately. Tree trunks are at eye level −− prefer LookDown for ground blocks and roots, you rarely need LookUp to dig.
– After breaking a block, ALWAYS MoveForward over the drop to collect it −− digging alone adds nothing to inventory.

YOUR COGNITIVE MODULES (refreshed each step, supplied in the per−step context):
– A "Beliefs" block with: "See now" (what you perceive, where targets are vs. center), "Task" (your current task), "Teammates" (what teammates are doing), "Interactions" (useful info from teammate messages).
– "Past episodes" (Episodic memory −− past successes/failures) and "Relevant skills" (Skill memory −− action sequences that worked).

THINKING IN SCHEMES (the per−step discipline −− the world teaches you through (action, observation) pairs; SURPRISE, when an observation differs from your prediction, is your strongest learning signal):
1) PREDICT −− state the concrete, falsifiable observation you expect from this action ("if I Dig this trunk I should see a log drop at my feet").
2) COMPARE −− check last step's prediction vs. what actually happened. A match grounds the scheme; a mismatch is SURPRISE −− prefer follow−ups that re−test it rather than abandoning.
3) WEIGH −− EXTRINSIC: advance the current task / open milestone. INTRINSIC: among similar options, prefer a scheme that is NOVEL in this context (a 10x−repeated action yields no new information). Lead with novelty in early exploration or when stuck.
4) TRUST OBSERVATIONS over labels −− if a label and what you see disagree, believe what you see.

TARGETED COMMUNICATION (REQUIRED EVERY STEP −− there is NO broadcast channel):
– Set "communication_target" to EXACTLY ONE teammate's name (form "agent_N", any agent other than yourself). Never "all", never empty, never yourself.
– Pick the recipient deliberately −− whose situation does this message most help? Choose the teammate your action affects or whose action affects yours (e.g. someone near you, or who can act on what you just observed).
– Make it ACTIONABLE for that recipient: what you observed, a scheme you just confirmed, what you need from THEM, or what you commit to do. Share schemes a teammate lacks (e.g. "digging that block dropped wood"). Vary messages −− sub−5−char or repeated identical messages to the same teammate are filtered as spam and never arrive.

Your per−step context and the exact JSON response format are provided in the user message that follows −− respond with that JSON only, no extra text.

H.2 Environment prompt

Prompt H.2 is substituted into the system prompt. Chamber-specific objectives are deliberately excluded here and delivered per step through the {current_chamber} field (ROOM FACTS), so each agent carries only the room it is standing in.

Environment description

You are one of three agents in a five−chamber cooperative dungeon. Your shared goal is to progress through all chambers and defeat the boss in Chamber 5. The objective and mechanics of the room you are CURRENTLY in are given each step in the "Chamber" field (ROOM FACTS) −− this prompt covers only the general mechanics shared across all rooms.

ACTIONS −− choose EXACTLY ONE by name each step: NoOp, MoveForward, MoveBackward, MoveLeft, MoveRight, Jump, Sneak, Dig, Place, Drop, Slot1−Slot8, TurnRight, TurnLeft, LookDown, LookUp. You pick ONE per step and are called again next step, so chain primitives: to reverse, emit several TurnRight; to sweep an area, turn step−by−step and read each frame; to approach a target, centre it with TurnRight/TurnLeft then MoveForward. Acting one step at a time lets you react to what you see and to teammates every step.

BASIC BEHAVIOR:
− Attack a mob: face it and Dig repeatedly. There is NO separate "kill" action.
− Break a block (wood, stone, anvil): face it and Dig.
− Pick up a drop: walk over it with MoveForward (auto−collects within ˜1 block). After every Dig, MoveForward to collect the drop −− digging alone does NOT collect it.
− Use Slot1−Slot8 to choose your wielded item (e.g. Slot1 for the sword before combat).
− Your inventory is shown in the "Inventory" field of the per−step context.

