



Wired Together: Reward-Modulated Hebbian Social Plasticity for Emergent Social Intelligence in Multi-Agent Systems

Ana Cristiana Marcu¹

Supervisor: Chirag Raman¹

¹EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
in Partial Fulfilment of the Requirements
for the Master of Data Science and AI Technology

September 14, 2026

Name of the student: Ana Cristiana Marcu
Final project course: MSc DSAIT Thesis project EEMCS
Supervisor: Chirag Raman
Thesis committee: Chirag Raman, Matthijs Spaan

WIRED TOGETHER: REWARD-MODULATED HEBBIAN SOCIAL PLASTICITY FOR EMERGENT SOCIAL INTELLIGENCE IN MULTI-AGENT SYSTEMS

Ana Cristiana Marcu
Delft University of Technology

ABSTRACT

Multi-agent systems increasingly rely on communication and orchestration to coordinate, yet their interactions are often transient rather than accumulated into persistent relationships. We ask whether persistent relational structure can instead emerge from interaction itself, and formalise a **neuron-agent isomorphism** that models a MAS as a social graph evolving through reward-modulated Hebbian plasticity. The graph is coupled back into agent behaviour through reward diffusion and weight-gated experience sharing for RL-fine-tuned agents, and through an inference-time social module for VLM-based agents. We evaluate the framework in *WIRE*, a five-stage embodied Craftium environment with increasing coordination demands. We characterise when social plasticity supports coordination, showing that its effect depends on the underlying agent and the choice of co-firing signal. On Gemma-E4B, Hebbian coupling increases mean task return from 521 to 570 and cooperative milestone completion from 8.5% to 12.1%, with longer-timescale reward modulation further increasing cooperative completion to 14.4%. Decentralised plasticity-based coordination is also competitive with centralised orchestration at matched compute. Different social cues induce distinct relational structures, with observation yielding the highest cooperative completion (16.3%) despite communication producing substantially stronger bonds. Finally, when initial bond strength is controlled, agents preferentially collaborate with teammates with whom they had previously co-fired with (0.56 vs. 0.40 under random re-pairing) and retain stronger bond, showing that **agents that co-fire together wire together** even after team recomposition. The code can be found at <https://github.com/tapri-lab/wired-together>.

1 INTRODUCTION

Multi-agent systems (MAS) are increasingly proposed as a substrate for problems that exceed the practical capacity of single agents, particularly in domains such as disaster response, transportation, and web automation, where parallelism, partial observability, and large-scale dynamics are intrinsic to the task [1, 2]. Multi-Agent Reinforcement Learning (MARL) offers a framework for such systems to learn cooperative joint policies through environmental interaction [1, 3]. In parallel, recent LLM-based agent systems combine language-model reasoning with planning, memory, and natural-language communication to coordinate toward shared goals [4, 5].

Yet across both lines of work, agents typically treat one another as transient sources of information rather than as enduring partners. Coordination is achieved through learned vector messages [6], LLM-mediated dialogue [5], attention over the population [7], or centralised and hierarchical orchestrators that decompose tasks, assign agents, or mediate information flow [8, 9]. Recent work makes the organisation itself adaptive [10], but the state these mechanisms carry forward is task-level: plans, progress, assignments, rather than the relationship that develops between agents. Computational trust and reputation models do maintain such persistent pairwise state [11], but represent it as an estimate of a partner’s reliability used at the point of partner selection rather than as structure that shapes the agents’ own learning. Self-evolving frameworks introduce online adaptation of agent components [12, 13, 14], but this adaptation largely occurs within individual agents, while the structure of relationships between agents remains fixed or implicit. This gap becomes consequential in *open*

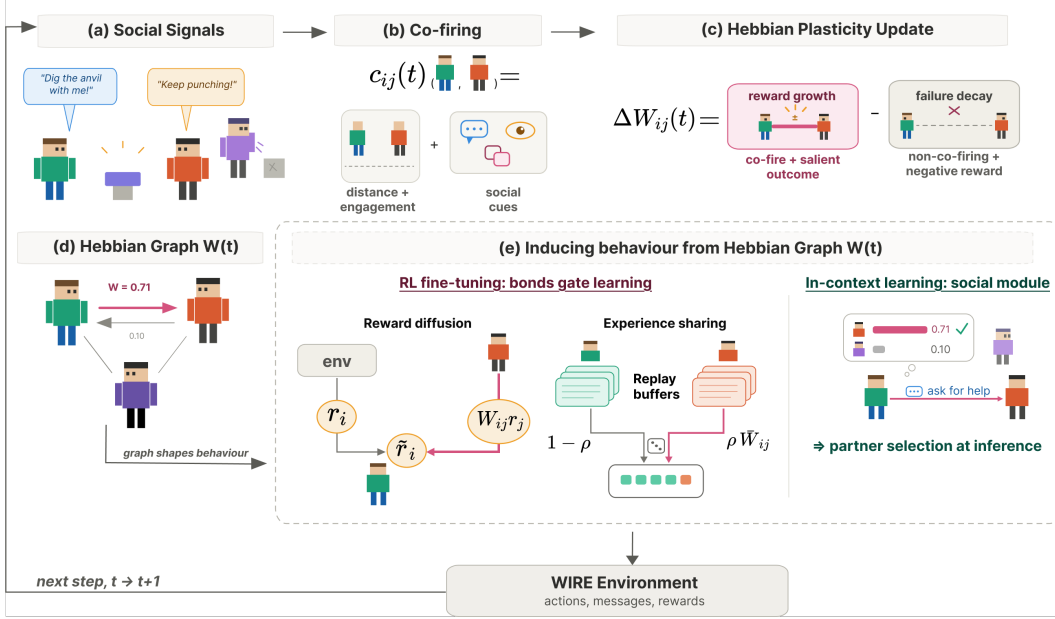


Figure 1: **Overview of Hebbian social plasticity in MAS.** Social interactions and spatial engagement define pairwise co-firing signals, which drive reward-modulated updates to a directed social graph $W(t)$. The learned graph is then coupled back to the agents through reward diffusion and experience sharing during RLFT, or through bond-conditioned partner selection for VLM agents at inference time.

environments, where populations and tasks change dynamically [1]. Task-level state is discarded when the task that produced it ends, so which agents worked well together must be rediscovered rather than reused.

This distinction has an analogue in human social learning. Humans acquire knowledge not only through direct experience but also from others, allowing skills and information to accumulate socially across individuals [15, 16, 17]. Moreover, social learning is selective: individuals do not learn from all partners equally, but condition whom they learn from on social cues and prior experience [18]. Existing work on social learning between artificial agents [19, 20] captures learning from others, but generally does not model persistent, partner-specific relationships that can shape whom an agent learns from across repeated interactions.

We propose persistent inter-agent relationships as a form of structural plasticity at the population level. Inspired by biological neural systems, where learning depends on plastic changes in the connections between neurons [21, 22], we model a MAS as an adaptive network: agents correspond to neurons, pairwise social bonds to synapses, and reward to a modulatory signal. We introduce a reward-modulated Hebbian rule under which social co-firing and outcome salience jointly shape these bonds over time. The resulting graph is coupled back to behaviour through reward diffusion and experience sharing for reinforcement learning fine-tuned (RLFT) based agents, and bond-conditioned social deliberation for VLM based agents. This formulation lets us ask whether agents that co-fire together also wire together: whether shared interaction can produce persistent relational structure that shapes whom agents collaborate with later. We view this adaptive graph as a step toward decentralised coordination in which information about previous collaborators is retained in the population structure itself and can shape subsequent interactions. Figure 1 gives an overview of the framework.

Contributions. This research makes the following contributions: (i) we formalise a neuron-agent isomorphism that models a MAS as a time-evolving social graph governed by reward-modulated Hebbian plasticity, and introduce graph couplings for reward diffusion, experience sharing, and inference-time social deliberation; (ii) we introduce *WIRE*, an open-source Craftium environment with five stages of increasing coordination demand; (iii) we characterise when social plasticity supports coordination, showing that its effect depends on the backbone and co-firing signal, that longer-

timescale reward modulation improves cooperative performance, and that decentralised plasticity can be competitive with compute-matched centralised orchestration; and (iv) we show that agents that co-fire together wire together: different social cues induce distinct relational structures, while shared co-firing history produces persistent partner preferences that imposed bond topology alone does not.

Research questions. We evaluate Hebbian social plasticity through three research questions:

- RQ1.** Does reward-modulated Hebbian plasticity over inter-agent bonds improve cooperative task performance relative to matched baselines without a social graph?
- RQ2.** How do different co-firing channels affect the strength and informativeness of learned bonds, and how do these differences relate to cooperative task performance?
- RQ3.** Do agents that fire together wire together?

2 BACKGROUND

Hebbian Learning. Hebbian learning describes a class of synaptic plasticity mechanisms in which connections between neurons are strengthened through correlated activity, commonly summarised as “cells that fire together, wire together” [21]. Repeated co-activation can therefore alter synaptic strength and support associative learning [23, 24]. Related forms of synaptic plasticity include long-term potentiation (LTP), which strengthens synaptic connections, and long-term depression (LTD), which weakens them [24].

Reward-modulated Hebbian plasticity extends this principle by introducing a third, global modulatory signal, such as dopamine-mediated reward or prediction error [22]. Synaptic change is then determined jointly by local co-activity and outcome salience, allowing associations formed during interaction to be selectively reinforced or weakened according to their subsequent outcomes. This three-factor formulation motivates our treatment of inter-agent relationships as plastic connections whose evolution depends on both social co-firing and reward.

Social Learning. Human learning is not limited to direct experience: individuals also acquire knowledge and behaviour from others, enabling information to propagate and accumulate socially [15, 17]. The Social Brain Hypothesis further links the evolution of primate cognition to the demands of maintaining complex social relationships [25]. Social learning can occur through several channels, including observation, imitation, and communication [16]. Importantly, it is selective: individuals differ in whom they attend to and learn from according to social cues and prior experience [18]. These mechanisms motivate the communication, observation, and imitation signals that we study as alternative sources of inter-agent co-firing.

Dec-POMDPs and cooperative MARL. Cooperative multi-agent environments are commonly formalised as Decentralised Partially Observable Markov Decision Processes (Dec-POMDPs) [26]. At each step, agent i receives a local observation o_i and selects an action a_i according to a decentralised policy $\pi_{\theta_i}(a_i \mid o_i)$. The agents jointly interact with a shared environment while optimising a cooperative objective under partial observability [2]. A common approach to learning in this setting is centralised training with decentralised execution (CTDE) [27]. During training, centralised information can be used to estimate values or learning signals, while each actor remains conditioned only on information available locally at execution.

PPO and MAPPO. Proximal Policy Optimisation (PPO) is an on-policy actor-critic algorithm that updates a policy using trajectories collected under its current behaviour. During optimisation, PPO compares the probability of each sampled action under the updated and behaviour policies and clips this probability ratio, preventing individual samples from driving overly large changes to the policy. A learned value function provides the baseline used to estimate advantages for the policy update.

Multi-Agent PPO (MAPPO) extends PPO to cooperative multi-agent learning under CTDE [28]. Each agent executes a decentralised actor conditioned on its local observation, while a centralised critic can use information from the joint system state during training. The critic provides advantage estimates for updating the actors without introducing centralised control at execution time.

3 RELATED WORK

This work connects research on multi-agent coordination and orchestration, neuron-level models of agency, Hebbian plasticity in artificial systems, and social learning in foundation-model agents.

Coordination and orchestration in multi-agent systems. Existing multi-agent systems coordinate behaviour through a range of mechanisms, including learned communication and attention in MARL [6, 7], social-influence objectives [29], and natural-language interaction between LLM agents [5]. LLM-based systems increasingly impose additional structure through centralised or hierarchical orchestrators that decompose tasks, assign agents, and regulate the progression of collaboration [30, 8, 9]. More recent approaches make this organisation adaptive, learning orchestration policies that dynamically select agents and induce task-dependent interaction topologies [10]. These approaches demonstrate that coordination structure can itself be adaptive, but typically represent this adaptation through task-level state or a central orchestration policy. Computational trust and reputation models instead maintain persistent pairwise state [11], but use it to select partners rather than to shape learning. Our work considers persistent pairwise relational state that evolves from agents’ shared interaction history and couples back into reward, experience sharing, and deliberation.

Social learning in artificial agents. Social learning has also been studied in artificial multi-agent systems, where agents acquire behaviour or knowledge through interaction with others [20]. Ndousse et al. [19], for example, show that agents can learn from experts by predicting their actions through an auxiliary objective, enabling socially acquired behaviour without direct access to the expert policy. These approaches focus primarily on how information or behaviour is transferred between agents.

Neuron as an agent. A complementary line of work models components of neural networks themselves as interacting agents. Neurons have been formulated as reinforcement-learning agents whose local policies collectively produce network-level computation [31, 32], while other work places the learning agent at the synapse, framing credit assignment as a local reinforcement-learning problem [33]. Distributed mechanisms have also been proposed for coordinating communication and reward among neural units [34], and Ott [35] argues more broadly that global intelligence can emerge from interactions among self-interested neurons balancing local and global objectives. We invert this perspective: rather than modelling neurons within a network as agents, we model a population of agents as a network and study plasticity in the connections between them.

Hebbian learning in artificial systems. Hebbian learning has been used as a local plasticity mechanism for adaptive behaviour in both neural and multi-agent systems. In multi-agent settings, Hebbian-style updates have been used to model social factors such as empathy and counter-empathy [36], to cluster agents from patterns of interaction in environments such as StarCraft II [37], and to study adaptive networks in which Hebbian plasticity interacts with synchronisation and social-bond dynamics [38]. In parallel, Hebbian rules have been explored as general learning mechanisms, including evolved shared plasticity rules [39] and meta-learned bidirectional Hebbian updates [40]. These approaches use Hebbian plasticity to adapt internal learning dynamics, behavioural relationships, or abstract network structure. Our work instead applies Hebbian plasticity directly to the pairwise bonds between interacting agents.

Lifelong and embodied foundation-model agents. Foundation models increasingly serve as the reasoning core of embodied agents, supporting planning, tool use, memory, and long-horizon interaction [4]. A prominent line of work focuses on accumulating reusable competence across experience. Voyager [41], for example, combines an automatic curriculum with a growing library of executable skills, allowing previously acquired behaviours to be reused across tasks. MindForge [14] extends this setting to multiple embodied agents, adding inter-agent communication, structured theory of mind, and a multi-component memory system. Optimus-2 [42] similarly combines high-level multimodal reasoning with a learned low-level policy conditioned on recent interaction history. These systems demonstrate how skills, memories, and behavioural knowledge can persist within individual agents.

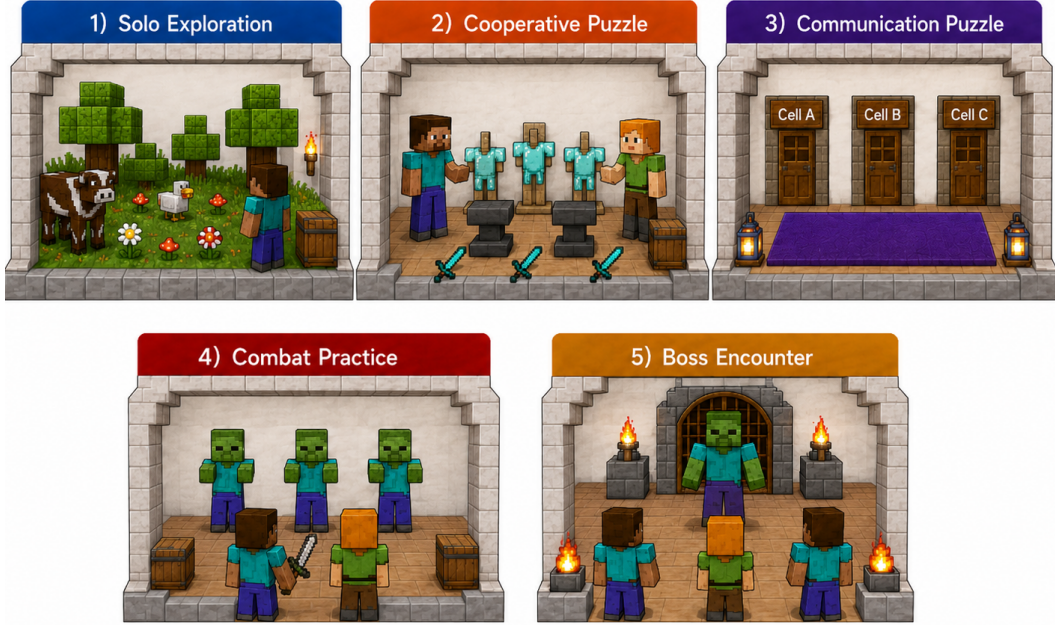


Figure 2: The WIRE environment (Wired Inter-agent Reasoning Evaluation). Agents progress through five chambers, each targeting a distinct coordination competency: solo skill acquisition (Ch1), cooperative resource acquisition requiring synchronised action (Ch2), targeted communication under partial observability (Ch3), team combat (Ch4), and a cooperative boss fight (Ch5). Episodes terminate on boss defeat or full team death.

4 METHOD

4.1 ENVIRONMENT AND AGENT ARCHITECTURE

We introduce *WIRE* (Wired Inter-agent Reasoning Evaluation), a five-stage embodied environment implemented in Craftium [43] on top of the Luan game engine, designed to study social plasticity under controlled coordination demands (Figure 2). Agents progress from individual skill acquisition to cooperative resource collection, targeted communication under partial observability, and cooperative combat; Figure 3 shows the corresponding first-person views available to the agents. Each stage introduces a different source of inter-agent dependence, creating repeated opportunities for social co-firing while varying the degree and form of coordination required. This controlled progression allows us to measure how social bonds form and how the resulting graph affects subsequent learning and behaviour.

