Benchmarking Language Agents on Open-Ended Multi-Agent Coordination in Game Worlds

Anonymous authors
Paper under double-blind review

Abstract

As language models are increasingly deployed as autonomous agents, they will need to coordinate with others in long-horizon, open-ended interactive tasks. Yet current evaluations rarely test these demands together, focusing instead on short interactions, single-agent open-ended tasks, or highly structured multi-agent settings. We introduce ALEM, a JAX-based, procedurally generated open-ended benchmark for multi-agent coordination built on Craftax-like dynamics. ALEM embeds procedurally generated coordination tasks, soft specialisation, communication, and controllable coordination difficulty into a long-horizon, game-like survival world with exploration, crafting, trading, and combat. We evaluate 13 modern LLMs zero-shot within homogeneous teams, with trained MARL agents as reference points. Most LLM agents struggle, averaging only $\sim 6\%$ of maximum reward, but their failures are not uniform. Gemini-3.1-Pro-High approaches MARL agents trained for one billion steps on HARD coordination (17.5% vs. 17.6% Coord.%), while GPT-5.4-High achieves strong base-task reward but much lower coordination reward. This contrast shows both the promise and the limitation of frontier LLM agents - a zero-shot model can approach trained MARL coordination performance, yet individual task competence does not imply coordination competence. Ablations further show that communication is central for sharing intent, while memory and reasoning help when they support multi-step planning. These results identify coordination as a distinct bottleneck for current LLM agents, separate from single-agent capabilities. ALEM makes this bottleneck measurable, providing a controlled setting for developing agents that can communicate, allocate roles, and execute shared plans.

1 Introduction

Large Language Models (LLMs) have evolved rapidly, from systems primarily used for text generation (Brown et al., 2020; Vaswani et al., 2017; Devlin et al., 2019) to agentic systems capable of completing *long-horizon*, *open-ended tasks* (Kwa et al., 2025; Yamada et al., 2025; Chan et al., 2025; Lu et al., 2026; Nakano et al., 2021; Wang et al., 2024a). As these systems are increasingly deployed as autonomous agents, economic and practical incentives are likely to accelerate their adoption in complex multi-agent settings (Hammond et al., 2025), where success is contingent on coordinating with others. Yet the mechanisms required for sustained multi-agent coordination remain poorly understood, especially when agents must communicate intent, allocate roles, maintain shared plans, and align actions over long horizons (Malone & Crowston, 1994; Guo et al., 2024). This creates a need to evaluate whether LLM agents can coordinate effectively in long-horizon, open-ended multi-agent environments, and to characterise the failures that arise when they cannot.

Existing benchmarks cover only part of this space. MultiAgentBench, Collab-Overcooked, and related multi-agent LLM benchmarks evaluate collaboration, but typically in shorter-horizon and more narrowly specified tasks (Zhu et al., 2025; Sun et al., 2025; Agashe et al., 2025; Wang et al., 2024b; Grötschla et al., 2025; Meireles et al., 2025). Others focus on long-horizon task completion

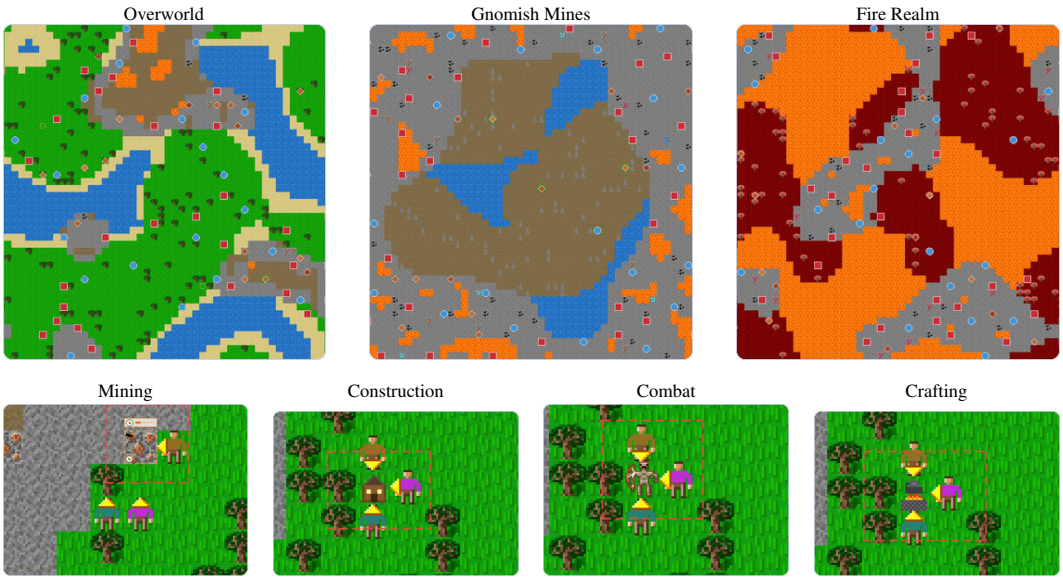


Figure 1: **ALEM extends Craftax-like open worlds (Matthews et al., 2024; Omari et al., 2025) into a controllable multi-agent coordination benchmark.** Top: procedurally generated levels with sampled coordination tasks (● 2-agent sync, ■ all-agent sync, ♦ handover). Bottom: coordinated mining, construction, combat, and crafting examples. By resampling tasks and coordination structure each episode, ALEM evaluates agents' ability to infer coordination needs from observation rather than memorise fixed task patterns.