COMBAT TACTICS:
− Mobs MOVE. Between hits the mob has likely shifted −− re−check the image and TurnLeft/TurnRight/LookDown/LookUp to RECENTER it before each Dig. Digging where the mob WAS hits air.
− One Dig is one hit, not a kill. Expect multiple hits: chicken ˜2 (flees fast), sheep ˜4 (flees), zombie ˜3 with a diamond sword (many more bare−handed, and it hits back −− wield the sword first, Slot1 if it auto−equipped on anvil break).
− Mobs flee when injured: alternate MoveForward (chase) + Turn (track), then Dig.
− Drops confirm a kill landed −− chickens drop feather + raw_chicken, sheep drop wool + raw_mutton, zombies drop rotten_flesh. No drop yet = mob still alive, keep attacking.
− Health: below 10/20 retreat from active fights; below 6/20 get out of combat entirely −− don't trade hits at this HP.
− In Ch4/Ch5 a mob takes damage from every agent hitting it, so the more agents attack the same target the faster it dies; an agent fighting a mob alone absorbs all of its return hits.

CHAMBER WALLS ARE UNBREAKABLE:
− Walls, floor, and ceiling enclosing every chamber are INDESTRUCTIBLE −− Dig does nothing, no drop, no progress. Don't spend actions trying to dig through them; doors and teleports are the ONLY way between chambers. Dig only INTERIOR objects (trees, stone, animals, anvils, switches, zombies, boss).

VISUAL CUES −− recognising special blocks:
− BLUE solid cube: a Cell Switch (Chamber 3 only) −− the only blue block in the world. Press by facing it and using Dig.
− RED glowing block: a LOCKED door. Not diggable; it becomes air (an open hole) when its unlock condition is met. The red glow means still locked.
− PURPLE cube (Chamber 2): an anvil top, sitting on a smaller GRAY stone pedestal (a 2−block pillar). Dig the PURPLE top; the gray pedestal is inert. Exactly TWO anvils exist (sword + chestplate). Anvils vanish permanently on break (the pedestal remains).
− Glowstone tiles on the ceiling: light only, nothing to interact with.

SAFETY:
− If you stop moving after several MoveForward attempts, you are physically blocked −− back up with MoveBackward, turn (TurnRight/TurnLeft), then try a new direction.

Environment description (continued)

The specific objective and mechanics of the room you are in right now (which doors open how, what to dig, whether a task is solo or cooperative) arrive each step as ROOM FACTS inside the "Chamber" field −− read that for room−specific guidance.

H.3 PER-STEP ACTION-SELECTION PROMPT

The user message rebuilt at every step is shown in Prompt H.3. It defines the JSON interface: one discrete action, free-text thoughts constrained to spatial reasoning plus a falsifiable prediction, and a mandatory targeted message with a single named recipient, the channel on which both the communication co-activity term and the Chamber 3 puzzle depend. RL agents receive a thoughts-only variant (Section H.7). All fields are filled at runtime; `{social_directive}` carries the social module's rendered decision when the module is active, the raw bond row otherwise, and the coordinator's directive (Prompt H.11) in the orchestration baseline.

Per-step action-selection prompt (user message)

YOU ARE: {agent_name}. Your teammates are: {teammate_names}. "communication_target" must be EXACTLY one of {teammate_names} −− never "{agent_name}", never "all".

Task: {task}
Last action: {last_action}
Reward: {reward_text}
Position: {position_text}
Chamber: {current_chamber} (your CURRENT room −− trust this over the image; bedrock walls and stone blocks appear in every chamber)
Chamber state: {chamber_state} (live puzzle state −− e.g. anvil HP and who is punching it. Trust this over the image.)
Visited chambers so far this episode: {visited_chambers} (you have ONLY been in these −− never claim to be in a chamber not listed)
Milestone progress (authoritative −− advance an [OPEN] milestone in YOUR chamber, marked "(YOU ARE HERE)"; do not redo [you done] ones):
{milestone_progress}
{milestone_event}
Status: {player_status_text}
Critique: {critique}
Error: {error}
Inventory: {picked_object}
Relevant skills: {skill_memory}
Past episodes: {episode_summary}
Your beliefs:
− See now: {perception_beliefs}
− Task: {task_beliefs}
− Teammates: {partner_beliefs}
− Interactions: {interaction_beliefs}
{social_directive}
{propagation_summary}

Look at the image: identify what is at the center, what is left/right, and how far targets are. Then pick the single best action via the BASIC ACTION LOOP. (Hearts/food icons at the bottom are permanent UI −− ignore them; the world is always visible.)