Each agent is built on the *MindForge* cognitive stack [14], which provides the underlying reasoning, memory, and communication components (see Appendix B). At each step, agent i receives a first-person visual observation, structured cognitive context, and messages from teammates, and produces a reasoning step, a discrete environment action, and a targeted natural-language message.

4.2 HEBBIAN SOCIAL PLASTICITY

Neuron-Agent Isomorphism. We model a multi-agent system as an adaptive network by drawing an analogy between neurons and agents, and between synapses and persistent inter-agent relationships. In biological neural systems, collective computation emerges not only from the activity of individual neurons, but also from plastic changes in the connections between them. We transfer this principle to MAS: agents act as computational units, while their pairwise social bonds form a plastic connectivity structure that evolves through interaction and reward. Table 1 summarises this analogy.

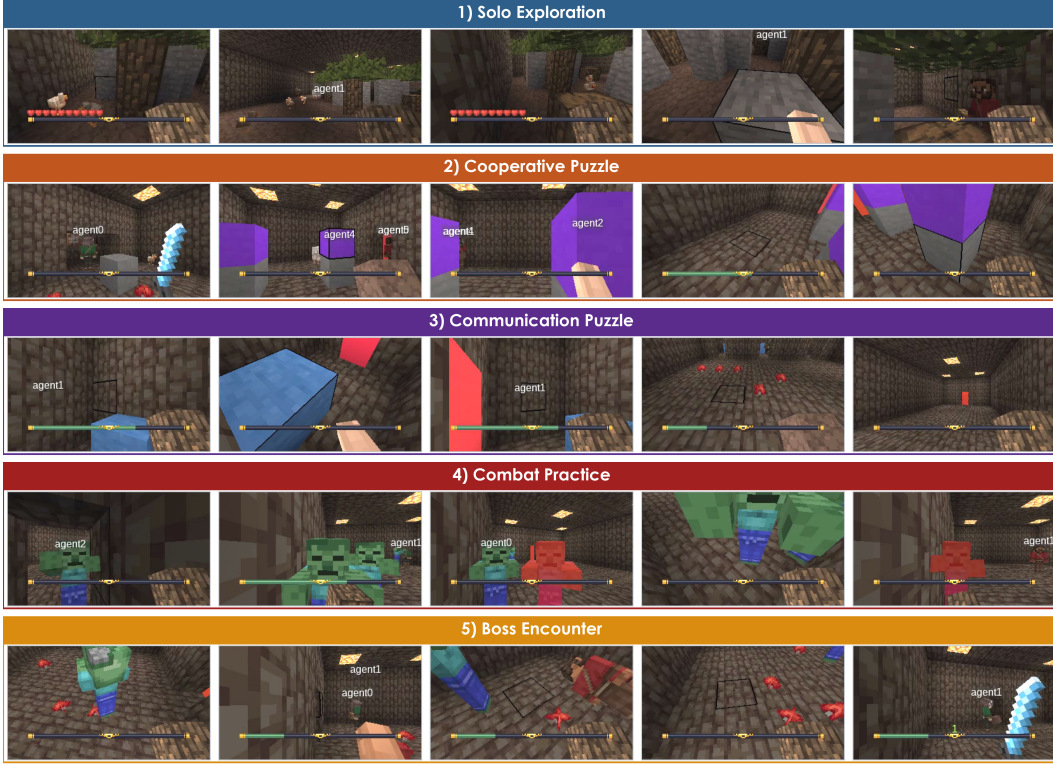


Figure 3: **First-person agent views in WIRE.** Observations from each of the five chambers.

Table 1: Neuron-agent mapping used to model the MAS as an adaptive network.

Biological component	Multi-agent analogue
Neuron	Agent
Synapse	Social bond W_{ij}
Neural co-activity	Social co-firing c_{ij}
Synaptic plasticity	Online update of $W_{ij}(t)$
Reward modulation	Outcome salience

Computational mapping. We formalise the MAS as a population of N agents with decentralised policies $\pi_{\theta_i}(a_i | o_i)$ and introduce a weighted social graph

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

where $W_{ij}(t)$ denotes the social bond strength from agent i to agent j at time t . The graph evolves online through a reward-modulated Hebbian rule: co-firing determines which bonds are eligible to change, while outcome salience modulates the magnitude of that change.

Illustrative example. A useful intuition comes from Pavlovian conditioning. Consider a neuron A activated by a tone and a neuron B involved in the response to food. Initially, the connection between them is weak. When tone and food repeatedly occur together around a salient outcome, the two neurons become co-active and their association strengthens. We transfer the same intuition to agents: repeated socially meaningful co-firing can strengthen the bond between them, allowing their shared interaction history to persist in the social graph.

Co-firing signal. The definition of co-firing can vary across environments, but its central idea is that meaningful social interaction, rather than proximity alone, should determine which agents are coupled. In Hebbian learning, synaptic plasticity is driven by neurons that co-fire, not merely by neurons that are anatomically close. We preserve this distinction at the agent level: spatial proximity provides

an opportunity for interaction, but does not by itself constitute a social event [44]. Instead, we treat communication, observation, and imitation as explicit forms of agent-level co-firing because the action or information produced by one agent directly participates in the activity of another. Together with spatial engagement, these signals define $c_{ij}(t)$, our operational analogue of neural co-activity.

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

$$c_{ij}^{\text{spat}}(t) = \mathbb{1}[\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\| \leq d] g_i(t) g_j(t), \quad (3)$$

$$c_{ij}^{(k)}(t) = \delta_k \mathbb{1}[\text{event}_{ij}^{(k)}(t)], \quad k \in \mathcal{K}, \quad (4)$$

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

The engagement score $g_i(t)$ captures whether agent i is actively involved in the current interaction, combining outcome salience with social activity; $r_{\max}$ normalises reward magnitude and $\mathcal{A}(t)$ denotes the agents participating in an active social event. Spatial co-firing requires both agents to be within distance d and actively engaged. Explicit social events contribute directly through $c_{ij}^{(k)}(t)$, where δ_k controls the contribution of cue k . Together, spatial engagement and the enabled social cues define the co-firing signal $c_{ij}(t)$.

We instantiate explicit social co-firing through $\{\text{comm}, \text{obs}, \text{imit}\}$, with an active subset $\mathcal{K}$ enabled in each experiment. Spatial co-firing remains present across conditions; our baseline uses communication as the sole explicit social cue. To study which social signals drive wiring and which support task performance, we additionally consider *observation*, the deliberate act of reading a teammate’s state and current beliefs, and *imitation*, the act of requesting and re-enacting a teammate’s recent action sequence. In both cases, the observed or imitated teammate receives an explicit signal that the interaction occurred. For imitation, an adoption event $\text{adopt}_{ij}(t)$ fires whenever the imitator executes the next action in the delivered sequence. In the extended experiments, each agent may perform at most one social act per step.

The update rule. We instantiate a reward-modulated Hebbian update [22] with three components: co-activity-driven growth, failure-gated weakening, and a smaller homeostatic term that prevents unbounded bond growth (Equation 7, Algorithm 2). Co-firing determines which bond is eligible for strengthening, while the magnitude of the experienced outcome determines how strongly that association is reinforced.

Growth is gated by co-activity and scaled by the salience of what happened to agent i at that step. This mirrors reward-modulated Hebbian potentiation: co-activity marks an eligible connection, while the modulatory signal determines the strength of the update. We use $|r_i^{\text{bond}}|$ so that a salient shared event can strengthen the association regardless of whether its outcome was positive or negative. The association floor η_0 further allows bonds to grow from repeated co-firing even on zero-reward steps.

Decay provides the complementary weakening mechanism. When recent co-activity between i and j falls below ϵ and agent i experiences a negative outcome, the failure gate $\phi_{ij}(t)$ produces additional bond-specific decay. A smaller homeostatic term continuously prevents unbounded growth.

$$\phi_{ij}(t) = \mathbb{1}[\text{coop}_{ij}(t) < \epsilon \wedge \text{neg}_i(t)], \quad (6)$$

$$\Delta W_{ij}(t) = \underbrace{\left(\eta_0 + \eta_+ \frac{|r_i^{\text{bond}}(t)|}{R}\right) c_{ij}(t) (1 - W_{ij}(t))}_{\text{reward-modulated growth}} - \underbrace{\eta_- \phi_{ij}(t) W_{ij}(t)}_{\text{failure-gated decay}} - \underbrace{\lambda W_{ij}(t)}_{\text{homeostatic decay}}, \quad (7)$$

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

4.3 SOCIAL GRAPH COUPLINGS

The learned Hebbian graph is coupled back into agent learning and behaviour through three mechanisms. For RLFT based agents, we intervene at two complementary points in the learning process: *reward diffusion* uses social bonds to shape the reward signal, while *weight-gated experience sharing* uses them to determine whose experience contributes to each agent’s policy update. For VLM agents, an inference-time *social module* uses the graph to guide partner selection and social deliberation.

Reward diffusion through social bonds. The first RL coupling acts on the learning signal. Before entering the PPO objective, agent i ’s reward is mixed with rewards received by teammates along the current social graph:

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

where $\bar{W}_{ij} = W_{ij} / \sum_{k \neq i} W_{ik}$ denotes the row-normalised bond and $\gamma_d \in [0, 1]$ controls diffusion strength. The co-firing term $c_{ij}(t)$ gates diffusion to currently active relationships, while $\bar{W}_{ij}$ determines how strongly each teammate contributes.

Weight-gated experience sharing. The second RL coupling acts on the experience used for learning. Each agent maintains a rollout buffer $\mathcal{D}_i$ containing its own transitions. At update time, agent i augments its training pool with experience from other agents according to its learned bonds:

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

where $\rho \in [0, 1]$ controls the fraction of socially shared experience. Agents whose bonds fall below the graph sparsity threshold do not contribute samples.

Reward diffusion and experience sharing therefore expose the graph to two different parts of the learning process: the former changes the reward associated with an agent’s trajectory, while the latter changes which trajectories the agent learns from. Because shared transitions originate from another agent’s policy, they are treated off-policy: each transition retains its behaviour log-probability under the source policy, and the PPO probability ratio provides the corresponding importance correction, with clipping limiting large cross-policy updates [45]. The learned graph therefore gates both *what* an RL agent learns from and *whose experience* contributes to that learning (Algorithm 4).

Social module. For VLM agents, the graph is instead coupled to behaviour at inference time. At each social-deliberation step, agent i receives its current bond row $\{W_{ij}\}_{j \neq i}$ together with the recent change in each bond ΔW_{ij} over a fixed window. These signals are incorporated into the agent’s reasoning context, allowing both bond strength and recent relational change to influence whether it requests help, offers help, or remains on its current task. When requesting help, stronger reachable bonds bias partner selection toward agents with whom i has accumulated greater interaction history; decreasing bonds provide the converse signal. The graph therefore conditions social deliberation rather than imposing a fixed communication topology. Full details are given in Algorithm 3.

4.4 REINFORCEMENT LEARNING LAYER

We formulate WIRE as a decentralised partially observable Markov decision process (Dec-POMDP) [26] with N homogeneous agents pursuing a common cooperative objective. At step t , agent i receives a local observation o_i^t , comprising its first-person visual frame, structured cognitive context, and incoming messages, and produces $(th_i^t, a_i^t, m_i^t, \tau_i^t)$: a reasoning step, discrete environment action, natural-language message, and message target. The environment then returns an agent-specific reward r_i^t .

Policy. Each actor is a LoRA-adapted vision–language model operating under *constrained decoding*. Given observation o , every valid environment action $a \in \mathcal{A}$ is scored by the summed token log-probability of its action string,

$$s_{\theta_i}(a, o) = \sum_k \log p_{\theta_i}(a^{(k)} \mid o, a^{(<k)}) , \quad (11)$$

and induces the discrete policy

$$\pi_{\theta_i}(a \mid o) = \frac{\exp s_{\theta_i}(a, o)}{\sum_{a' \in \mathcal{A}} \exp s_{\theta_i}(a', o)}. \quad (12)$$

Only the LoRA adapter is trained; the shared base model remains frozen [46, 47]. The reasoning step and message are generated before environment action selection. The reasoning is appended to the agent’s context, after which a_i^t is sampled from $\pi_{\theta_i}(a \mid o_i^t, th_i^t)$. Thus, reasoning can condition action selection, while PPO optimises only the discrete action policy; reasoning and message generation receive no direct policy-gradient objective.

Centralised critic and advantage estimation. We train the actors with PPO under two critic configurations: *MAPPO*, using centralised training with decentralised execution (CTDE), and *IPPO*, using independent local critics [28, 27]. Under MAPPO, a shared critic $V_\psi(s^t)$ receives an explicit encoding of the joint environment state, including agent state, inventory and milestone progress, together with embeddings of recent actions and communication. Actors remain decentralised and condition only on their local observations. Advantages are computed from the diffused reward r_i^t introduced in Equation 9:

$$\delta_{i,t} = r'_{i,t} + \gamma(1 - d_{i,t})V_\psi(s^{t+1}) - V_\psi(s^t), \quad (13)$$

$$\hat{A}_{i,t} = \sum_{l \geq 0} (\gamma\lambda)^l \delta_{i,t+l}, \quad (14)$$

where $d_{i,t}$ denotes termination. Advantages are standardised per rollout before the actor update.

The shared MAPPO critic is trained on returns derived from the team-mean diffused reward over the set of living agents $\mathcal{N}_t$,

$$\bar{r}'_t = \frac{1}{|\mathcal{N}_t|} \sum_{i \in \mathcal{N}_t} r'_{i,t}. \quad (15)$$

Under IPPO, each agent instead maintains a local value head over its own model representation, with no cross-agent information available to the critic.

Optimisation and training procedure. Actors are optimised with the standard clipped PPO objective, together with an entropy bonus annealed over training [28]. When weight-gated experience sharing is enabled (Equation 10), transitions originating from another agent retain their source-policy behaviour log-probability so that the corresponding PPO probability ratio accounts for the cross-policy update [45].

Rollouts are collected synchronously. At each step, the joint state s^t is encoded for the centralised critic, while all living agents concurrently select actions from their local observations o_i^t . The environment then advances under the resulting joint action, and the per-agent rewards are diffused through the current social graph before being stored in the corresponding rollout buffers. Each actor performs a PPO update after accumulating K transitions, while the centralised critic is updated on the same cadence from its team-level buffer. An N -agent run therefore maintains a single frozen copy of the base model together with N lightweight trainable adapters. Full optimisation objectives, critic architecture, and hyperparameters are provided in subsection E.1.

5 EXPERIMENTS AND RESULTS

We evaluate the Hebbian Social Graph through three experiment families, corresponding to the research questions introduced in section 1. Unless otherwise stated, experiments use $N = 3$ agents in WIRE with episodes of up to 1,000 steps, divided evenly across the five chambers. When a chamber’s budget expires (20% of the step budget each), the team is rescue-teleported to the next chamber without receiving its remaining milestones; episodes terminate early only if all agents die in Chamber 5.

We report mean $\pm$ standard deviation over pooled episodes across seeds. *Task return* is computed from the original environment reward, excluding Hebbian reward diffusion so that all conditions are evaluated on the same scale. Full hyperparameters and qualitative coordination examples are provided in Appendix E and Appendix G, respectively.

RQ1: Does reward-modulated Hebbian plasticity over inter-agent bonds improve cooperative task performance relative to matched baselines without a social graph? We compare each Hebbian condition (+Heb) against a matched no-graph baseline across Qwen3.5-2B, Qwen3.5-9B, and Gemma-E4B VLM agents (from Gemma-4 family of models), together with IPPO and MAPPO RLFT agents. For RLFT agents, the graph enters learning through reward diffusion and experience sharing; for zero-shot VLM agents, it enters inference through the social module.

We additionally compare Gemma-E4B against a compute-matched centralised orchestration baseline. The centralised baseline, inspired by VillagerAgent [8], observes global task state, decomposes the objective into subtasks, and assigns them to agents while leaving primitive control and communication decentralised (Appendix C).

RQ2: How do different co-firing channels affect the strength and informativeness of learned bonds, and how do these differences relate to cooperative task performance? Using Gemma-E4B, we vary the explicit social cues included in the co-firing signal, $\mathcal{K} \subseteq \{\text{comm, obs, imit}\}$. We evaluate each cue individually and in combination against a *Null* condition with no explicit social acts; proximity-based co-firing remains active in all conditions. Each condition uses 3 seeds with 3 episodes per seed.

RQ3: Do agents that fire together wire together? We separate the effect of bond magnitude from the effect of shared interaction history through two complementary experiments. First, a frozen-topology ablation tests whether imposed social structure alone changes cooperative behaviour. Second, a two-phase partner-transfer experiment tests whether relationships formed through genuine co-firing wire together.

For the frozen-topology ablation, we evaluate three fixed graphs on Qwen3.5-9B: *Allied-all*, where every pair has bond strength 0.8; *Allied-pair*, where only agents 0 and 1 are bonded; and *No-bonds*, where $\mathbf{W} = 0$. The graph is exposed through the social module but never updated, isolating the effect of imposed topology.

For the partner-transfer experiment, agents first interact in isolated dyads during Phase A. Two dyads co-fire through successful cooperative anvil interactions, while a third co-fires through communication only. In Phase B, all six agents are merged into one population starting in Chamber 3. Agents retain their Phase A memories, but all within-pair bonds are reset to the same value ($W = 0.265$), eliminating bond magnitude as a confound. We compare *Co-fired Partner*, where agents are reunited with their actual Phase A partners with whom they co-fired, against *Re-paired Partner*, where they are paired with agents from different Phase A runs.

5.1 RESULTS AND ANALYSIS

Metrics. We report three primary task-performance metrics. *All milestones* is the fraction of the 25 environment milestones completed at least once by the team; *cooperative milestones* restricts this measure to the 17 milestones in Chambers 2–5 (these include per-agent milestones that are only reachable after cooperative progress); and *task return* is the team-summed undiffused environment reward. *Steps to first completion* additionally measures the first successful completion of selected cooperative milestones, reporting both median step and number of successful episodes.

RQ1: DOES REWARD-MODULATED HEBBIAN PLASTICITY OVER INTER-AGENT BONDS IMPROVE COOPERATIVE TASK PERFORMANCE RELATIVE TO MATCHED BASELINES WITHOUT A SOCIAL GRAPH?

Table 2 compares task performance across the VLM and RL policies, with each baseline paired with its corresponding Hebbian variant. To further examine how the learned social graph relates to behaviour, Table 3 reports the correlation between final bond strength and directed messaging, joint digging, and proximity. Full milestones completion timeline can be found Table 12.

The effect of Hebbian plasticity depends on the underlying base model. The largest gains occur for Gemma-E4B and MAPPO. For Gemma-E4B, Hebbian plasticity increases task return from 521 to 570 and cooperative milestone completion from 8.5% to 12.1% (rows (e)–(g)); for MAPPO, return increases from 390 to 470 and cooperative completion from 4.6% to 7.8% (rows (j)–(k)). In contrast,

Table 2: **RQ1: Does reward-modulated Hebbian plasticity over inter-agent bonds improve cooperative task performance relative to matched baselines without a social graph?** Mean $\pm$ std over pooled episodes across completed seeds (3 episodes per seed). All conditions run across 6 seeds except the RLFT Hebbian rows (i) and (k) and Gemma-E4B+Heb2.0 (g'), which run across 3 seeds. Within each matched baseline–Hebbian pair, the better result is shown in **bold**; the best result within each policy family is additionally underlined. For steps to first completion, x_n denotes a median completion step of x over n successful episodes; “–” indicates that the milestone was never completed. For these metrics, the number of successful episodes is prioritised over speed. **Finding:** Hebbian plasticity improves aggregate performance for Gemma-E4B and MAPPO, but not uniformly across backbones. On Gemma-E4B, it also achieves higher mean performance than a compute-matched centralised orchestrator (3.78 vs. 3.81×10^{17} inference FLOPs per run).

Condition	Task return $\uparrow$	Milestone %		Steps to first completion			
		Milest. $\uparrow$	Coop. $\uparrow$	M8 anvil	M9 anvil	M13 switch	M17 mob
(a) Qwen3.5-2B	358 $\pm$ 163	22.9 $\pm$ 4.6	9.2 $\pm$ 4.9	–	–	434 ₁₄	–
(b) Qwen3.5-2B+Heb	363 $\pm$ 196	20.4 $\pm$ 5.6	6.9 $\pm$ 5.6	–	–	421 ₁₀	756 ₁
(c) Qwen3.5-9B	<u>622</u> $\pm$ 150	<u>31.1</u> $\pm$ 5.1	<u>17.6</u> $\pm$ 6.2	201 ₂	218 ₂	<u>415</u> ₁₈	<u>679</u> ₁₀
(d) Qwen3.5-9B+Heb	574 $\pm$ 123	27.3 $\pm$ 5.2	14.1 $\pm$ 5.6	301 ₁	–	403 ₁₇	696 ₇
(e) Gemma-E4B	521 $\pm$ 117	16.4 $\pm$ 4.8	8.5 $\pm$ 4.9	–	199 ₁	406 ₇	670 ₁₀
(f) Gemma-E4B+Central Orch.	505 $\pm$ 119	21.1 $\pm$ 5.8	11.1 $\pm$ 5.5	–	–	407 ₁₁	692 ₁₂
(g) Gemma-E4B+Heb	570 $\pm$ 125	22.0 $\pm$ 5.5	12.1 $\pm$ 5.0	–	–	407 ₁₂	662 ₁₃
(g') Gemma-E4B+Heb2.0	569 $\pm$ 99	23.1 $\pm$ 4.5	14.4 $\pm$ 4.9	–	–	440 ₇	686 ₈
(h) Qwen3.5-2B IPPO	425 $\pm$ 99	18.3 $\pm$ 4.5	6.9 $\pm$ 5.8	–	–	488 ₇	–
(i) Qwen3.5-2B IPPO+Heb	427 $\pm$ 108	18.2 $\pm$ 4.7	5.2 $\pm$ 5.8	–	–	456 ₄	–
(j) Qwen3.5-2B MAPPO	390 $\pm$ 86	17.8 $\pm$ 4.5	4.6 $\pm$ 5.7	–	–	486 ₇	–
(k) Qwen3.5-2B MAPPO+Heb	470 $\pm$ 110	20.4 $\pm$ 4.4	7.8 $\pm$ 5.5	–	–	490 ₆	–

Table 3: **Correlation between bonds and cooperation events.** Spearman ρ between end-of-episode bond W_{ij} and per-pair counts of messages, joint digs, and proximity steps. **Finding:** The learned bonds correlate strongly with who communicates, but not consistently with who physically cooperates.

Condition	Messages	Joint digging	Proximity
Qwen3.5-2B+Heb	+0.58 $\pm$ 0.18	−0.03 $\pm$ 0.33	+0.07 $\pm$ 0.24
Qwen3.5-9B+Heb	+0.64 $\pm$ 0.07	+0.36 $\pm$ 0.17	+0.40 $\pm$ 0.24
Gemma-E4B+Heb	+0.58 $\pm$ 0.25	+0.04 $\pm$ 0.26	−0.20 $\pm$ 0.26
IPPO+Heb	+0.72 $\pm$ 0.05	−0.20 $\pm$ 0.27	+0.30 $\pm$ 0.22
MAPPO+Heb	+0.76 $\pm$ 0.06	+0.03 $\pm$ 0.27	+0.32 $\pm$ 0.14

Qwen3.5-2B and IPPO show little change in task return while losing some cooperative completion, and Qwen3.5-9B decreases across the aggregate metrics. These results indicate that social plasticity does not provide a uniform performance gain: its effect depends on how the underlying base model uses the relational information exposed by the learned graph.

The effect of the social module depends on underlying coordination strategies. Behavioural logs suggest that the backbone-dependent effect arises partly from how each model interprets the relational signal. Qwen3.5-9B already coordinates through frequent language-based negotiation, but under the Hebbian condition it requests help in fewer than 3% of social deliberations and directed request traffic falls by roughly 40%. Its deliberations also place greater weight on the absolute bond values than on recent ΔW : because bonds remain relatively low, small positive changes are often treated as insufficient evidence to alter partner choice, suppressing negotiation that the baseline would otherwise initiate. Gemma-E4B shows the complementary pattern: its baseline rarely initiates

purposeful negotiation, while the persistent bond context keeps selected teammates salient across subsequent action prompts. Consistent with this interpretation, Gemma-E4B maintains the same partner for roughly eight times longer than its baseline. Thus, the module’s effect depends both on how the backbone reads the bond signal and on the coordination behaviour already present without it.

Hebbian coordination is competitive with centralised orchestration at matched compute. On Gemma-E4B, both Hebbian variants achieve higher mean performance than the centralised orchestrator at comparable inference cost. The original Hebbian condition reaches 570 task return, 22.0% overall milestone completion, and 12.1% cooperative completion, compared with 505, 21.1%, and 11.1% for centralised orchestration. Heb2.0 further increases milestone completion to 23.1% overall and 14.4% cooperative while preserving task return at 569. Since the original Hebbian and centralised conditions have nearly identical inference cost (3.78 vs. 3.81×10^{17} FLOPs per run), these gains are not explained by additional inference compute.

Learned bonds primarily capture communication structure. Across Hebbian conditions, bond strength correlates consistently with directed messaging ($\rho = 0.58\text{--}0.76$), but much less consistently with joint digging or proximity (Table 3). This suggests that the choice of co-firing signal strongly shapes the relational structure learned by the graph.

Longer-timescale reward modulation improves cooperation. The original Hebbian rule modulates bond growth using only reward available at the current step, which limits temporal credit assignment under WIRE’s sparse rewards. Heb2.0 instead maintains an eligibility trace over recent co-activity, allowing delayed rewards to reinforce the interactions that preceded them, and uses weaker homeostatic decay (subsection F.1). On Gemma-E4B, Heb2.0 preserves task return while increasing overall milestone completion from 22.0% to 23.1% and cooperative completion from 12.1% to 14.4%. These results suggest that longer-timescale credit assignment can better align learned relationships with delayed cooperative outcomes. We hypothesise that the same modification may benefit backbones for which the original update is less effective, although this remains to be tested.

RQ2: HOW DO DIFFERENT CO-FIRING CHANNELS AFFECT THE STRENGTH AND INFORMATIVENESS OF LEARNED BONDS, AND HOW DO THESE DIFFERENCES RELATE TO COOPERATIVE TASK PERFORMANCE?

Table 4 examines how the choice of social co-firing cue changes both the learned graph and cooperative performance. We compare communication, observation, and imitation individually and in combination against a condition with no explicit social acts.

Bond strength alone does not determine cooperative value. The three social cues produce different relationships between wiring and task performance. Observation achieves the highest overall (24.0%) and cooperative (16.3%) milestone completion while forming lower bonds ($W = 0.10$, row (b)). Communication shows the opposite pattern: it is used in 84% of social-action opportunities and forms substantially stronger bonds ($W = 0.28$), yet cooperative completion falls to 9.2% (row (d)). Imitation is rarer (9%) and produces weak bonds ($W = 0.02$), but 96% of its bond growth is directly attributable to imitation events (row (c)). Together, these results show that the magnitude of a learned bond reflects how frequently and specifically agents co-fire, but stronger connectivity does not by itself imply more effective cooperation.

Adding more social cues does not improve coordination. Combining cues does not recover the performance of the strongest single channel. *Comm+Obs* reaches 9.2% cooperative completion, while *Comm+Obs+Imit* reaches 6.5%, both below observation alone (16.3%) and no better than communication on its own. The per-cue statistics also reveal strong competition between social actions: observation use drops from 27% in isolation to 8.2% with communication and 4.3% when all cues are available, while imitation is not selected at all in the three-cue condition. Because each agent may perform at most one social act per step, enabling additional cues does not add social capacity but instead turns cue selection into a further decision the agent must make at every step, and the agents resolve it almost entirely in favour of communication. Thus, exposing agents to more possible social interactions does not ensure that informative cues are actually used. Whether cues can be made

Table 4: **RQ2: How do different co-firing channels affect the strength and informativeness of learned bonds, and how do these differences relate to cooperative task performance?** Each condition enables the listed explicit social cues; *None* retains only proximity-based co-firing. ΔW_{own} is the fraction of bond growth attributable to the listed cue, and $\rho(W, \text{acts})$ is the Spearman correlation between final bond strength and directed cue use. Each condition uses 3 seeds with 3 episodes per seed. Task return is reported without social cue based reward. Best and second-best results in each column are shown in **bold** and underlined, respectively. **Finding:** Observation achieves the highest task and cooperative performance, while communication produces the strongest bonds and imitation the most cue-specific bond growth.

Condition	Task return $\uparrow$	Milestone %		Mean W	Cue	Per-cue wiring		
		All $\uparrow$	Coop. $\uparrow$			Act use (%)	ΔW (%)	$\rho(W, \text{acts})$
(a) None	<u>307</u> $\pm$ 92	19.1	<u>11.1</u>	0.00	–	–	–	–
(b) Obs	324 $\pm$ 74	24.0	16.3	0.10	Obs	27	<u>87</u>	0.34
(c) Imit	<u>307</u> $\pm$ 81	<u>21.8</u>	<u>11.1</u>	0.02	Imit	9	96	0.49
(d) Comm	217 $\pm$ 68	18.7	9.2	<u>0.28</u>	Comm	84	80	<u>0.76</u>
(e) Comm+Obs	222 $\pm$ 70	18.2	9.2	0.24	Comm	64	66	0.80
					Obs	8.2	13	0.40
					Comm	<u>76</u>	70	0.71
(f) Comm+Obs+Imit	225 $\pm$ 57	16.4	6.5	0.29	Obs	4.3	5.9	0.29
					Imit	0.0	0	–

complementary rather than competing by structuring how an agent chooses between them remains open.

Co-firing produces reciprocal bonds. Although the update rule is directed, the learned graph is nearly symmetric in practice. Across Hebbian conditions, the two directions of a pair differ by a mean of only 0.0028. This reciprocity emerges from the co-firing events themselves: proximity and communication typically involve both members of a pair, providing similar evidence for strengthening W_{ij} and W_{ji} . Directional cues such as observation and imitation produce somewhat larger asymmetries when used in isolation, but the two directions remain close.

RQ3: DO AGENTS THAT FIRE TOGETHER WIRE TOGETHER?

RQ3 separates the effect of bond magnitude from the effect of shared interaction history. Table 6 first tests whether an imposed social topology alone changes cooperative performance. Table 5 then tests whether relationships formed through genuine co-firing persist in a new social context, while controlling the initial bond strength.

Imposed bond structure alone does not yield a consistent cooperative advantage. In the frozen-topology experiment, neither *Allied-pair* nor *Allied-all* consistently outperforms *No-bonds*. At 1,000 steps, *No-bonds* achieves the highest cooperative completion and task return (14.7%, 571), whereas at 2,000 steps *Allied-all* achieves the highest cooperative completion (17.6%). The ordering therefore changes with episode horizon and remains within the observed variability. These results indicate that imposing strong bonds is not sufficient on its own to reliably improve cooperation.

Genuine shared history produces persistent partner preference. All pairs enter Phase B with identical within-pair bond strength ($W = 0.265$), yet agents in the *Co-fired Partner* condition message their real former partners more often than agents in the *Re-paired Partner* condition (0.56 vs. 0.40; rows (a)–(b)). Former partners also exchange more within-pair messages and finish with higher bond strengths. Because initial bond magnitude is controlled, the difference is associated with whether the current partner matches the agent’s retained interaction history. Thus, shared co-firing history continues to influence partner selection after the population is recomposed.

Richer co-firing histories are associated with stronger partner preference. The two dyads that shared the cooperative anvil experience show the largest partner preferences (0.63 and 0.59; rows

Table 5: **RQ3: Do agents that fire together wire together?** *Co-fired Partner* reunites agents with their true Phase A partners; *Re-paired Partner* pairs agents with agents from different Phase A runs. All within-pair bonds are equalised to $W = 0.265$ at the start of Phase B. Results are mean $\pm$ population SD over 3 seeds $\times$ 3 episodes. Phase B starts in Chamber 3, so Chamber 1 and 2 milestones are unreachable; percentages are not comparable to other results. **Finding:** Agents prefer their genuine former partners despite identical initial bond strengths; pairs with richer cooperative co-firing show the strongest partner preferences.

Condition	Task return $\uparrow$	Milestone %		Pref. $\uparrow$	Pair	Per-pair		
		All $\uparrow$	Coop. $\uparrow$			Partner pref.	Msgs within	Bond W
(a) Co-fired Partner	402 $\pm$ 224	13.8 $\pm$ 2.0	17.0 $\pm$ 1.8	0.56 $\pm$ 0.12	(a-1) Pair 1 (co-fired)	0.63 $\pm$ 0.25	1045 $\pm$ 423	0.216 $\pm$ 0.082
					(a-2) Pair 2 (co-fired)	0.59 $\pm$ 0.32	957 $\pm$ 510	0.207 $\pm$ 0.099
					(a-3) Pair 3 (control)	0.53 $\pm$ 0.31	857 $\pm$ 481	0.186 $\pm$ 0.100
(b) Re-paired Partner	306 $\pm$ 250	8.4 $\pm$ 3.5	9.2 $\pm$ 6.9	0.40 $\pm$ 0.12	(b-1) Pair 1 (strangers)	0.41 $\pm$ 0.28	536 $\pm$ 484	0.182 $\pm$ 0.097
					(b-2) Pair 2 (strangers)	0.33 $\pm$ 0.28	509 $\pm$ 424	0.156 $\pm$ 0.073
					(b-3) Pair 3 (strangers)	0.44 $\pm$ 0.33	688 $\pm$ 506	0.169 $\pm$ 0.076

Table 6: **RQ3: Frozen-topology horizon.** Fixed *No-bonds*, *Allied-pair*, and *Allied-all* graphs are evaluated at 1,000 and 2,000 steps on the same two seeds. **Finding:** No imposed topology consistently dominates across horizons; the relative ordering changes between 1,000 and 2,000 steps.

Condition	Steps	Milestone %		Task return $\uparrow$
		Coop. $\uparrow$	All $\uparrow$	
No-bonds	1,000	14.7 $\pm$ 4.5	26.7 $\pm$ 5.0	571 $\pm$ 98
	2,000	15.7 $\pm$ 6.5	29.3 $\pm$ 7.2	821 $\pm$ 230
Allied-pair	1,000	10.8 $\pm$ 5.3	26.0 $\pm$ 3.1	532 $\pm$ 91
	2,000	16.7 $\pm$ 6.3	30.0 $\pm$ 2.0	644 $\pm$ 124
Allied-all	1,000	12.7 $\pm$ 2.2	26.0 $\pm$ 3.1	539 $\pm$ 84
	2,000	17.6 $\pm$ 6.8	30.7 $\pm$ 5.0	796 $\pm$ 222

(a-1)–(a-2)), while the communication-only control pair shows a lower preference (0.53; row (a-3)). The same control pair still prefers its real former partner over its re-paired counterpart (0.44; row (b-3)). This suggests that communication alone can support a persistent partner preference, while richer cooperative interaction is associated with stronger selectivity.

Partner preference survives temporary re-pairing. Partner selection can vary across episodes even when previous relationships remain recoverable. In one run, agents return to their original dyads in episodes 1 and 3 but reorganise into different pairs in episode 2. The re-emergence of the original pairs in the following episode suggests that the learned preference acts as a persistent bias rather than a fixed assignment: agents can temporarily explore alternative partners without permanently erasing prior relational structure.