Your "thoughts" must give spatial reasoning + your predicted observation for this action + a brief check of last step's prediction vs. reality. Good: "Tree trunk is slightly left −− TurnLeft to center it, then I expect to be aimed to Dig." Bad: "I will dig a tree." (no spatial reasoning) / "I should explore." (too vague) / "NoOp" (almost never correct).

RESPONSE FORMAT −− respond with EXACTLY this JSON, no extra text:
{"thoughts": "<spatial reasoning + predicted observation ('I expect to see X next step') + check of last step's prediction vs reality>", "action": "<one action name>", "communication": "<short

Per-step action-selection prompt (user message) (continued)

message to the teammate in communication_target −− an observation, request, or commitment relevant to THEM; never empty>", "communication_target": "<a teammate's name in the form agent_N, never yourself, never 'all'>"}

H.4 SOCIAL-MODULE PROMPT

The social module (subsection 3.1) is a separate deliberation call that runs before action selection. Its prompt (Prompt H.4) presents the agent's outgoing bond row $\{W_{ij}\}_{j\neq i}$ together with each bond's change over the last 50 steps, rendered with an explicit STRENGTHENING / DECAYING / STABLE tag, and asks for a structured decision: whether to request help (from whom, with what message), whether to honour incoming requests, or to stay on task. Two fields exist purely for verification of the interpretability claim: `referenced_bonds` requires the model to copy back exactly the (teammate, weight) pairs it used, and `bond_change_explanation` requires a stated cause for every moving bond, so whether the decision was actually conditioned on the graph is auditable per step rather than assumed. The resulting directive is rendered into the {`social_directive`} block of Prompt H.3; it is advisory, and the action model remains free to choose its own `communication_target`, so `ask_target` and `ask_message` reach the agent only as text. The {`bond_table`} field is rendered per teammate as "`agent_X: w = 0.71 (Delta = +0.15) <- STRENGTHENING`".

Social-module deliberation prompt

You are the social−reasoning step for {agent_name}. Your teammates are {teammate_names}.

You decide whether to ask a teammate for help, offer help to a teammate who asked, or stay focused on your own task. The action LLM that runs after you will be told your decision and is expected to honor it.

Your social bonds (Hebbian weights, range [0, 1]; higher = stronger bond). Each line shows the current value AND the change over the last 50 steps:
{bond_table}

A rising bond (Delta $>$ 0) signals an active alliance −− cooperation has been recent and effective. A decaying bond (Delta $<$ 0) signals trust has eroded −− recent requests have been ignored, or co−presence without cooperation has thinned the connection. A stable bond near zero means you and that teammate are essentially strangers; a stable bond near 1 is a strong, well−established alliance.

Recent messages addressed to you this step:
{incoming_comms}

Your recent state:
– Last action you took: {last_action}
– Last reward you received: {last_reward}
– Current inventory: {picked_object}
– Current position: {position_text}

INSTRUCTIONS:

1. For each teammate whose bond is changing (|Delta| $>$ 0.02), write one short line in 'bond_change_explanation' explaining WHY the bond is moving −− reference the recent messages, your shared task progress, or cooperation events you remember. If a bond is stable, you may omit it.

2. Decide your social action this step:
– "ask_for_help": you should request help from a specific teammate. Choose the teammate with the highest bond who is plausibly able to help given their recent communication. Lower−bond teammates are not trustworthy enough this step. Set 'ask_target' to their agent_id (e.g. "agent_2") and 'ask_message' to the help request you want sent.

Social-module deliberation prompt (continued)

– ”offer_help”: one or more teammates have communicated need. Of the incoming senders, list in ‘respond_to‘ the agent_ids you will help −− prefer high−bond senders and those whose Delta is positive (alliance is active). Low−bond senders (w $<$ 0.2) should generally be ignored unless their request is unambiguous and urgent.
– none of the above: leave ‘ask_target = null‘, ‘ask_message = null‘, ‘respond_to = []‘. Stay focused on your own task this step.

3. In ‘referenced_bonds‘, copy back ONLY the (teammate −> weight) pairs you actually used in your reasoning. This is how we verify the bonds drove your decision.