5.2 COMPUTE–PERFORMANCE TRADE-OFFS

We examine how the social coupling trades off performance against inference compute along two axes: how frequently agents deliberate over the learned graph and how the total compute scales with population size. Compute is reported as inference FLOPs per episode, estimated as $2N_{\text{eff}}(n_{\text{prompt}} + n_{\text{decode}})$ and summed over all VLM calls made by agents and auxiliary modules.

Sparse social deliberation retains comparable performance. 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 $T_{\text{soc}} \in \{2, 8, 20, 50, 100\}$ on Gemma-E4B against the no-module baseline from RQ1.

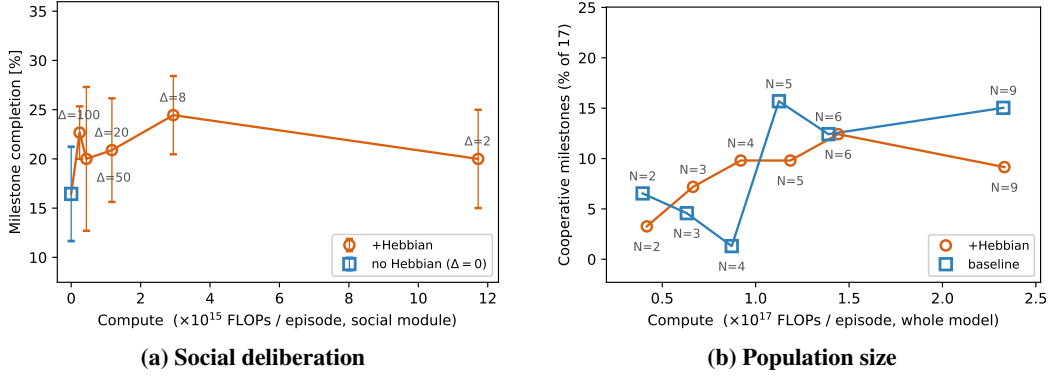


Figure 4: **Compute–performance trade-offs of Hebbian social plasticity.** (a) Milestone completion as social deliberation frequency T_{soc} varies, against social-module inference compute. (b) Cooperative milestone completion as population size N varies, against whole-system inference compute.

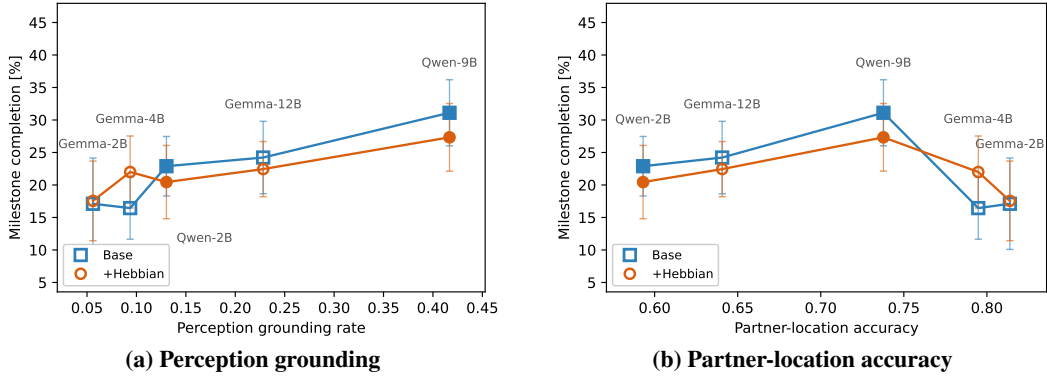


Figure 5: **Task performance against backbone perception measures.** Milestone completion is plotted against (a) perception grounding and (b) partner-location accuracy across five backbones. Perception grounding measures whether free-text observations identify objects consistent with the current chamber; partner-location accuracy measures whether claims about teammates match their true chamber.

Performance varies little across deliberation intervals (Figure 4a): all settings reach approximately 20–22% milestone completion, compared with 16.4% for the baseline, while social-module compute decreases sharply as deliberation becomes less frequent. In particular, $T_{\text{soc}} = 100$ reaches the same mean completion as the best-performing $T_{\text{soc}} = 8$ setting while requiring roughly 12 $\times$ less social-module compute. The differences between deliberation intervals remain within the observed variability, suggesting that the learned graph need not be queried at every few environment steps to remain useful.

Population scaling does not yield a consistent advantage from the social module. We vary the population over $N \in \{2, 3, 4, 5, 6, 9\}$ using Gemma-E4B while keeping the remaining social hyperparameters fixed. Cooperative milestone completion is measured at the team level, whereas whole-system inference compute grows approximately linearly with population size.

The effect of the social module is non-monotonic (Figure 4b). It improves cooperative completion over the matched baseline for some intermediate populations, most notably $N = 3$ and $N = 4$, but provides little or negative benefit for several other team sizes. By $N = 9$, the uncoupled population achieves higher cooperative completion despite lower social-module overhead. Thus, increasing population size adds inference compute without producing a consistent gain from the current social coupling. One likely limitation is that the graph and social hyperparameters are held fixed as the number of potential partners grows.

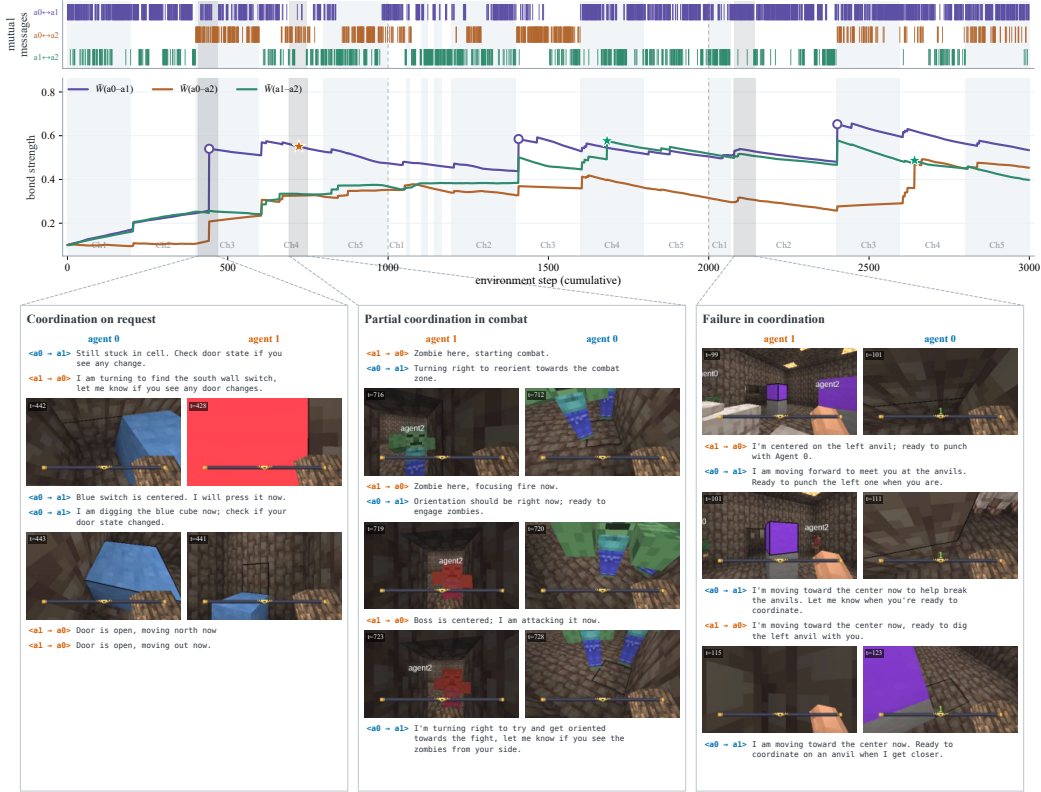


Figure 6: **Qualitative coordination dynamics under Hebbian social plasticity.** Gemma-E4B+Heb2.0, seed 123, across three episodes. *Top*: mutual pairwise messages and bond strengths $\bar{W}$ over the five WIRE chambers; circles denote joint milestone completions and stars denote mob kills. *Bottom*: representative interaction windows showing successful request-driven coordination on the communication puzzle, partial coordination during combat, and failed synchronisation on the cooperative anvil task. Together, these examples illustrate that persistent interaction and communication can structure collaboration, but do not by themselves guarantee successful joint execution.

5.3 BACKBONE CAPABILITY ANALYSIS

Figure 5 examines whether the effectiveness of Hebbian social plasticity depends on the perceptual capabilities of the underlying backbone. General perception grounding does not show a clear relationship with the benefit of the social coupling (Figure 5a): Qwen3.5-9B achieves the strongest grounding but still decreases in milestone completion when the Hebbian mechanism is introduced.

Hebbian gains concentrate among backbones with stronger partner localisation. Partner-location accuracy shows a different relationship with Hebbian performance (Figure 5b). The backbones with lower partner-location accuracy, namely Qwen3.5-2B, Gemma-12B, and Qwen3.5-9B, all show lower milestone completion under the Hebbian coupling. In contrast, the two backbones with the highest partner-location accuracy, Gemma-E4B and Gemma-E2B, show a positive or approximately neutral effect. This suggests that relational information is difficult to exploit when an agent cannot reliably ground its teammates in the environment: knowing *who* is strongly bonded is useful only if the agent can also identify *where* that partner is. The Hebbian gains are concentrated among the backbones with the highest partner-location accuracy, suggesting that accurate localisation of social partners may help agents translate learned relational structure into effective coordination.

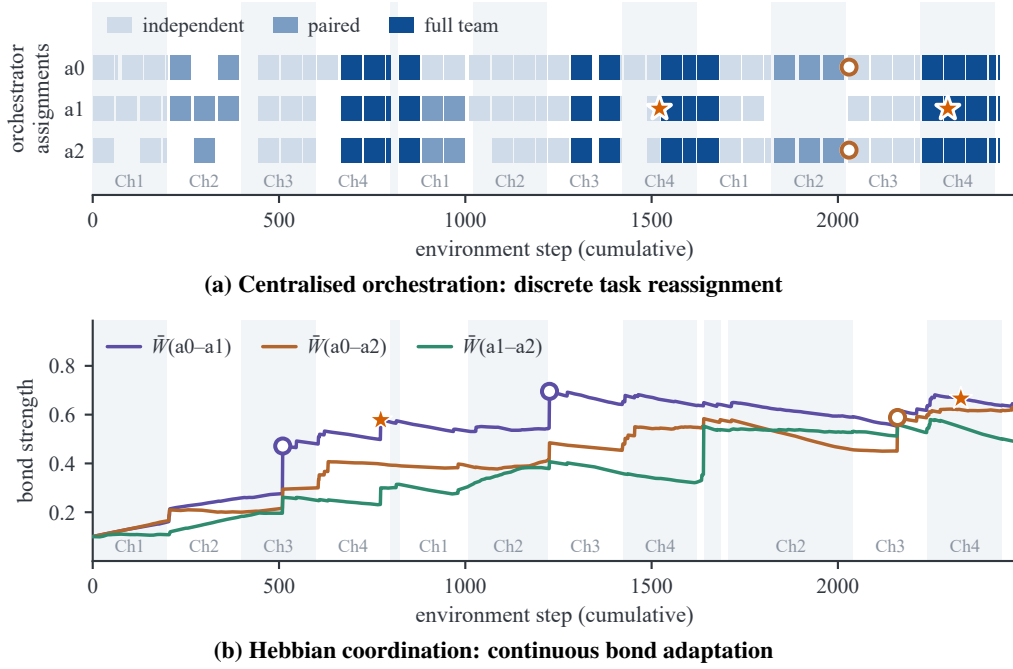


Figure 7: **Centralised task assignments and Hebbian bond dynamics.** Gemma-E4B, across three episodes in separate runs of the two conditions. (a) Orchestrator task assignments for each agent, grouped by coordination scale. (b) Symmetrised pairwise bond strengths under Heb2.0. Circles denote joint milestone completions and stars denote mob kills.

6 DISCUSSION

Relational structure does not guarantee joint execution. Figure 6 illustrates the distinction between forming a relationship and successfully coordinating through it. In the first example, an explicit request identifies a concrete dependency: one agent reports being blocked, the other locates the corresponding switch, and the pair completes the interaction. In combat, agent 0 responds to agent 1’s call for support and attempts to reorient towards the fight, but fails to position itself in time, leaving agent 1 to engage alone. The anvil example shows a different failure: both agents repeatedly communicate readiness to cooperate, yet fail to synchronise their actions. Together, these cases show that social information can induce appropriate collaborative intent without guaranteeing successful embodied execution. Effective cooperation still depends on the underlying agents’ ability to ground, navigate, and temporally coordinate their actions.

Centralised orchestration schedules collaboration but does not sustain it. Figure 7 places the orchestrator’s task assignments and the Hebbian bond trajectories side by side on the same seed. The two mechanisms differ in what they carry forward. The orchestrator allocates *tasks*: partner structure is a by-product of an assignment and is dissolved whenever the plan is revised. Paired work is rare (16% of assigned agent-steps, against 51% individual and 33% full-team), a further 16% of agent-steps carry no assignment at all, and pairings are recomposed whenever a subtask fails, so no partner structure survives from one chamber to the next. Because most objectives are issued to a single agent, an agent has little basis for recognising that its own subtask is one half of a joint one. The Hebbian graph instead carries the *relationship*: bonds are the coordination state, they are updated continuously rather than reissued, and they persist across chamber boundaries. On this seed, the same pair a_0-a_1 remains the strongest bond for 94% of the run, and each credited joint milestone is completed by the currently top-ranked pair. The comparison therefore illustrates a distinction between explicitly scheduling collaboration and maintaining an adaptive relational state across tasks.

Limitations. Our experiments remain small-scale, with populations of three to nine agents and relatively short interaction horizons, limiting the opportunity for richer partner differentiation and

population structure to emerge. The number of runs also leaves several effects noisy. Moreover, the graph influences behaviour indirectly, so learning an informative relationship does not ensure that the underlying policy can execute the corresponding coordinated behaviour, as Figure 6 illustrates. Communication, observation, and imitation also occur at different frequencies, making absolute bond magnitude difficult to compare directly across cues. Although Heb2.0 improves temporal credit assignment, it has so far been evaluated only on Gemma-E4B. Finally, evaluation is restricted to WIRE and a small set of model backbones.

Future work. A first direction is to evaluate Heb2.0 across the remaining backbones and learning settings to test whether longer-timescale reward modulation also improves conditions in which the original rule provides limited or negative gains. Larger and more heterogeneous populations, together with longer interaction horizons, would further test whether persistent relational state supports meaningful partner differentiation and coalition formation. Stronger graph-behaviour couplings are another important direction: rather than conditioning deliberation alone, learned bonds could directly influence action selection, delegation, communication routing, or experience sharing. Finally, evaluating in more open and dynamic environments would test whether learned relationships remain useful when collaborators, tasks, and population composition change over longer timescales.

7 CONCLUSION

We introduced *reward-modulated Hebbian social plasticity*, modelling a multi-agent system as an adaptive network whose pairwise relationships evolve from social co-firing and reward. Across VLM and RLFT agents in WIRE, the effect is not uniform: Hebbian coupling improves performance for Gemma-E4B and MAPPO, while longer-timescale reward modulation further improves cooperative completion on Gemma-E4B. On the same backbone, Hebbian coordination also achieves higher mean performance than a compute-matched centralised orchestrator, showing that adaptive relational structure can be competitive with explicit central planning at comparable inference cost.

The learned structure depends strongly on what constitutes co-firing. Communication produces the strongest bonds, whereas observation achieves the highest cooperative performance, showing that bond magnitude alone does not determine cooperative value. The usefulness of the graph also depends on the underlying agent: gains concentrate among backbones that more reliably localise their partners, suggesting that relational information is only useful when agents can ground the collaborators it refers to.

Most importantly, our results provide evidence that *agents that fire together wire together*. Imposed bond topologies provide no consistent cooperative advantage, whereas agents reunited with genuine former partners with whom they had previously co-fired preferentially return to them and sustain stronger bonds than under random re-pairing, even when initial bond strengths are matched. Together, these results suggest that social plasticity can support persistent social structure grounded in shared experience. More broadly, they point toward multi-agent systems that retain interaction history not only as individual memory, but in the evolving relationships that shape future collaboration.

8 IMPACT ON THE FIELD OF DATA SCIENCE AND AI TECHNOLOGIES

From centralised orchestration to self-organising coordination. Multi-agent systems increasingly rely on centralised or hierarchical orchestration to decompose tasks, assign agents, and structure collaboration [30]. Even when this organisation is adaptive, coordination is typically represented through a central policy or task-level state [10]. Our work explores a different direction, in which part of the coordination structure emerges from local agent interactions. Inspired by neural plasticity, we treat pairwise connectivity as adaptive and update it through reward-modulated Hebbian learning. The broader implication is that coordination can itself be treated as an object of learning. Rather than fixing how agents interact or repeatedly reconstructing this structure from a central policy, multi-agent systems learn and retain their own patterns of organisation over time based on previous interaction signals. Moreover, decentralised coordination of this form requires no component to reason over the population as a whole, which may matter as populations grow beyond what a single orchestration policy can usefully represent.

Relational state as a flexible computational substrate. The learned social graph is not tied to a single behavioural interpretation. In our framework, the same pairwise state modulates reward diffusion, experience sharing, and inference-time partner selection, acting as an interface between interaction history and subsequent computation. More generally, relational state could influence information flow, delegation, learning, or aggregation across a population. For example, independent peer evaluations at test time could be weighted according to learned relational state, distributing verification across agents rather than relying solely on additional reasoning within a single model [48]. This positions inter-agent relational state as a general design variable for organising computation in multi-agent systems.

Evaluating adaptive multi-agent organisation. Lifelong multi-agent learning also calls for a different evaluation paradigm. Benchmarks built from isolated cooperative tasks reveal whether agents can coordinate in a particular setting, but not how their organisation evolves as the demands of collaboration change over time. Evaluating adaptive organisation therefore requires environments that expose the same population to a sequence of distinct forms of inter-agent dependence, allowing previously learned coordination structure to be reused, adapted, or replaced. WIRE provides one controlled instance of this principle. Evaluation should likewise extend beyond task performance alone, incorporating measures of relational structure, behavioural alignment, and how organisation persists or reorganises as tasks and collaborators change. More broadly, lifelong multi-agent benchmarks should capture both what a population achieves and how its organisation evolves across experience.

9 SOCIETAL IMPACT

The following discussion considers the most relevant potential benefits and risks of this work. Since our experiments are conducted in a simulated environment, these implications concern possible future uses of Hebbian Social Learning rather than demonstrated real-world effects.

9.1 POSITIVE IMPACTS

Adaptive coordination in multi-agent systems. Persistent relational state could support systems in which specialised agents must repeatedly coordinate under changing conditions. Examples include robotics, autonomous systems, healthcare, and disaster response, where the usefulness of a collaborator may depend on previous interactions rather than only on the current task. Allowing coordination structure to adapt from experience could make such systems more responsive to changing capabilities, team composition, and operating conditions. Our results are limited to WIRE, but they provide an initial mechanism for studying how such persistent relationships might be represented and updated.

Explainability of multi-agent coordination. Explicit relational state can make multi-agent coordination more explainable. When collaboration emerges through model-internal representations or extended interaction histories, it may be difficult to understand why particular agents repeatedly interact or influence one another. The social graph exposes an interpretable intermediate representation of these relationships, showing how pairwise connections evolve and which collaborators become more strongly linked. While this does not explain the agents’ internal reasoning, it provides a transparent view of the coordination structure itself, which could support auditing, debugging, and human oversight of increasingly autonomous multi-agent systems.

9.2 RISKS

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 explicitly, 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 they do so, 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 when necessary.

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. Others may be progressively excluded from collaboration, even when the initial differences were small or spurious. Because relational state feeds back into future interaction, such biases can become self-reinforcing over time. Systems that learn social structure should therefore be evaluated not only for task performance, but also for whether interaction opportunities and influence become systematically concentrated across the population.

Computational and environmental cost. Scaling intelligence through populations of agents can substantially increase computational demand, since additional agents introduce parallel inference, communication, and coordination overhead. Our population-size experiments show that greater computational cost does not automatically imply better cooperative performance, highlighting the importance of evaluating the trade-off between additional compute and the gains it provides. More broadly, as multi-agent systems scale, performance should be considered alongside resource use to assess whether larger populations or more frequent coordination are justified. This is important both for practical scalability and for limiting the environmental cost of increasingly compute-intensive agent systems.

10 USE OF AI

AI tools, primarily Claude and OpenAI models, were used throughout this thesis as supporting tools during both the research and writing process. Their use varied across stages of the project, including brainstorming experimental ideas, discussing possible ways to structure analyses, debugging implementation issues, and assisting with scripts for data extraction, processing, and visualisation. They were also used to explore alternative ways of presenting figures and results, and to improve the clarity, conciseness, and organisation of parts of the written report.

AI tools additionally assisted with technical tasks such as L^AT_EX formatting, code outlining, and identifying possible issues in drafts or implementations. One of the figures in the thesis, Figure 2, was also generated with the assistance of an AI image-generation tool and subsequently selected and incorporated by the author. In all cases, the resulting suggestions were reviewed and adapted before being incorporated into the thesis. All AI-assisted outputs were reviewed and validated by the author before being incorporated into the thesis.

11 ACKNOWLEDGMENT

Five years ago, I arrived at TU Delft not quite knowing where this journey would take me. As I write these final pages, I realise that these years shaped far more than my education; they changed the way I think, the way I work, and, in many ways, the person I became. This thesis felt like a fitting final chapter to that journey, because it reinforced something I had learnt along the way: that research is often less about finding the right answers and more about figuring out which questions are worth asking.

During this thesis, I was fortunate to be part of the TAPRI lab, a community where we could share unfinished ideas, learn from each other’s experiences, and go through the ups and downs of research together. It made what can often be a solitary journey feel much less so.

I want to thank Adina, Lemon, Lara, and Mirela for the many conversations and moments we shared along the way; Madalina, Kees, Max, and Bogdan for helping me through the uncertainty of this research and, just as importantly, the laughs in between; and Mircea, Alex, and Artur for being part of this journey. Thank you also to Ojas for the early brainstorming sessions that helped shape the methodology when much of the project was still only an idea.

A particular thank you goes to Anuj. I am especially grateful that, whenever the direction of the thesis felt unclear, he made the time to offer his guidance and help me find a way forward. His feedback helped me navigate some of the “earthquakes” that inevitably come with research, and helped shape the thesis into what it is today.

My deepest thanks go to my supervisor, Chirag. His role extended far beyond supervising a Master’s thesis. He gave me the freedom to pursue ambitious ideas while knowing when to challenge them, and helped me with not research decisions, but also questions about my career and personal growth. His approach to supervision has shaped my own understanding of what good mentorship can be, and it is the kind of mentor I can only hope to become one day. Thank you for believing in me, including at times when I struggled to see it myself.

Beyond the thesis, I owe an enormous amount to the people who have been beside me throughout these five years. To my parents and my sister, Alexandra, thank you for your constant love and support. To the friends I met at university, thank you for all the memories along the way.

And to Mati, thank you for being beside me through the uncertainty of this final chapter, for listening to the doubts, the ideas, the frustrations, and the excitement, and for making the difficult days easier and the good ones better.

There are inevitably people and moments that cannot fit onto this page. To everyone who has been part of these five years, whether for all of them or only a small part: thank you. This thesis may mark the end of my time as a student at TU Delft, but the people I met and the things I learned along the way are what I will remember most.

REFERENCES

- [1] L. Yuan, Z. Zhang, L. Li, C. Guan, and Y. Yu, “A survey of progress on cooperative multi-agent reinforcement learning in open environment,” *arXiv preprint arXiv:2312.01058*, 2023.
- [2] A. Wong, T. Bäck, A. V. Kononova, and A. Plaat, “Deep multiagent reinforcement learning: Challenges and directions,” *Artificial Intelligence Review*, vol. 56, no. 6, pp. 5023–5056, 2023.
- [3] A. Oroojlooy and D. Hajinezhad, “A review of cooperative multi-agent deep reinforcement learning,” *Applied Intelligence*, vol. 53, no. 11, pp. 13677–13722, 2023.
- [4] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, *et al.*, “The rise and potential of large language model based agents: A survey,” *Science China Information Sciences*, vol. 68, no. 2, p. 121101, 2025.
- [5] C. Sun, S. Huang, and D. Pompili, “Llm-based multi-agent reinforcement learning: Current and future directions,” *arXiv preprint arXiv:2405.11106*, 2024.
- [6] C. Zhu, M. Dastani, and S. Wang, “A survey of multi-agent deep reinforcement learning with communication,” *Autonomous Agents and Multi-Agent Systems*, vol. 38, no. 1, p. 4, 2024.
- [7] S. Iqbal and F. Sha, “Actor-attention-critic for multi-agent reinforcement learning,” in *International conference on machine learning*, pp. 2961–2970, PMLR, 2019.
- [8] Y. Dong, X. Zhu, Z. Pan, L. Zhu, and Y. 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.
- [9] A. Fourney, G. Bansal, H. Mozannar, C. Tan, E. Salinas, Z. Erkang, F. Niedtner, G. Proebsting, G. Bassman, J. Gerrits, *et al.*, “Magentic-one: A generalist multi-agent system for solving complex tasks, 2024,” URL <https://arxiv.org/abs/2411.04468>.
- [10] Y. Dang, C. Qian, X. Luo, J. Fan, Z. Xie, R. Shi, W. Chen, C. Yang, X. Che, Y. Tian, *et al.*, “Multi-agent collaboration via evolving orchestration,” *Advances in neural information processing systems*, vol. 38, pp. 165025–165059, 2026.
- [11] J. Sabater and C. Sierra, “Review on computational trust and reputation models,” *Artificial intelligence review*, vol. 24, no. 1, pp. 33–60, 2005.
- [12] J. Fang, Y. Peng, X. Zhang, Y. Wang, X. Yi, G. Zhang, Y. Xu, B. Wu, S. Liu, Z. Li, *et al.*, “A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems,” *arXiv preprint arXiv:2508.07407*, 2025.

-
- [13] H.-a. Gao, J. Geng, W. Hua, M. Hu, X. Juan, H. Liu, S. Liu, J. Qiu, X. Qi, Y. Wu, *et al.*, “A survey of self-evolving agents: On path to artificial super intelligence,” *arXiv preprint arXiv:2507.21046*, 2025.
 - [14] M. T. Lică, O. Shirekar, B. Colle, and C. 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.
 - [15] M. Tomasello, A. C. Kruger, and H. H. Ratner, “Cultural learning,” *Behavioral and brain sciences*, vol. 16, no. 3, pp. 495–511, 1993.
 - [16] W. Hoppitt and K. N. Lala, “Social learning: an introduction to mechanisms, methods, and models,” in *Social learning*, Princeton University Press, 2013.
 - [17] R. Boyd, P. J. Richerson, and J. Henrich, “The cultural niche: Why social learning is essential for human adaptation,” *Proceedings of the National Academy of Sciences*, vol. 108, no. supplement_2, pp. 10918–10925, 2011.
 - [18] R. L. Kendal, N. J. Boogert, L. Rendell, K. N. Laland, M. Webster, and P. L. Jones, “Social learning strategies: Bridge-building between fields,” *Trends in cognitive sciences*, vol. 22, no. 7, pp. 651–665, 2018.
 - [19] K. K. Ndousse, D. Eck, S. Levine, and N. Jaques, “Emergent social learning via multi-agent reinforcement learning,” in *International conference on machine learning*, pp. 7991–8004, PMLR, 2021.
 - [20] D. Hillier, C. Tan, and J. Jiang, “Social learning through interactions with other agents: A survey,” *arXiv preprint arXiv:2407.21713*, 2024.
 - [21] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.
 - [22] Ł. Kuśmierz, T. Isomura, and T. Toyoizumi, “Learning with three factors: modulating hebbian plasticity with errors,” *Current opinion in neurobiology*, vol. 46, pp. 170–177, 2017.
 - [23] R. E. Brown, “Donald o. hebb and the organization of behavior: 17 years in the writing,” *Molecular Brain*, vol. 13, no. 1, p. 55, 2020.
 - [24] A. Lazari, P. Salvan, M. Cottaar, D. Papp, M. F. Rushworth, and H. Johansen-Berg, “Hebbian activity-dependent plasticity in white matter,” *Cell Reports*, vol. 39, no. 11, 2022.
 - [25] R. I. Dunbar, “The social brain hypothesis,” *Evolutionary Anthropology: Issues, News, and Reviews: Issues, News, and Reviews*, vol. 6, no. 5, pp. 178–190, 1998.
 - [26] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, “The complexity of decentralized control of markov decision processes,” *Mathematics of operations research*, vol. 27, no. 4, pp. 819–840, 2002.
 - [27] C. Amato and F. Oliehoek, “Scalable planning and learning for multiagent pomdps,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.
 - [28] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, “The surprising effectiveness of ppo in cooperative multi-agent games,” *Advances in neural information processing systems*, vol. 35, pp. 24611–24624, 2022.
 - [29] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J. Z. Leibo, and N. De Freitas, “Social influence as intrinsic motivation for multi-agent deep reinforcement learning,” in *International conference on machine learning*, pp. 3040–3049, PMLR, 2019.
 - [30] N. Tomašev, M. Franklin, and S. Osindero, “Intelligent ai delegation,” *arXiv preprint arXiv:2602.11865*, 2026.
 - [31] Z. Wang and M. Cai, “Reinforcement learning applied to single neuron,” *arXiv preprint arXiv:1505.04150*, 2015.

-
- [32] M. Chalk, G. Tkacik, and O. Marre, “Training and inferring neural network function with multi-agent reinforcement learning,” 2019.
 - [33] A. Bhargava, M. R. Rezaei, and M. Lankarany, “Gradient-free neural network training via synaptic-level reinforcement learning,” *AppliedMath*, vol. 2, no. 2, pp. 185–195, 2022.
 - [34] S. Ohsawa, K. Akuzawa, T. Matsushima, G. Bezerra, Y. Iwasawa, H. Kajino, S. Takenaka, and Y. Matsuo, “Neuron as an agent,” 2018.
 - [35] J. Ott, “Giving up control: Neurons as reinforcement learning agents,” *arXiv preprint arXiv:2003.11642*, 2020.
 - [36] J. Chen and C. 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.
 - [37] B. Kang and S. 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.
 - [38] Y. Mukeriiia, J. Treur, and S. Hendrikse, “A multi-adaptive network model for human hebbian learning, synchronization and social bonding based on adaptive homophily,” *Cognitive Systems Research*, vol. 84, p. 101187, 2024.
 - [39] J. W. Pedersen and S. 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.
 - [40] M. Sandler, M. Vladymyrov, A. Zhmoginov, N. Miller, T. Madams, A. Jackson, and B. A. Y. Arcas, “Meta-learning bidirectional update rules,” in *International Conference on Machine Learning*, pp. 9288–9300, PMLR, 2021.
 - [41] G. Wang, Y. Xie, Y. Jiang, A. Mandlkar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, “Voyager: An open-ended embodied agent with large language models,” *arXiv preprint arXiv:2305.16291*, 2023.
 - [42] Z. Li, Y. Xie, R. Shao, G. Chen, D. Jiang, and L. Nie, “Optimus-2: Multimodal minecraft agent with goal-observation-action conditioned policy,” in *Proceedings of the computer vision and pattern recognition conference*, pp. 9039–9049, 2025.
 - [43] M. Malagón, J. Ceberio, and J. A. Lozano, “Craftium: Bridging flexibility and efficiency for rich 3d single-and multi-agent environments,” in *Forty-second International Conference on Machine Learning*.
 - [44] E. T. Hall, “A system for the notation of proxemic behavior,” *American anthropologist*, vol. 65, no. 5, pp. 1003–1026, 1963.
 - [45] F. Christianos, L. Schäfer, and S. Albrecht, “Shared experience actor-critic for multi-agent reinforcement learning,” *Advances in neural information processing systems*, vol. 33, pp. 10707–10717, 2020.
 - [46] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, *et al.*, “Lora: Low-rank adaptation of large language models,” *Iclr*, vol. 1, no. 2, p. 3, 2022.
 - [47] C. Qian, E. C. Acikgoz, Q. He, H. Wang, X. Chen, D. Hakkani-Tür, G. Tur, and H. Ji, “Toolrl: Reward is all tool learning needs,” *arXiv preprint arXiv:2504.13958*, 2025.
 - [48] S. Zhao, Z. Xie, M. Liu, J. Huang, G. Pang, F. Chen, and A. Grover, “Self-distilled reasoner: On-policy self-distillation for large language models,” *arXiv preprint arXiv:2601.18734*, 2026.
 - [49] O. L. Georgeon, J. B. Marshall, and S. 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.

A ENVIRONMENT

This section explains the environment selection decision, as well as the need for a new environment, and details the structure of the new environment, as well as the reward-milestone system.

A.1 ENVIRONMENT SELECTION

The choice of environment is consequential because progress in RL is closely coupled to the affordances and limitations of the simulator used. Classic platforms such as the Arcade Learning Environment enabled breakthroughs like DQN on Atari games, while more recent work on embodied and open-ended agents relies on visually rich 3D worlds such as Minecraft-based MineDojo. These environments expose a fundamental trade-off: researchers must often choose between computationally efficient but overly simplistic settings (e.g. small 2D gridworlds) and substantially slower but richer 3D simulators. Many of the richest environments are also built on closed-source commercial games such as Minecraft, which limits how much their mechanics can be modified or extended.

A second limitation is the lack of flexibility. Commonly used benchmarks either restrict customisation or require specifying environments in complex domain-specific languages, and many only support 2D worlds. This makes it difficult to design targeted scenarios for analysing specific behaviours such as long-horizon cooperation, coalition formation, or persistent social structure.

Minecraft. Minecraft is a first-person 3D voxel world, which makes it a partially observable environment: an agent sees only the local view in front of it and must act under uncertainty about the global state, so its policy has to integrate information over time rather than react to a full state vector. This partial observability compounds with the other properties that make Minecraft a difficult setting: high-dimensional visual observations of a varied and dynamic world, a large action space, and rewards that are sparse and delayed. Together these turn exploration into the central obstacle: informative reward is rarely encountered by chance, and the agent must discover long, structured behaviours from a first-person stream that never reveals the full task at once.

Why Craftium. Craftium [43] is a fully open-source 3D voxel world built on the Luanti (previously called Minetest) game engine, supporting both single and multi-agent scenarios through standard Gymnasium and PettingZoo interfaces. It is explicitly designed for rich 3D navigation, building, and resource-collection tasks, remains computationally efficient compared to Minecraft-based frameworks, and is highly customisable via a Lua API.¹ Its main drawback is that it is a framework rather than a fixed benchmark: additional engineering effort is required to design, implement, and validate specific social scenarios.

From OpenWorld to Five Chambers. Our initial design used Craftium’s OpenWorld environment with three heterogeneous roles (Engineer, Hunter, Guardian) corresponding to the three skill tracks (tools, hunt, defend). Pilot runs revealed that at small population sizes and short horizons, agents rarely co-located in shared regions, so co-activity signals required by the Hebbian update rule did not fire often enough for bonds to form. We considered scaling up the population and episode length to address this, but the compute requirements quickly exceeded the computation available for this research. 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 Hebbian mechanism can be evaluated under conditions where co-activity is reliably present. This environment structure now introduces more opportunities for collaboration, reducing the need for a much longer horizon. Studying the mechanism at scale in genuinely open environments remains an open direction.