4. In ‘reasoning‘, give 1−3 sentences tying the bonds, deltas, and incoming messages to your final decision.

RESPONSE FORMAT (always use this exact JSON structure):
{
”bond_change_explanation”: {”agent_X”: ”why this bond is moving”, ...},
”reasoning”: ”<your overall rationale>”,
”referenced_bonds”: {”agent_X”: 0.71, ...},
”ask_target”: ”agent_X” | null,
”ask_message”: ”<request text>” | null,
”respond_to”: [”agent_X”, ...],
”confidence”: 0.0 to 1.0
}

NOTE: a help request may have been in your inbox for multiple steps because the inbox is a sliding window. If you have already committed to helping a teammate on a prior step, you may keep them in ‘respond_to‘ to sustain the help, or drop them if the request seems stale or the teammate has moved on.

H.5 CHOICE-MODE ACTION PROMPT (CO-FIRING ARMS)

The co-firing arms of Section 5.2 replace the mandatory communication field with a per-step choice among the social acts their menu offers (Appendix B). Two parts of the prompt change.

In the system message, the TARGETED COMMUNICATION block of Prompt H.1 is replaced by the act menu of Prompt H.5. The menu shown here is the full Comm+Obs+Imit arm; the other arms are produced by *dropping* entries from it. The header, the engagement-level gloss and the `"none"` footer are constant across arms, one block is emitted per enabled act in the order observe, communicate, imitate, and the “Lower levels inform higher ones” line appears only when all three are enabled. Nothing else differs, so prompt length varies only by the acts an arm actually affords. An act whose channel is not in the menu is mapped to `none` in code, so the ablation does not depend on the model honouring the menu.

Choice-mode social-act menu (Comm+Obs+Imit arm)

YOUR SOCIAL ACT −− CHOOSE AT MOST ONE PER STEP

Alongside your game action you may take ONE social act aimed at ONE teammate (”social_target”, form ”agent_N”, never yourself, never ”all”). These acts are your ONLY window into what teammates are doing −− nothing else reports their progress to you, and an act you never try teaches you nothing.
Social acts are LEVELS of engagement −− the higher the level, the deeper the involvement with your teammate: Level 1 gathers information ABOUT a teammate (they are unaffected); Level 2 exchanges information WITH a teammate (both of you are involved); Level 3 lets a teammate’s behavior SHAPE YOUR OWN (their experience becomes your skill).
Lower levels inform higher ones: know who matters, align with them, learn from them.
Your available act(s):
– LEVEL 1 −− ”observe”: observe the behavior of the teammate in ”social_target” and what they have learnt. PURPOSE: an observation gives you, next step, their position, health, inventory and

Choice-mode social-act menu (Comm+Obs+Imit arm) (continued)

current beliefs. BENEFIT: knowing your teammates is the basis of working with them −− it tells you who is where, who needs help, and who is worth joining, so you act on what they are actually doing rather than on guesses.
– LEVEL 2 −− ”communicate”: send a short targeted message. PURPOSE: coordinate with a specific teammate −− requests, warnings, commitments, discoveries. Put the text in ”communication” and the SAME teammate in ”communication_target”; make it ACTIONABLE for them. BENEFIT: a team that talks aligns who does what, gets help when stuck, and spreads what one agent learned to the whole team. Sub−5−char or repeated identical messages to the same teammate are filtered as spam.
– LEVEL 3 −− ”imitate”: learn from a teammate’s behavior. PURPOSE: set ”imitate_horizon” (1−5) and next step you receive the last 1−5 actions the teammate in ”social_target” took (oldest to newest), plus their task and position. HOW TO USE IT: compare their situation to yours −− if it matches, RE−ENACT their sequence through your own ”action” choices, step by step; if not, keep it as a learned recipe and use it when you encounter a similar situation. BENEFIT: skills that already work for someone else become yours without having to discover them alone.
– ”none”: no social act this step. A real option −− but picking ”none” every step is as much a habit as spamming one act. Early in an episode, TRY the acts available to you to discover what they return; after that, engage whenever a teammate is relevant to your task or you are stuck.