Why a sequential cooperative environment. Existing cooperative Minecraft benchmarks such as VillagerBench [8] evaluate collaboration across a suite of *separate* scenarios (construction, farm-to-table cooking, and an escape room) each a self-contained task that tests a different facet of teamwork and is solved in isolation. This is the right design for measuring coordination on any single task, but it is the wrong setting for our question. A persistent social structure can only be studied if it persists:

¹See [43] for benchmarks and examples of single and multi-agent tasks, open-world survival scenarios, and procedurally generated continual-learning curricula.

bonds must form, carry forward, and be reused across tasks rather than being reset at every scenario boundary. We therefore designed WIRE as a single sequential curriculum in which the chambers build on one another: the primitive skills acquired in Chamber 1 are the building blocks of the cooperative objectives in later chambers, and the bonds a pair forms while solving one chamber are carried into the next, where they shape how the team approaches a harder, more coordination-dependent task. Cooperation and the social graph that encodes it are thus cumulative quantities that compound over the episode, which is exactly the regime our Hebbian mechanism is meant to operate in and the property that a collection of independent tasks cannot expose.

A.2 ENVIRONMENT STRUCTURE

WIRE is structured as five sequential chambers connected by one-way transitions, 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. To progress through the chambers, agents must solve the current chamber’s puzzles; if they fail to do so within a time budget of 20% of the episode, they are teleported into the next chamber instead. Teleportation time-gates progression without granting the milestone rewards for completing the chamber.

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

Chamber 2 (cooperative resource acquisition). The agents face a set of anvils that, once broken, dispense and auto-equip diamond equipment (M8–M11). Each anvil is resistant to any single agent: it yields only when at least two agents dig it simultaneously, so acquiring the equipment requires the team to synchronise their actions in time rather than divide the work spatially. This simultaneity constraint is what makes the chamber cooperative by construction, no agent can make progress alone, and the reward is reachable only through coordinated joint action.

Chamber 3 (communication under partial observability). The N agents are placed into individual isolation cells (M12) and arranged in a single dependency cycle: each agent’s switch releases not its own door but the next agent’s, so that agent i controls the door of agent $(i \bmod N) + 1$ and the chain closes back on itself ($1 \rightarrow 2 \rightarrow \dots \rightarrow N \rightarrow 1$). No agent can free itself, and an agent cannot observe the cell its own switch governs; release therefore propagates only if every agent acts on a teammate’s behalf, and an agent learns that its own door has been opened only through the environment, not through any signal it can produce alone. Escaping the cycle thus depends on agents communicating their state and their intent across cells (M13–M14), and the chamber completes once all agents have regrouped in the communal room (M15). Because the dependency structure makes isolated action insufficient by construction, this is the chamber in which the communication channel is load-bearing.

Chamber 4 (team combat). A practice combat arena populated with three zombies (M16–M19). The chamber rewards clearing all hostiles (M18) and grants the team an additional bonus for finishing with every agent intact (M19). Combat here is deliberately non-lethal: the zombies cannot kill an agent, and an agent that would otherwise have died instead incurs a -10 penalty for entering a fatal situation. This turns Chamber 4 into a low-stakes rehearsal of the combat coordination that the boss fight in Chamber 5 demands under genuine threat, the team acquires the joint-engagement skills it will later need without paying the cost of losing members, so the competence built here carries forward into the terminal chamber.

Chamber 5 (cooperative boss fight). The terminal chamber is a single boss encounter against a powered zombie, with its health raised from the default 20 to 60, so it absorbs three times the damage of an ordinary mob (M20–M24). Two properties make this the chamber where cooperation is most consequential. First, the forgiving invincibility of the earlier chambers is lifted: damage is real, a lethal hit is a permanent death, and the episode ends when either the boss falls or the entire team is down. Second, the boss’s enlarged health pool cannot be depleted by a single agent within the chamber’s time budget, so the team must concentrate sustained damage on it while surviving a

now-lethal opponent (the diamond gear distributed after the Chamber 2 cooperation is what equips the team for this fight if the anvil was broken, a direct instance of an earlier chamber’s outcome being carried 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 dealt real damage to the boss to share the reward rather than merely surviving alongside the team.

A.3 MILESTONE STRUCTURE

The per-agent reward stream r_i^t described in subsection 4.4 is driven primarily by a milestone ladder embedded in the WIRE environment (Table 7). Milestones are fired by a Lua-side event system in the Luantu server, which writes to an event log polled by the Python training loop each step. When a milestone fires, its reward is credited to one or more *contributors*, the agents whose action triggered the event. Communication milestones are not included in the milestone and cooperation milestone completion metrics.

Track structure. Milestones are organised into six tracks. The first five correspond to the five chambers and scale in reward with the coordination demand of the chamber. The sixth track is the *communication track*, which provides a per-chamber reward when an agent sustains valid targeted communication in that chamber.

Co-completion and contributor credit. Several milestones (M8, M9, M15, M18, M19, M22–M24) are inherently 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) rather than per-agent. For these, the event log records all agents currently contributing to the condition as contributors, and the per-milestone reward is split equally across them at the metric layer; damage-share credit rules are applied to the combat milestones (M18, M21–M23) so that contribution shares track actual damage dealt rather than just presence. The per-agent reward delivered to the policy gradient at runtime, however, is the full milestone value, not the split share, the split is a post-hoc analytical construct.

Communication reward. The communication track is delivered through a separate mechanism. Each valid message earns a small base payment, capped at a per-agent per-episode total to prevent reward farming through chat spam; in addition, when an agent emits more than a threshold number of valid messages in a single chamber, that chamber’s communication milestone fires once. Validity rules (minimum length, no duplicates, rate-limited) and target-validity rules (no self-targeting, no broadcast, no missing target) gate both rewards: messages that fail validity earn nothing, and messages with malformed targets are rerouted but their sender does not receive the base reward.

A.4 AUXILIARY REWARDS AND PENALTIES

Beyond the milestone ladder of Table 7, the per-agent reward stream contains a small set of auxiliary terms (Table 8) that shape the signal received by both the RL layer and the Hebbian update.

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 instead costs -10 , fired at most once per agent per episode. In Chamber 5 death is real: a fatal hit removes the agent for the remainder of the episode and costs -50 . The pitch-futility penalty supports the camera clamps described in Appendix B: vertical camera movement is limited to two `LookDown` and one `LookUp` steps from level, and an action that attempts to exceed the clamp costs -1 , discouraging the sky and floor-staring loops observed in early runs. Communication payments reward sustained *valid* targeted messaging at two tiers: a small per-message payment with a per-episode cap to prevent chat-spam farming, and the once-per-chamber communication milestones of Table 7.

Table 7: **Milestone reward schedule for WIRE.** Per-chamber rewards scale with coordination demand. *Type* marks per-agent (P) versus team-level (T) milestones, the latter credited to every contributing agent; each fires at most once per agent per episode.

ID	Track	Trigger	Reward	Type
<i>Chamber 1 — solo skill acquisition</i>				
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
<i>Chamber 2 — cooperative resource acquisition</i>				
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
<i>Chamber 3 — communication under partial observability</i>				
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
<i>Chamber 4 — team combat</i>				
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
<i>Chamber 5 — cooperative boss fight</i>				
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
<i>Communication track</i>				
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 8: **Auxiliary rewards and penalties.** All values are per agent. The communication milestones themselves are listed in Table 7.

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	capped at 50 messages total per agent per episode

B AGENT ARCHITECTURE DETAILS

This appendix documents the MindForge stack [14] that each WIRE agent inherits.

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 [49]; *partner beliefs* maintain one entry per teammate summarising their chamber, gear, commitments, and reported schemes; *interaction beliefs* extract shared actionables (role assignments, switch and door status, combat coordination, team-level warnings) from recent conversation; *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 failure signal; and *skills* stores 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 that contain no actionable keyword, replacing them with a chamber-appropriate fallback. A critic module evaluates task success every twenty steps from the frame, last action, reward, status, inventory, and current task. The verdict drives three downstream effects: successful tasks promote their action into the skill store, repeated failures trigger task reassignment, and every verdict is appended to episodic memory.

Action selection. The action-selection prompt is assembled from the frame, current task and context, recent beliefs, retrieved skills and episode summary, live chamber state, inventory, position, status, and a hard-grounded list of chambers the agent has actually visited this episode, the last item rules out a hallucination mode in which the model claims to be in a chamber it has never entered. The prompt returns a JSON object with action, rationale, message, and target.

Action space. The action space consists of the twenty-two discrete primitives listed in Table 9: 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 leave eight near-duplicate low-value actions inflating the policy’s action entropy. The VLM-only conditions retain the full set.

Action execution semantics. Three environment-side conventions shape how a primitive translates into world ticks. (i) `DIG` is *sustained*: the key is held for ten environment steps (thirty physics ticks at frame-skip 3), so a single action suffices to break a block, and `JUMP` expands to a jump-then-forward sequence so that single jumps clear ledges. (ii) Camera pitch is clamped (at most two `LOOKDOWN` and one `LOOKUP` steps from level); a pitch-capped action is redirected to `NOOP` and charged a small futility penalty, preventing the policy from staring into the sky or floor. (iii) Malformed action strings emitted by the VLM are recovered where the intent is unambiguous through a synonym map (e.g. `Attack`, `Mine` → `DIG`; `Turn` → `TurnRight`) and otherwise clamped to `NOOP`; an idle guard converts repeated `NOOP`s into `MoveForward` so an agent cannot silently stall. Both recovery and idle-forcing counts are logged per run as action-health diagnostics.

Table 9: **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-generation policy of subsection 4.4.

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

C CENTRALISED ORCHESTRATION BASELINE

To compare decentralised Hebbian coordination with an explicit centralised alternative, we implement a non-embodied coordinator inspired by VillagerAgent [8]. The coordinator replaces the Hebbian Social Graph entirely: no bonds are learned or exposed to the agents, and none of the Hebbian couplings are active. Instead, a single coordinator maintains a global view of team progress and assigns objectives to the individual agents.

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. This is information no individual agent has, though 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, represented as a dependency graph so that an objective 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 $T_{\text{soc}}=8$ steps, matching the deliberation frequency of the Hebbian social module.

Assignments are binding only at the objective level. The coordinator determines *what* an agent should work towards, while the agent retains control over *how* to achieve it: its task-generation prompt is constrained to propose tasks that advance the assigned objective rather than unrelated goals, but its reasoning, primitive actions and messages remain its own.

Progress verification and replanning. Each subtask is associated with one or more observable 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 self-report success. A subtask fails if an assigned agent dies, if no associated milestone fires within a fixed step budget, or if the team is teleported past the chamber its milestones belong to. Failed subtasks are retired rather than retried: the coordinator sees the failure and proposes a different route to the same goal. The graph is rebuilt at the start of each episode, so the coordinator carries no memory across episodes.

Communication. The coordinator does not determine who communicates with whom and does not route messages between agents. Agents retain the same communication mechanism as in the corresponding Gemma-4-E4B baseline. Centralization therefore operates through task allocation rather than through explicit control of the communication topology, following the VillagerAgent design.

Algorithm 1 Centralised orchestration baseline

- 1: **Input:** environment state s^t ; milestone status $\mathcal{M}^t$; agent states $\{x_i^t\}_{i=1}^N$; subtask graph $\mathcal{G}$ with assignments $\mathcal{U}^t$
 - 2: {Bookkeeping, every step}
 - 3: complete a subtask when any of its milestones fires; fail it if an assigned agent died or no milestone fired within $T_{\text{fail}}=60$ steps of assignment
 - 4: propagate failure to unstarted dependents; free the agents of finished subtasks (failed subtasks are retired, not retried)
 - 5: $\mathcal{E}^t \leftarrow$ unstarted subtasks whose dependencies have succeeded; $\mathcal{F}^t \leftarrow$ alive, unassigned agents
 - 6: **if** $\geq T_{\text{soc}}$ steps since the last decomposition **and** the plan needs extending ($\mathcal{G}$ exhausted, a subtask just finished, or $\mathcal{E}^t=\emptyset$ while $\mathcal{F}^t\neq\emptyset$) **then**
 - 7: 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
 - 8: add the valid proposals to $\mathcal{G}$ and recompute $\mathcal{E}^t$
 - 9: **end if**
 - 10: **if** $\mathcal{E}^t \neq \emptyset$ **and** $\mathcal{F}^t \neq \emptyset$ **then**
 - 11: coordinator assigns executable subtasks to available agents with a per-agent role, multi-agent subtasks only if fully crewed; active assignments are retained
 - 12: $\mathcal{U}^{t+1} \leftarrow \mathcal{U}^t \cup \text{accepted assignments}$
 - 13: **end if**
 - 14: constrain each assigned agent's auto-curriculum to its assigned objective
 - 15: **Return** $\mathcal{U}^{t+1}$, rendered as directives in the agents' action prompts; each agent independently chooses its reasoning, primitive actions, and messages
-

D PSEUDO-CODE

Algorithm 2 Hebbian social plasticity step at env. step t . The growth and decay branches are mutually exclusive by construction: the co-activity window includes the current step, so $\text{coop}_{ij} < \epsilon$ implies $c_{ij}(t) = 0$ and hence $G_{ij} = 0$. The always-on homeostatic term $-\lambda W_{ij}$ gives the rule a stable interior fixed point $W_{ij}^* = \kappa_i c_{ij} / (\kappa_i c_{ij} + \lambda)$ rather than saturating at the clip.

```

1: State: bonds  $W \in [0, 1]^{N \times N}$  (init.  $W_0$ , zero diagonal); co-activity and reward windows  $\mathcal{C}, \mathcal{R}$  of
   length  $n$ ; running max  $r_{\max}$ 
2: Input: positions  $\{\mathbf{p}_i\}$ ; message pairs  $\mathcal{M}_t$ ; bondable rewards  $\{r_i^{\text{bond}}\}$  (death-excluded, gated to
   cooperative chambers); total rewards  $\{r_i\}$ 
3:  $\triangleright$  Co-firing signal (Eqs. 2–5)
4:  $r_{\max} \leftarrow \max(r_{\max}, \max_i |r_i^{\text{bond}}|)$ 
5: for each agent  $i$  do
6:    $g_i \leftarrow \text{clip}(\alpha |r_i^{\text{bond}}| / r_{\max} + (1 - \alpha) \mathbb{1}[i \in \mathcal{A}], 0, 1)$ 
7:    $\kappa_i \leftarrow \eta_0 + \eta_+ |r_i^{\text{bond}}| / R$  growth coefficient
8: end for
9: for each ordered pair  $(i, j), i \neq j$  do
10:    $s_{ij} \leftarrow \mathbb{1}[\|\mathbf{p}_i - \mathbf{p}_j\| \leq d]$  spatial gate
11:    $c_{ij} \leftarrow \text{clip}(s_{ij} g_i g_j + \sum_{k \in \mathcal{K}} \delta_k \mathbb{1}[\text{event}_{ij}^{(k)}], 0, 1)$ 
12:   if  $c_{ij} < \epsilon$  then  $c_{ij} \leftarrow 0$ 
13: end for
14:  $\triangleright$  Windowed statistics
15: append  $c(t)$  to  $\mathcal{C}$  and  $\{r_i\}$  to  $\mathcal{R}$ 
16:  $\text{coop}_{ij} \leftarrow \max_{s \in \mathcal{C}} c_{ij}(s)$ ;  $\text{neg}_i \leftarrow \mathbb{1}[\sum_{s \in \mathcal{R}} r_i(s) < -\theta]$ 
17:  $\triangleright$  Bond update (Eqs. 7–8)
18: for each ordered pair  $(i, j), i \neq j$  do
19:    $G_{ij} \leftarrow \kappa_i c_{ij} (1 - W_{ij})$  reward-modulated growth
20:    $D_{ij} \leftarrow \eta_- \mathbb{1}[\text{coop}_{ij} < \epsilon \wedge \text{neg}_i] W_{ij}$  failure-gated decay
21:    $W_{ij} \leftarrow \text{clip}(W_{ij} + G_{ij} - D_{ij} - \lambda W_{ij}, 0, 1)$ 
22: end for
23: Return  $W, c(t)$ 

```

Algorithm 3 Social module: bond-conditioned deliberation for agent i (inference-time coupling of W , subsection 4.3). 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. In *prompt* mode σ_i reaches the agent only through the action prompt; in *bias* mode its help-request target additionally overrides message routing (Alg. 4).

```

1: 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)
2: if fewer than  $T_{\text{soc}}$  steps have passed since the last deliberation and a cached  $\sigma_i$  exists then
3:   return cached  $\sigma_i$  (directive unchanged between deliberations)
4: end if
5: render each bond as  $(W_{ij}, \Delta W_{ij})$ , tagged rising / decaying / stable
6:  $\sigma_i \leftarrow$  VLM deliberation over the bond table,  $\mathcal{I}_i^t$ , and recent state, choosing one of:
7:   request help: target the highest-bonded teammate plausibly able to help; draft the request
   message
8:   offer help: respond to inbox senders with high  $W_{ij}$  and rising  $\Delta W_{ij}$ ; ignore low-bond senders
   ( $W_{ij} < w_{\min}$ )
9:   stay on task: no request, no response
10:  $\sigma_i$  also records the bonds it referenced and a short rationale (audit trail for bond-driven decisions)
11: Return  $\sigma_i$ , rendered as a directive block for the action prompt