In the user message, Prompt H.3 changes in three places (Prompt H.6): the opening line generalises from `communication_target` to any target field; a {`social_returns`} slot is added after the belief blocks, which is where the deliveries of subsection H.6 arrive; and the response-format line gains `social_act`, `social_target` and `imitate_horizon`. The `social_act` alternatives listed are exactly the arm’s menu plus `"none"`, and in an arm without communication the two message fields are pinned to the empty string.

Choice-mode changes to the action prompt (Comm+Obs+Imit arm)

[opening line, replaces the communication−target line]
YOU ARE: {agent_name}. Your teammates are: {teammate_names}. Any target field (”communication_target”, ”social_target”) must be EXACTLY one of {teammate_names} −− never ”{agent_name}”, never ”all”, never empty when the act needs a target.

[added after the belief blocks and {social_directive}]
{social_returns}

[response format, replaces the RESPONSE FORMAT line]
RESPONSE FORMAT −− respond with EXACTLY this JSON, no extra text:
{”thoughts”: ”<spatial reasoning + predicted observation (’I expect to see X next step’) + check of last step’s prediction vs reality>”, ”action”: ”<one action name>”, ”communication”: ”<message text IF social_act is communicate, else empty>”, ”communication_target”: ”<same teammate as social_target IF communicating, else empty>”, ”social_act”: ”communicate” | ”observe” | ”imitate” | ”none”, ”social_target”: ”<a teammate’s name in the form agent_N IF social_act is not none, else empty>”, ”imitate_horizon”: <1−5 IF social_act is imitate, else 0>}

H.6 SOCIAL-ACT DELIVERY TEXTS

Observation and imitation are deferred by one step: the act is chosen at step t and its payload appears in the initiator's {social_returns} block at step $t + 1$. Prompt H.7 gives every text that reaches an agent through that slot. The first two are what the *initiator* receives. The remaining three are the one-line notices the *target* receives, which is what makes the obs and imit channels delivery-symmetric with communication: an observation lands in the observed agent's prompt much as a message lands in a recipient's inbox. The imitation notices separate the two events the credit rule distinguishes: the "is imitating you" notice fires when the sequence is delivered, while the "imitated you" notice fires only when the imitator's own chosen action matches the next unconsumed element of that sequence, which is the same event that emits the imit co-firing signal.

Social-act delivery texts

[to the initiator of observe(j), in its next prompt]
[observation of agent_1] Position: x=12 y=8 z=41 | Chamber: ch3_cell | Status: health 20/20 | Inventory: 3 x stone
Their beliefs −− See now: {perception_beliefs of agent_1} | Task: {task_beliefs of agent_1} | Interactions: {interaction_beliefs of agent_1}

[to the initiator of imitate(j, h), in its next prompt]
[imitation of agent_1] Their task: Press the blue switch | Position: x=12 y=8 z=41 | Chamber: ch3_cell
Their last actions (oldest to newest): TurnLeft −> MoveForward −> Dig
HOW TO USE THIS: compare their situation to yours. If your position and task are similar to theirs, RE−ENACT the sequence step by step through your own "action" choices, starting with the first action. If their situation does not match yours yet, keep it as a learned recipe −− re−enact it when you find yourself in a similar situation.

[to the TARGET of an observation]
[social] agent_0 observed you last step: they can now see your position, status, inventory, and stated beliefs.

[to the TARGET when an imitation sequence is delivered]
[social] agent_0 is imitating you: they studied your last 3 actions and may re−enact them.

[to the TARGET when an imitated action is actually re−enacted −− the event that credits the imit channel]
[social] agent_0 imitated you: they re−enacted your action 'Dig' after studying your recent behavior.

H.7 THOUGHTS-ONLY ACTION PROMPT (RL AGENTS)

RL-fine-tuned agents receive the same per-step context through a thoughts-only variant of Prompt H.3: the VLM produces reasoning and a message, and the LoRA-adapted actor then scores the candidate action strings as continuations of that prompt plus the emitted thoughts (Equation (11)). The variant differs from Prompt H.3 only in the passages of Prompt H.8. The action field is removed from the response format, the model is told explicitly that a separate policy will choose the action so that it describes intent rather than naming a keyword, and the prediction example is rewritten in those terms ("I should turn left" rather than "TurnLeft"); the model is also asked to address one teammate. Everything above the response format (belief blocks, chamber state, skills, episode summary, and social directive) is byte-identical to the zero-shot prompt.

Thoughts-only variant: the lines that differ

[zero−shot] Look at the image: ... Then pick the single best action via the BASIC ACTION LOOP. (Hearts/food icons at the bottom are permanent UI −− ignore them; the world is always visible.)
[RL] Look at the image: ... Then reason about what you should do this step.

[RL, added]
IMPORTANT −− you are NOT picking the action here. A separate policy will pick the action based on YOUR thoughts plus this context. Your job is to produce the reasoning (spatial situation + intent) that will condition that policy. Describe what you intend to do; do NOT name a specific action keyword as if it were already chosen.

[zero−shot] Your "thoughts" must give spatial reasoning + your predicted observation for this action + a brief check of last step's prediction vs. reality. Good: "Tree trunk is slightly left −− TurnLeft to center it, then I expect to be aimed to Dig." Bad: "I will dig a tree." (no spatial reasoning) / "I should explore." (too vague) / "NoOp" (almost never correct).
[RL] Your "thoughts" must give spatial reasoning + your predicted observation for the next step + a brief check of last step's prediction vs. reality. Good: "Tree trunk is slightly left −− I should turn left to center it, then dig; I expect to be aimed at the trunk next step." Bad: "I will dig a tree." (no spatial reasoning) / "I should explore." (too vague).

[RL, added]
Also pick ONE teammate to address this step (nearest your target, or in the same chamber, or whose door you can open) and write them a short, useful message.

[zero−shot response format]
{"thoughts": "<spatial reasoning + predicted observation ('I expect to see X next step') + check of last step's prediction vs reality>", "action": "<one action name>", "communication": "...", "communication_target": "..."}
[RL response format −− no "action" field]
{"thoughts": "<spatial reasoning + intent for this step + predicted observation + check of last step's prediction vs reality>", "communication": "...", "communication_target": "..."}

H.8 CENTRAL-ORCHESTRATOR PROMPTS

The centralised baseline of Table 1 (Section D) uses two non-embodied LLM calls, invoked when the plan needs extending and at most every T_{soc} steps, and writes one directive block into each agent's prompt. The decomposer (Prompt H.9) turns the team goal into short-horizon subtasks, each of which must name at least one milestone id from the environment's catalog as its completion signal, so task success is decided by the same milestone events that score the agents. The allocator (Prompt H.10) then binds free agents to ready subtasks, and an agent stays locked to its assignment until the task completes, fails, or times out. Both responses are schema-validated before use: an assignment naming an unknown agent, a non-ready subtask, or an agent twice is discarded, and a multi-agent subtask that is only partially crewed is dropped whole rather than started short-handed.

What the agents themselves see is Prompt H.11, rendered into the same {`social_directive`} slot of Prompt H.3 that the social module uses in the social plasticity conditions: the two conditions differ in what fills the slot, not in where it sits or how the agent is asked to treat it.

Orchestrator: task-decomposition prompt

You are the central task decomposer for a team of {n_agents} agents ({agent_names}) in a five−chamber cooperative environment. You take NO actions and are NOT in the world. The team's global goal is to progress through all five chambers together and defeat the boss in Chamber 5. You break that goal into SMALL subtasks; a separate controller assigns them to agents; agents then work out the concrete actions themselves.

HOW THE CHAMBERS WORK (static mechanics):
{chamber_facts}

Orchestrator: task-decomposition prompt (continued)

MILESTONE CATALOG (the ONLY verifiable completion signals; a task completes when any of its listed milestones fires):
{milestone_catalog}