```

Algorithm 4 WIRE training loop with Hebbian social plasticity. The graph W and long-term memory (skill and episode stores) persist across episodes; only the environment and per-episode working memory are reset. With social replay disabled ($\rho=0$) the shaded steps are no-ops. They are active ($\rho=0.3$) in the RLFT conditions and inactive in the zero-shot VLM conditions.

```

1: Init: frozen base LM with per-agent LoRA policies  $\pi_{\theta_i}$ ; shared critic  $V_\psi$ ; social graph  $W \leftarrow W_0(\mathbf{1}\mathbf{1}^\top - I)$ ; replay mixing strength  $\rho$ 
2: for each episode do
3:   reset environment and per-episode working memory (task, beliefs); skills, episodic memory, and  $W$  persist
4:   for  $t = 0, 1, 2, \dots$  until termination do
5:     encode the joint state  $s^t$  once for the critic
6:     for each living agent  $i$ , concurrently do
7:        $o_i^t \leftarrow$  (visual frame, structured context, inbox  $\mathcal{I}_i^t$ )
8:        $\sigma_i \leftarrow \text{SOCIALDELIBERATE}(\{W_{ij}\}_{j \neq i}, \{\Delta W_{ij}\}_{j \neq i}, \mathcal{I}_i^t)$  (Alg. 3)
9:       generate reasoning  $th_i^t$  and message  $(m_i^t, \tau_i^t)$  from  $o_i^t$  and the directive  $\sigma_i$ 
10:       $a_i^t \sim \pi_{\theta_i}(\cdot \mid o_i^t, th_i^t)$  by constrained decoding (Eq. 12)
11:    end for
12:    deliver each message  $m_i^t$  to its target  $\tau_i^t$  (bias mode:  $\tau_i^t \leftarrow \sigma_i$ 's help-request target when set)
13:    advance the environment under the joint action; collect per-agent rewards  $r_i^t$  and message pairs  $\mathcal{M}_t$ 
14:     $W, c(t) \leftarrow \text{HEBBIANSTEP}(\{\mathbf{p}_i\}, \mathcal{M}_t, \{r_i^{\text{bond}}\}, \{r_i^t\})$  (Alg. 2)
15:     $r_i'^t \leftarrow (1 - \gamma_d) r_i^t + \gamma_d \sum_{j \neq i} \bar{W}_{ij} c_{ij} r_j^t$  (Eq. 9)
16:    write  $(o_i^t, a_i^t, r_i'^t, \log \pi_{\theta_i}(a_i^t \mid o_i^t, th_i^t))$  to agent  $i$ 's buffer  $\mathcal{D}_i$ ; write  $(s^t, \bar{r}_t^t)$  to the critic buffer
17:    if agent  $i$ 's buffer holds  $K$  transitions then
18:      if  $\rho > 0$  and no snapshot is active then snapshot all buffers  $\{\mathcal{D}_j\}_j$  — one shared pre-update view per update cycle
19:       $\mathcal{S}_i \leftarrow$  draw  $m = \lceil K\rho/(1 - \rho) \rceil$  snapshot transitions from neighbours  $j \neq i$  with  $W_{ij} \geq 0.05$ , in proportion to  $\bar{W}_{ij}$ ; recompute GAE on copies under the source agent's baseline (Eq. 10)
20:      PPO update of adapter  $\theta_i$  over  $\mathcal{D}_i \cup \mathcal{S}_i$  with GAE advantages (Eq. 14); for shared transitions the clipped ratio  $\pi_{\theta_i}/\pi_{\theta_j}$  is the importance weight
21:      clear  $\mathcal{D}_i$ ; retire the snapshot once every living agent has updated
22:    end if
23:    if the critic buffer is full then
24:      update  $V_\psi$  on the team-mean diffused return (Eq. 15)
25:    end if
26:  end for
27: end for

```

E HYPERPARAMETERS

E.1 REINFORCEMENT-LEARNING HYPERPARAMETERS

Table 10 lists the settings used for the MAPPO layer described in subsection 4.4. Reward normalisation is applied only in the IPPO ablation: under the centralised critic both the per-agent and team rewards are kept on the raw milestone scale, so that the GAE deltas $r'_{i,t} + \gamma V_\psi(s^{t+1}) - V_\psi(s^t)$ are computed on a single consistent scale.

Table 10: Reinforcement-learning hyperparameters for the WIRE MAPPO layer.

Hyperparameter	Value
<i>Base model & adapters</i>	
Base VLM	Qwen3.5-2B (frozen)
LoRA rank r / α / dropout	8 / 16 / 0.05
LoRA target modules	q_proj, v_proj
Precision / grad. checkpointing	fp16 / on
Prompt length cap	256 tokens
<i>PPO / MAPPO</i>	
Discount γ	0.995
GAE λ_{GAE}	0.95
Clip ϵ	0.2
Entropy coef. β_t	0.05 $\rightarrow$ 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	128 steps
Rollout buffer size	2048
<i>Optimisation</i>	
Optimiser	Adam
Actor learning rate	1×10^{-4}
Critic learning rate	3×10^{-4}
Critic hidden width	256

E.2 HEBBIAN SOCIAL-PLASTICITY HYPERPARAMETERS

Table 11 lists the hyperparameters of the Hebbian social module (subsection 4.2). Unlike the RL settings above, none of these are learned: the graph $\mathbf{W}$ carries no gradient, and every value below is a fixed constant of the local update rule. The symbols match the co-activity signal (Eqs. (2)–(5)), the failure gate (Eq. (6)), the three-term bond update (Eq. (7)), its Heb2.0 counterpart (Eqs. (16)–(17)), reward diffusion (Eq. (9)), and weight-gated experience sharing (Eq. (10)). The reward scale r_{max} is not a tuned constant but a running maximum of observed absolute rewards, kept here for completeness.

Main rule and the Heb2.0 variant. The table reports two settings. The *main rule* is Eq. (7) as stated in subsection 4.2, and it produced every result in the paper except one: all +Heb conditions in Table 2, the bond–behaviour correlations of Table 3, every co-firing arm of Table 4, and both RQ3 experiments. *Heb2.0* is the longer-timescale variant described in subsection F.1 and reported as row (g') of Table 2. It differs in three ways: reward acts on the eligibility trace e_{ij} rather than on the current step's co-activity, so the salience term moves out of the growth coefficient; the co-failure gate ϕ_{ij} is replaced by a signed failure term that routes the death and would-die penalties through that same trace at rate η_-^d , with the penalty magnitude capped at r_{cap} before normalisation; and spatial co-activity gains a floor c_0 , so a co-located pair registers co-activity even when neither agent is otherwise engaged. Because reward is sparse the trace term cannot saturate the weights, which is what allows η_0 and λ to be lowered so that earned credit persists. The trace decay is written ρ_e

Table 11: Hebbian social-plasticity hyperparameters for the WIRE social module. Symbols match subsection 4.2. All values are fixed constants of the local update rule, not learned. *Main rule* is Eq. (7), used for every reported condition except row (g') of Table 2; *Heb2.0* is the longer-timescale variant of subsection F.1. “–” marks a constant that does not exist in that rule.

Hyperparameter	Symbol	Main rule	Heb2.0
<i>Co-activity signal (Equation 2–Equation 5)</i>			
Interaction radius	d	5.0	5.0
Engagement reward weight	α	0.5	0.5
Social-cue credit (every cue k)	δ_k	0.5	0.5
Co-activity floor (co-location)	c_0	–	0.25
Co-activity threshold	ϵ	0.05	0.05
Reward scale	$r_{\max}$	running max	running max
<i>Eligibility trace (Equation 16)</i>			
Trace decay	ρ_e	–	0.9
<i>Bond update (Equation 6–Equation 7, Equation 17)</i>			
Initial bond (warm start)	W_0	0.1	0.1
Association-floor growth	η_0	0.005	0.001
Reward-modulated growth rate	η_+	0.05	0.05
Salience reward normaliser	R	50	50
Failure-gated decay rate	η_-	0.025	–
Co-activity / loss window	n	50 steps	–
Loss threshold for neg_i	θ	5.0	–
Signed failure decay rate	η_-^d	–	0.05
Failure-signal cap	r_{cap}	–	10
Homeostatic decay rate	λ	0.005	0.001
<i>Graph couplings (Equation 9, Equation 10)</i>			
Diffusion strength	γ_d	0.2	–
Experience-sharing mixture	ρ	0.3	0
Social deliberation interval	T_{soc}	8 steps	8 steps

throughout, to keep it distinct from the experience-sharing mixture ρ of Eq. (10). The signed failure term was added after the first Heb2.0 arm, so runs launched before it are equivalent to $\eta_-^d = 0$.

Per-experiment exceptions. The frozen-topology ablation (Table 6) holds $\mathbf{W}$ fixed at its imposed values and disables reward diffusion ($\gamma_d = 0$), so the graph is read by the social module but never written and never shapes the reward. Weight-gated experience sharing applies only to the RLFT conditions: the IPPO+Heb and MAPPO+Heb rows of Table 2 set $\rho = 0.3$, where neighbours with $W_{ij} < 0.05$ contribute no samples. It is inactive ($\rho = 0$) in the zero-shot VLM conditions, which perform no policy update for shared transitions to enter. The transplant experiment overrides the initial graph with the equalised Phase A bonds ($W_{ij} = 0.265$ within pairs) rather than the warm start W_0 . The deliberation interval T_{soc} is 8 steps everywhere except the sweep of subsection 5.2, which varies it over $\{2, 8, 20, 50, 100\}$.

F ADDITIONAL RESULTS

Table 12: **Steps to first completion of every milestone.** VLM conditions only. Each cell is the median environment step at which the milestone was first completed, over the episodes that completed it; the subscript is the number of completing episodes. “-” means never completed. Lower step and larger subscript are better. Chamber-entry milestones (M12, M16, M20) are only credited when the team advances on its own before the chamber timer, never for rescue teleports. Every condition pools 6 seeds $\times$ 3 episodes (18) except Gemma-E4B+Heb2.0, which pools 3 seeds (9), so its subscripts are bounded by 9 and are not directly comparable in magnitude with the others.

ID	Milestone	Qwen3.5-2B	Qwen3.5-9B	Qwen3.5-2B+Heb	Qwen3.5-9B+Heb	Gemma-E4B	Gemma-E4B+Central Orch.	Gemma-E4B+Heb	Gemma-E4B+Heb2.0
<i>Chamber 1 — solo skill acquisition</i>									
M1	Move >5 from spawn	10 ₁₈	6 ₁₈	7 ₁₅	5 ₁₈	1 ₁₈	2 ₁₈	1 ₁₈	4 ₉
M2	Dig 3 any	14 ₁₈	8 ₁₈	11 ₁₇	8 ₁₈	45 ₁₃	24 ₁₇	56 ₁₄	63 ₈
M3	Pick up 3 items	23 ₁₇	36 ₁₇	23 ₁₇	20 ₁₆	86 ₄	68 ₉	125 ₈	14 ₃
M4	Dig 5 wood	115 ₁	143 ₈	90 ₂	150 ₃	-	-	182 ₁	-
M5	Kill 1 animal	-	-	-	-	-	-	-	-
M6	Kill 2 animals	-	-	-	-	-	-	-	-
M7	Dig 3 stone	84 ₃	35 ₇	183 ₁	34 ₇	-	-	30 ₁	-
M.door1	Door 1 unlocked	13 ₁₇	8 ₁₇	11 ₁₃	8 ₁₆	45 ₁₃	24 ₁₇	51 ₁₃	63 ₈
<i>Chamber 2 — cooperative resource acquisition</i>									
M8	First anvil break	-	201 ₂	-	301 ₁	-	-	-	-
M9	Second anvil break	-	218 ₂	-	-	199 ₁	-	-	-
M10	Sword equipped	-	201 ₂	-	301 ₁	-	-	-	-
M11	Chestplate equipped	-	218 ₂	-	-	199 ₁	-	-	-
<i>Chamber 3 — communication under partial obs.</i>									
M12	Enter isolation cell	-	-	-	-	-	-	-	-
M13	Press switch	434 ₁₄	415 ₁₈	421 ₁₀	403 ₁₇	406 ₇	407 ₁₁	407 ₁₂	440 ₇
M14	Door opened	434 ₁₄	415 ₁₈	421 ₁₀	403 ₁₇	406 ₇	407 ₁₁	407 ₁₂	440 ₇
M15	All in communal room	-	-	-	-	-	-	-	-
<i>Chamber 4 — team combat</i>									
M16	Enter Chamber 4	-	-	-	-	-	-	-	-
M17	First mob kill	-	679 ₁₀	756 ₁	696 ₇	670 ₁₀	692 ₁₂	662 ₁₃	686 ₈
M18	All mobs killed	-	-	-	-	-	-	-	-
M19	All survive (damage)	-	-	-	-	-	-	-	-
<i>Chamber 5 — cooperative boss fight</i>									
M20	Enter Chamber 5	-	-	-	-	-	-	-	-
M21	First boss damage	-	-	-	-	-	-	-	-
M22	Boss at 50% HP	-	-	-	-	-	-	-	-
M23	Boss defeated	-	-	-	-	-	-	-	-
M24	All alive at kill	-	-	-	-	-	-	-	-
<i>Communication track</i>									
M.comm.ch1	Valid messaging, Ch1	5 ₁₈	5 ₁₈	5 ₁₈	5 ₁₈	5 ₁₈	5 ₁₈	5 ₁₈	6 ₉
M.comm.ch2	Valid messaging, Ch2	204 ₁₈	204 ₁₈	204 ₁₈	204 ₁₈	138 ₁₈	128 ₁₈	88 ₁₈	122 ₉
M.comm.ch3	Valid messaging, Ch3	604 ₁₈	502 ₁₈	604 ₁₈	560 ₁₈	604 ₁₈	604 ₁₈	542 ₁₈	570 ₉
M.comm.ch4	Valid messaging, Ch4	612 ₁₈	613 ₁₈	612 ₁₈	614 ₁₈	612 ₁₈	608 ₁₈	611 ₁₈	612 ₉
M.comm.ch5	Valid messaging, Ch5	850 ₁	805 ₄	810 ₃	806 ₂	809 ₁₀	813 ₇	811 ₁₀	813 ₇

F.1 ATTRIBUTING BOND GROWTH TO THE UPDATE RULE

To examine how the individual terms of Eq. 7 shape the learned graph, we replay Algorithm 2 offline using the logged per-step agent positions, social events, and reward streams. The same parameters as in Table 11 are used, with bond state carried continuously across episodes.

The replay decomposes the update around an example joint milestone in Chamber 3. In the main rule, reward modulates the growth term through the reward available at the current step. Because task rewards in WIRE are sparse, this additional modulation appears at isolated rewarded events, whereas the co-activity floor η_0 and homeostatic term $-\lambda W_{ij}$ act throughout the trajectory. Consequently, sustained co-activity has substantially more opportunity to determine the evolution of the bond than individual milestone rewards. Around the illustrated joint completion, reward briefly increases bond growth; once the agents cease co-acting, the subsequent trajectory is again determined primarily by the balance between co-activity-driven growth and homeostatic decay. The co-failure gate ϕ_{ij} (Eq. 6) does not activate in this example.

This decomposition motivates *Heb2.0*, the longer-timescale variant reported as row (g') of Table 2. Each pair keeps an eligibility trace e_{ij} that accumulates co-activity while the two agents interact and

fades geometrically when they do not (Eq. 16; with $\rho_e = 0.9$, a memory of roughly ten steps), and reward acts on this accumulated trace rather than on the current step’s co-activity alone:

$$e_{ij}(t) = \rho_e e_{ij}(t-1) + c_{ij}(t), \quad (16)$$

$$\begin{aligned} \Delta W_{ij}(t) = & \underbrace{\eta_0 c_{ij}(t) (1 - W_{ij}(t))}_{\text{association floor}} + \underbrace{\eta_+ \frac{|r_i^{\text{bond}}(t)|}{R} e_{ij}(t) (1 - W_{ij}(t))}_{\text{reward-gated consolidation}} \\ & - \underbrace{\eta_-^d \frac{\tilde{r}_i^{\text{death}}(t)}{R} e_{ij}(t) W_{ij}(t)}_{\text{signed failure decay}} - \underbrace{\lambda W_{ij}(t)}_{\text{homeostatic decay}}. \end{aligned} \quad (17)$$

A milestone credited at step t therefore strengthens the bond in proportion to how much the pair worked together over the preceding steps, whether or not they happen to be co-active at t itself, and a pair with no recent joint activity receives no credit. Failure is routed through the same trace, so a setback weakens the bonds to the partners the agent has recently been working with: $\tilde{r}_i^{\text{death}}(t) = \min(|r_i^{\text{death}}(t)|, r_{\text{cap}})$ is agent i ’s death and would-die penalty at step t , capped at r_{cap} so that a near-miss and a fatal outcome assign comparable blame and no single event erases a strong bond. This term replaces the co-failure gate ϕ_{ij} of the main rule, which requires a pair to be simultaneously disengaged and failing — conditions that rarely co-occur here, since failures happen amid heavy messaging. Heb2.0 additionally floors spatial co-firing, $c_{ij}^{\text{spat}}(t) = \mathbb{1}[\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\| \leq d] \max(g_i(t)g_j(t), c_0)$, so that a co-located pair registers co-activity even when neither agent is otherwise engaged. Because reward is sparse, the trace term cannot by itself saturate the weights, which is what allows the homeostatic decay λ to be reduced in Figure 10 so that credit persists rather than being erased within a few hundred steps.

Figure 8–10 replay the trace on the same logged trajectories as an open-loop counterfactual: agent actions, messages, and rewards remain those generated under the main rule, so these panels illustrate bond dynamics rather than task-performance effects. The signed failure term is inactive in them ($\eta_-^d = 0$), as it is in the closed-loop arm of row (g’), which predates its introduction (subsection E.2); that arm therefore isolates the contribution of the trace and the weaker homeostatic decay.

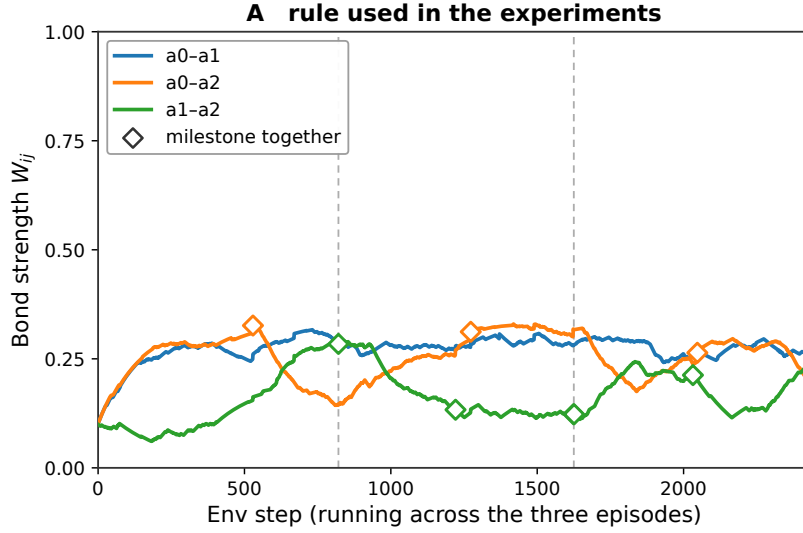


Figure 8: **Open-loop comparison of longer-timescale reward modulation: (A) the update rule used in the experiments.** The logged trajectory (seed 456) is replayed under the rule as run. Diamonds mark joint milestone completions. Because behaviour and rewards are fixed to those generated by the original experiment, these curves illustrate the resulting bond dynamics rather than task-performance effects.

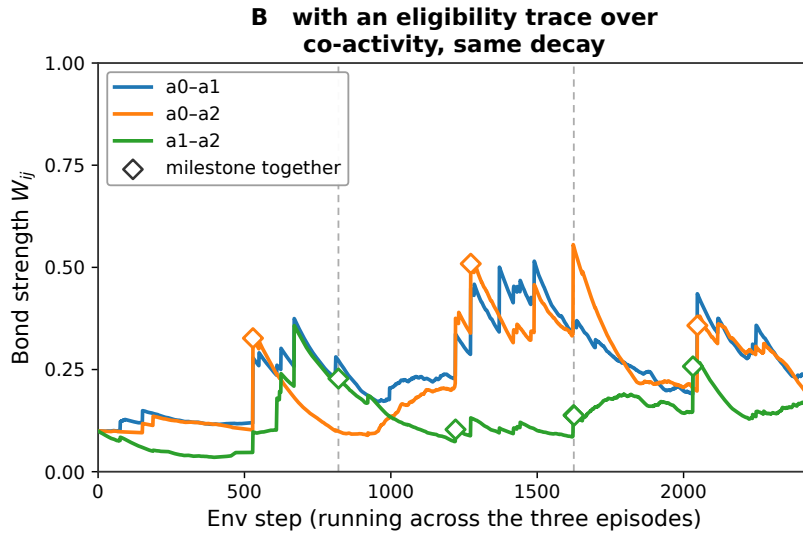


Figure 9: **Open-loop comparison of longer-timescale reward modulation: (B) with an eligibility trace over co-activity, same decay.** Same replay as Figure 8, with the trace-based update rule described above and the homeostatic decay unchanged from the rule used in the experiments.

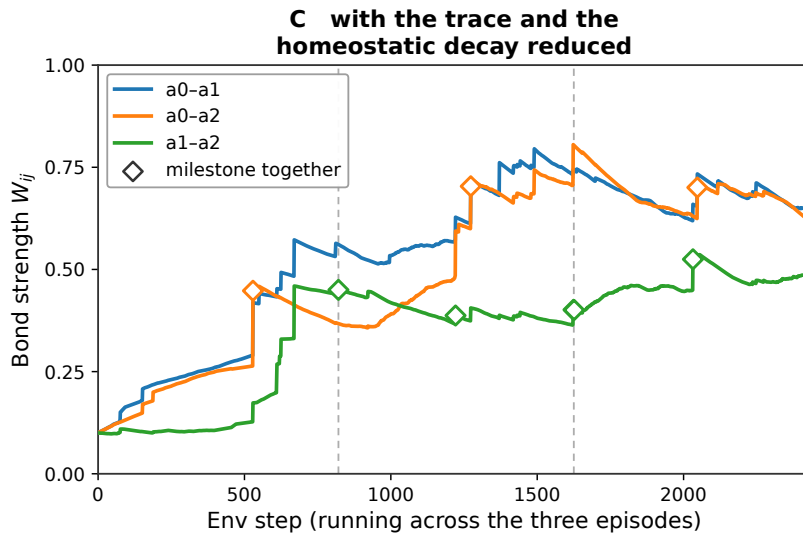


Figure 10: **Open-loop comparison of longer-timescale reward modulation: (C) same trace, with the homeostatic decay reduced.** Same replay as Figure 8; reducing λ lets the reward-gated credit from the trace persist instead of being erased within a few hundred steps.

G QUALITATIVE ANALYSIS OF INTERACTION

This appendix complements the aggregate results with interaction-level examples showing how learned relational structure manifests in communication and embodied behaviour.

G.1 COORDINATION WITH AND WITHOUT LEARNED BONDS (RQ1)

Relational structure captures interaction, not execution. Figure 11 provides a run-level view of the distinction observed quantitatively in RQ1. Pairwise bonds differentiate over time while communication remains distributed across the population, and the same run contains both successful and unsuccessful cooperative interactions. In the successful anvil case, action-directed messages are followed by a joint act; in the failure case, similarly explicit coordination persists despite an incorrect interpretation of the environment state. The combat example further shows that interaction need not collapse to a single dyad. Together, these examples illustrate why bond strength tracks communication more consistently than physical cooperation: the graph records *who interacts with whom*, but the underlying agent must still ground and execute the collaboration correctly.

Example 1 — Negotiated simultaneous action		Qwen3.5-9B, seed 123, ep. 1, Ch2, $t=199-234$
$t=201$	a2 → a1 [Dig]	“I am Digging the anvil in front of me, please Dig the other anvil!”
	...	(25 steps of mutual alignment: turning, positioning, and re-stating the pact)
$t=226$	a1 → a0 [Dig]	“agent_0, I am digging the anvil at the center. Please join me to break it together.”
$t=227$	a1 → a0 [Dig]	“Keep punching the anvil! It needs 2 agents to break.”
$t=228$	a0 → a1 [Dig]	“Keep punching the anvil! We need 2 agents to break it.”
$t=229$	a2 → a1 [Dig]	“I am Digging the back anvil now. Please Dig simultaneously with me to break it.”
⇒ all three agents dig on the same step; M9_ANVIL.B1 (+40) fires.		

The next two examples isolate what a learned bond does and does not change. The within-pair weights ($W=0.269$ vs. $W=0.280$) at the moment of the request are the same for both examples, yet the outcomes are opposite. This is the interaction-level face of the correlation pattern in Table 3: bonds track who talks to whom, but bond magnitude does not guarantee that a request converts into action.

Example 2 — A request fulfilled under a learned bond		Qwen3.5-9B+Heb, seed 1011, ep. 2, Ch3, $t=401-431$, $W=0.269$
$t=401$	a0 → a1 [Dig]	“I am pressing my switch now. Please move to your cell and press your switch next.”
$t=404$	a2 → a1 [Dig]	“I am pressing the switch now to open your door.”
$t=407$	a1 → a2 [MoveForward]	“I see you near the switch. Please press it to open my door.”
$t=409$	a2 → a0 [Dig]	“I am pressing the switch to open Door 3. Please move towards the door once it opens.”
$t=410$	a0 → a1 [Dig]	“I am pressing the switch in my cell to open your door, agent_1.”
⇒ at $t=411$ M13_SWITCH_PRESSED (+40) and M14_DOOR_OPENED (+60) fire.		
$t=423$	a1 → a0 [TurnRight]	“Agent_0, your switch press worked! I see the door opening.”

Example 3 — The same-size bond, ignored		Qwen3.5-2B+Heb, seed 456, ep. 3, Ch3, $t=557-587$, $W=0.280$
$t=557$	a1 → a2 [Press]	“Please press the switch to enter Ch4.”
$t=562$	a2 → a0 [LookDown]	“Switch is centered, please press to enter Ch4.”
$t=564$	a1 → a2 [TurnLeft]	“Switch is centered, please press to enter Ch4.”
	...	(the same request circulates for 17 consecutive steps; every agent forwards it onward and no press registers)
$t=583$	a0 → a2 [PressSwitch]	“Great job on entering Ch4! What’s the next instruction or task you need help with?”
⇒ no milestone fires; the team is still in Chamber 3, and agent 0’s congratulation hallucinates a success that never happened.		

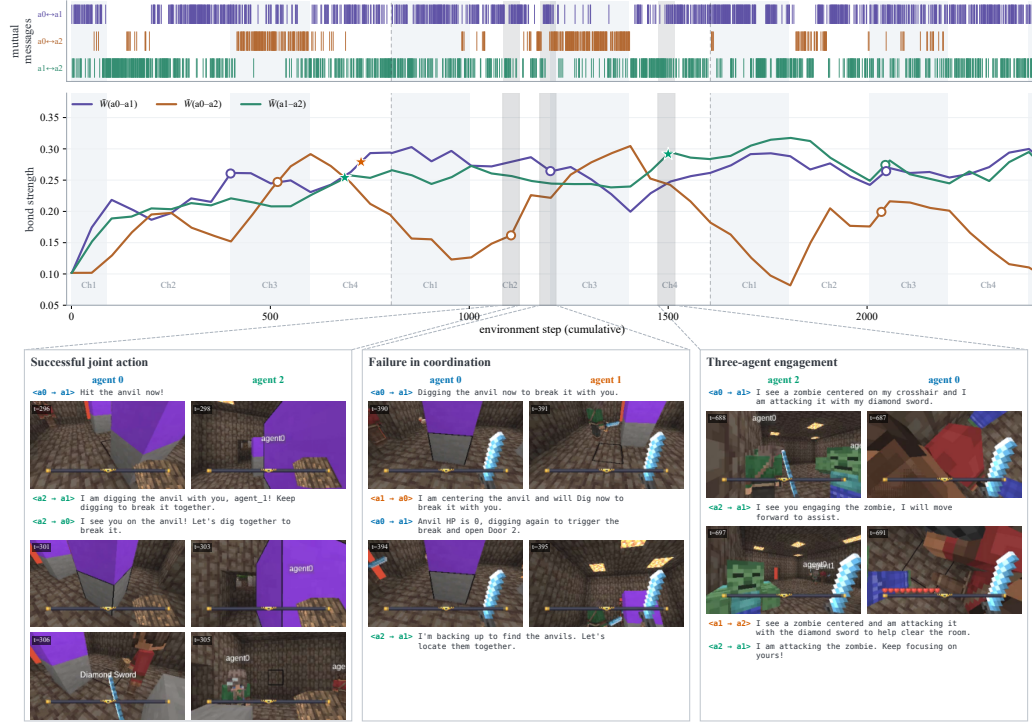


Figure 11: **Qualitative coordination dynamics for Qwen3.5-9B+Heb.** Mutual pairwise messages and bond strengths are shown over three episodes (seed 42), with representative interaction windows expanded below. *Left:* agents coordinate a successful joint anvil action after explicit mutual confirmation. *Centre:* agents repeatedly communicate intent to cooperate but continue acting on an already exhausted anvil, and no joint completion fires. *Right:* all three agents engage in the combat task. The examples illustrate that learned bonds capture persistent interaction structure, while successful cooperation still depends on grounding and executing the intended joint action.

G.2 WHAT DIFFERENT CO-FIRING CHANNELS LOOK LIKE (RQ2)

The co-firing comparison of Table 4 found that imitation produces the most behaviourally informative bonds while communication produces the strongest but least useful ones. The two examples below show what each channel looks like at the step level in the Gemma-4-E4B co-firing arms.

Example 4 — Imitation: co-firing with no communication channel Imitation arm (Gemma-4-E4B), seed 123, ep. 3, Ch3, $t=510-521$

- $t=510-519$ *agent 2 imitates agent 0 on every step; the imitation co-firing events raise $W_{2,0}$ from 0.007 to 0.012 (+0.0025 per event).*
- $t=520$ *agent 0 [Dig] presses the switch: M13_SWITCH_PRESSED (+40) and M14_DOOR_OPENED (+60) fire. The same step logs another imitation tick ($W_{2,0}=0.014$) and the episode's largest co-firing event, between agents 0 and 1 ($\Delta W=0.010$, spatial channel).*
- $t=521$ *agent 2 is still imitating agent 0; $W_{2,0}=0.017$.*

Example 5 — Communication: planning with no action Communication arm (Gemma-4-E4B), seed 42, ep. 2, Ch4, $t=690-720$, $W=0.254$

$t=690$	a0 → a2 [TurnRight]	“Agent 2, can you confirm the location of the 3 zombies in Ch4? I need to orient myself to attack.”
$t=692$	a2 → a1 [MoveForward]	“I am moving forward to locate the zombies. Agent 1, what is your priority target?”
$t=695$	a1 → a2 [NoOp]	“I propose we flank them from the south while you engage from the main path.”
$t=697$	a2 → a1 [none]	“Agent 1, I’m ready to engage. Should I move towards the main path or flank south as you suggested?”
	...	(agent 1 repeats its flanking proposal verbatim eight times; agent 0’s question goes unanswered six times; the logged actions stay MoveForward/NoOp)

⇒ no zombie is attacked in the window. The plan is coherent as text and never executed, while the pair’s bond keeps growing through the communication channel.

G.3 IMPOSED GRAPHS VERSUS EARNED RELATIONSHIPS (RQ3)

The frozen-topology experiment (Table 6) found that imposing bonds does not by itself improve cooperation. Example 6 shows one existing failure case of coordination.

Example 6 — An imposed maximal bond does not synchronise Allied-all (Qwen3.5-9B, frozen $W=0.8$), seed 1011, ep. 1, Ch3, $t=424-472$

$t=429$	a1 → a0 [LookDown]	“Agent_0, I am looking down to find the exit path. My door status is LOCKED.”
$t=430$	a1 → a0 [TurnLeft]	“I am turning left to find the exit gap in my cell. Let me know if you see anything useful.”
$t=431$	a0 → a1 [TurnLeft]	“I am turning left to find the exit gap in my cell. Let me know if you see anything useful.”
$t=437$	a2 → a0 [Dig]	“Switch pressed, opening your door.”
$t=444$	a0 → a1 [TurnRight]	“I am checking my left wall for the blue switch or exit corridor.”

⇒ agents 0 and 1 keep asking each other for door status until $t=472$ without either exiting.

The transplant experiment (Table 5) contrasts these imposed graphs with relationships that were actually experienced. Example 7 shows the reunion in practice: in episode 1 of a transplant run, all three transplanted dyads run their own switch-press negotiation with their former partner in parallel, each agent’s dominant message target is its Phase-A partner.

Example 7 — Reunion: three transplanted dyads coordinate in parallel Transplant (Gemma-4-E4B), seed 42, ep. 1, Ch3, $t=110-122$

$t=111$	a2 → a3	“I’m pressing the switch now, let me know if the door state changes!”
$t=113$	a3 → a2	“Door state?”
$t=113$	a1 → a0	“Waiting for switch press. Let me know when you confirm door change.”
$t=116$	a5 → a4	“I see the blue cube slightly to the right; I’m turning to face it now.”
$t=119$	a0 → a1	“Digging switch now. Tell me when the door status changes.”
$t=122$	a0 → a1	“Pressing switch now, check door status.”

⇒ M13_SWITCH_PRESSED (+40) fires at $t=122$. All six agents are attached to their transplanted partner; the three dyads work the same puzzle side by side without cross-talk.

Example 8 shows the re-pairing event described in the main text. The shuffled arm shows no such behaviour, in the matched shuffled run, agent 0’s dominant target drifts from agent 2 (episodes 1–2) to agent 5 (episode 3) and its seatmate share never exceeds 0.08, while the one stranger dyad that does establish a partnership (pair 4–5, shares 0.71–0.92 in episodes 1–2) dissolves again in episode 3.

Example 8 — The re-pairing episode and the return Transplant (Gemma-4-E4B), seed 42, eps. 2–3, Ch3

Episode 2 — the team reorganises into three new reciprocal dyads ($0 \leftrightarrow 4$, $1 \leftrightarrow 2$, $3 \leftrightarrow 5$); the share of messages to the former partner collapses to ≤ 0.03 :

t=41 a2 → a1 “Still locked. I’m pressing the switch again.”

t=45 a1 → a2 “I’m here and watching the door state closely. Let me know when you press it again.”

t=115 a4 → a0 “Switch still locked? Try again.”

t=199 a0 → a4 “Agent 4, check door state again.”

Episode 3 — the transplanted pairs re-emerge ($0 \leftrightarrow 1$, $2 \leftrightarrow 3$, $4 \leftrightarrow 5$; partner shares 0.50–0.65):

t=22 a2 → a3 “I am digging the switch again, hoping to trigger the door change. Let me know if you see the door open!”

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 three 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, and the social-module prompt that conditions the agent on its Hebbian bond row. The remaining templates (auto-curriculum, critic, belief updates, episode summarisation) follow the same conventions and are available in the released code.

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 4.4); 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).
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.

System prompt template (continued)

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.

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 the Hebbian communication co-activity term and the Chamber 3 puzzle both depend on. In RL mode the prompt is identical except that the `action` field is removed from the response format and the model is told explicitly that “a separate policy will pick the action based on YOUR thoughts plus this context”; the emitted thoughts are appended to the scoring prompt so the policy samples from $p(a \mid \text{prompt, thoughts})$. All fields are filled at runtime; `{social_directive}` carries the social module’s rendered decision when the module is active, and the raw bond row otherwise.

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 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 4.2) 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 (*prompt coupling*); under the *bias* coupling its `ask_target` additionally overrides the action model’s `communication_target` at the routing layer. 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.
 - “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.

Social-module deliberation prompt (continued)

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.