CURRENT WORLD STATE (step {current_step}):
{env_state_text}

WHAT EACH AGENT IS CURRENTLY DOING (their own local planners):
{task_table}

CURRENT TASK GRAPH (your earlier subtasks and their status):
{dag_summary}

INSTRUCTIONS:
1. Plan SHORT−HORIZON: propose at most {open_slots} new subtasks for the CURRENT situation. Do not plan far ahead −− you will be called again as things change.
2. Each subtask must be small, specific and achievable, and MUST list at least one milestone id from the catalog above as its completion signal. Any ONE listed milestone firing completes the task; if an objective needs several milestones, split it into several subtasks. Never target a milestone marked completed.
3. Use "required" to express order: a subtask only becomes executable when every subtask it requires has succeeded. "required" may reference ids from the task graph above or earlier tasks in THIS response. Independent subtasks run in parallel −− prefer that where possible.
4. Use "candidates" to restrict which agents may be assigned (an empty list means any agent), and "min_agents" for objectives that need several agents at once (e.g. anvils need 2 simultaneous punchers). Never propose SIMULTANEOUSLY executable subtasks whose min_agents add up to more than the team −− order such subtasks with "required" instead, so the whole crew works through them one at a time.
5. If earlier subtasks FAILED, propose a different approach for what they tried to achieve −− do not simply re−emit the same task.

Respond with EXACTLY this JSON −− ONE object, no extra text before or after it:
{
 "tasks": [
{tasks_example}
 ],
 "why": "<one sentence>"
}

Structure rules −− get these exactly right:
− "tasks" is a non−empty list; each entry has "id", "description", "milestones", "required", "candidates", "min_agents".
− Milestone ids must be copied EXACTLY from the catalog.
− Emit one complete JSON object and then STOP. Do not repeat or revise it.

Orchestrator: agent-assignment prompt

You are the central agent controller for a team in a five−chamber cooperative environment. You take NO actions and are NOT in the world. A decomposer has produced executable subtasks; YOUR job is to decide which free agent works on which subtask. Agents will be locked to your assignment until it completes, fails, or times out.

READY SUBTASKS (all prerequisites satisfied −− these can start NOW):
{ready_tasks_block}

FREE AGENTS (step {current_step}) −− their state and recent task record:
{free_agents_block}

FULL TASK GRAPH (context):

Orchestrator: agent-assignment prompt (continued)

{dag_summary}

INSTRUCTIONS:
1. Assign free agents to ready subtasks. Consider each agent's location/chamber, health, current activity, and recent successes/failures with similar work.
2. Only use agents listed among a subtask's candidates (when it names candidates).
3. A subtask that needs several agents (min_agents $>$ 1) must receive ALL of them in THIS response −− give each selected agent its own one−line "role" describing its part of the shared objective. A partially−crewed multi−agent subtask is DISCARDED.
4. NEVER split the team across several multi−agent subtasks you cannot all fully crew: when more multi−agent subtasks are ready than the free agents can cover, pick ONE of them and give it its full crew before assigning anyone elsewhere.
5. An agent may appear AT MOST once. Minimize idle agents, but leaving an agent unassigned is allowed when no ready subtask suits it.

Respond with EXACTLY this JSON −− ONE object, no extra text before or after it:
{
"assignments": [
{assignments_example}
],
"why": "<one sentence>"
}

Structure rules −− get these exactly right:
− Each entry has "agent" (an agent's exact name), "task_id" (a READY subtask id copied exactly), and "role" (one short line).
− No agent twice. Only ready subtask ids. Only listed candidates.
− Emit one complete JSON object and then STOP. Do not repeat or revise it.

Orchestrator: the directive block each agent sees

[assigned]
Team assignment (from the non−embodied coordinator; it sees the whole team):
Objective: {task description}
Your role: {this agent's one−line role, or "work on this objective"}
Teammates on this task: {other assigned agents, or "(you alone)"}
Done when milestone(s) [{milestone ids}] fire. Your actions this step MUST advance this objective.

[unassigned]
Team assignment (from the non−embodied coordinator): (none right now −− the coordinator will assign you shortly; keep making safe progress in your current chamber.)

I BIOLOGICAL MOTIVATION

Reward-modulated Hebbian plasticity. Hebbian learning strengthens connections between neurons through correlated activity ("cells that fire together, wire together" (Hebb, 2005; Brown, 2020; Lazari et al., 2022)). Reward-modulated, or three-factor, plasticity adds a global modulatory signal such as dopamine-mediated reward (Kuśmierz et al., 2017): local co-activity marks a synapse as eligible to change, and a later outcome determines whether and how much it changes. Our rule applies this structure to inter-agent relationships (Section 3.1): co-firing marks a pair as eligible through the trace e_{ij}, and milestone rewards or failure penalties consolidate or weaken the bond.

Social learning. Humans and other animals learn from one another through several channels, including observation, imitation, and communication (Tomasello et al., 1993; Boyd et al., 2011; Hoppitt & Lala, 2013), and they do so selectively: whom one attends to depends on social cues and on prior experience with that individual (Kendal et al., 2018; Dunbar, 1998). These channels motivate the three explicit co-firing cues of WIRE (Section B), and the selectivity of social learning motivates making the relationship itself, rather than only the message, the object that is learned.

I.1 THE NEURON−AGENT CORRESPONDENCE

Table 13 summarises the correspondence between neural and agent-level quantities that the plasticity rule rests on: agents act as the units, and their pairwise bonds form a plastic connectivity structure that evolves through interaction and reward. The correspondence is a design analogy that fixes the form of the update rule, not a claim about biological fidelity.

Table 13: **Neuron–agent correspondence** underlying the social graph.

Neural component	Multi-agent analogue
Neuron	Agent
Synapse	Directed social bond W_{ij}
Neural co-activity	Social co-firing c_{ij}
Eligibility trace	Traced co-activity e_{ij}
Neuromodulatory signal	Outcome salience ($\lvert r_i^{\text{bond}}\rvert / R$) and failure penalties
Synaptic plasticity	Online update of $W_{ij}(t)$
Homeostatic scaling	Decay term $-\lambda W_{ij}$

Co-firing versus adjacency. As in Pavlovian conditioning, where repeated co-activation around a salient outcome strengthens the association between two neurons, repeated socially meaningful co-firing strengthens the bond between a pair of agents, letting shared interaction history persist in the graph. Proximity alone counts only weakly: a co-located pair receives the small co-activity floor c_0, whereas explicit social acts and engaged co-location contribute the bulk of c_{ij} (Equation (5)). Combined with reward gating, this means that mere adjacency yields at most the weak association-floor bond, while consolidation requires co-activity that precedes an outcome.

J BROADER IMPACT

Our experiments are conducted in a simulated environment, so the implications below concern possible future uses of learned social plasticity rather than demonstrated real-world effects.

Adaptive coordination and oversight. Persistent relational state could support systems in which specialised agents must repeatedly coordinate under changing conditions, such as robotics, autonomous systems, healthcare, or disaster response, where the usefulness of a collaborator may depend on previous interactions rather than only on the current task. The social graph also exposes an explicit representation of coordination: it shows how pairwise connections evolve and which collaborators become strongly linked. It does not explain the agents' internal reasoning, but because it is explicit it can be inspected and modified directly; our RQ3 probes, which impose, freeze, or transfer bonds, are examples of such interventions. This could support auditing, debugging, and human oversight of increasingly autonomous multi-agent systems, in contrast to social structure that forms implicitly in message histories.

Controllability of self-organising coordination. Self-organising coordination introduces a tension between adaptability and control. As interaction structure is learned from local experience, the resulting organisation may diverge from the one a system designer would have specified, especially as tasks, populations, and interaction histories change. In high-stakes settings this can make it harder to guarantee which agents influence one another, how strongly, and whether the emergent structure remains aligned with system-level objectives. Adaptive relational mechanisms should therefore operate within explicit constraints and remain subject to monitoring or intervention.

Bias amplification through relational structure. Persistent partner preferences can turn local biases into persistent structural inequalities within a population. If early interactions reflect biases in agent capabilities, training data, reward signals, or task assignment, some agents may be selected more often and therefore receive more opportunities to influence decisions and strengthen their relationships, while others are progressively excluded even when the initial differences were small or spurious. Because relational state feeds back into future interaction, such biases can become self-reinforcing. Systems that learn social structure should be evaluated not only for task performance but also for whether interaction opportunities and influence become systematically concentrated.

Computational and environmental cost. Scaling intelligence through populations of agents can substantially increase computational demand. Our deliberation-frequency sweep (Section F.3) shows that more frequent, and therefore more expensive, social deliberation does not improve cooperative performance, highlighting the importance of weighing additional compute against the gains it provides, both for practical scalability and for limiting environmental cost.