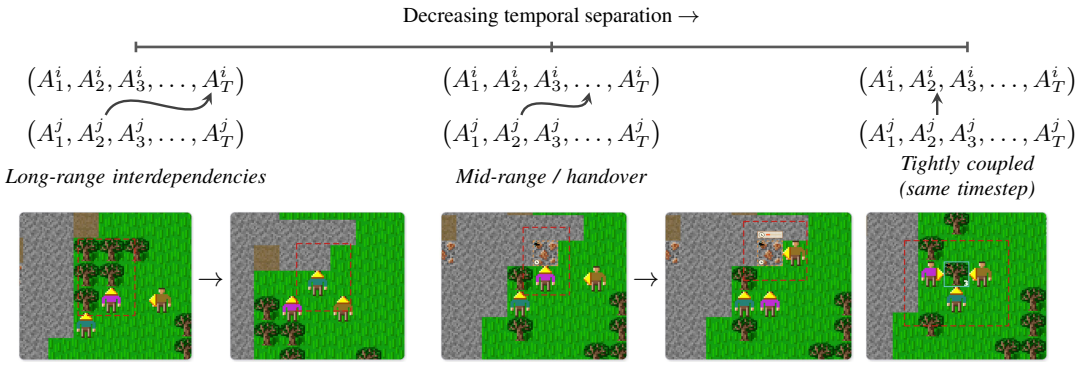
in primarily single-agent settings (Paglieri et al., 2025; Zhang et al., 2026; Sinha et al., 2026; Wang et al., 2025; Zhao et al., 2026; He et al., 2026; Grady et al., 2026), or include multiple agents with limited coordination demands (Yang et al., 2025). The community therefore lacks evaluations that jointly test *long-horizon* interaction, *open-ended* task structure, and *coordination*. Games are established testbeds for sequential decision-making (Bellemare et al., 2013; Brown & Sandholm, 2018; Schrittwieser et al., 2020; Vinyals et al., 2019; Campbell et al., 2002), and procedural generation is especially useful because it reduces memorisation and contamination while enabling controlled variation across tasks and difficulty levels (Paglieri et al., 2025; Shojaee et al., 2026).

To address these gaps, we introduce ALEM[1](Fig. 1), a benchmark for open-ended[2] multi-agent coordination built on Craftax-like dynamics (Matthews et al., 2024; Omari et al., 2025). ALEM procedurally generates complex environments where coordination difficulty is modulated by a single parameter, ranging from loose temporal dependencies and handovers to tightly joint actions (Fig. 2). Implemented in JAX (Bradbury et al., 2018), the benchmark facilitates scalable and reproducible evaluation of how multi-agent failure modes emerge as coordination demands increase.

Using ALEM, we evaluate 13 LLMs in zero-shot homogeneous teams, with trained MARL agents as reference points. Most LLM agents struggle, averaging 6% reward with varied failure modes. GPT-5.4-High's performance shows that individual competence does not imply coordination. Communication is the biggest contributor to aligning shared intent in Gemini and Gemma models. We find that multi-turn scratchpad planning and reasoning are critical for coordinated execution. Furthermore, heterogeneous teams perform near the average of their components rather than inheriting the strongest member's performance. These results demonstrate that open-ended coordination is a distinct challenge from individual task performance.

[1]Alem translates to "world" in a number of languages.

[2]Following Craftax-style benchmarks (Matthews et al., 2024; Omari et al., 2025), we use *open-ended* to mean a large, procedurally generated but finite goal space, not unbounded novel goals as in Hughes et al. (2024).



(a) Long-range (b) Handover (c) Synchronous

Figure 2: **Coordination coupling spectrum in ALEM.** ALEM tasks vary the temporal separation $\Delta t = |t - t'|$ between coupled actions A_t^i and $A_{t'}^j$. We show representative examples: *long-range* (2a, large Δt), where one agent gathers wood used to craft a pickaxe, enabling another agent to mine stone later; *handover* (2b, small Δt), where agent i initiates a mining task that agent j must complete within a short window; and *synchronous* (2c, $\Delta t = 0$), where agents act on the same target at the same timestep. Dashed boxes mark the coordination-critical regions.

Our contributions are: **i**) ALEM, a JAX-based benchmark for long-horizon, open-ended multi-agent coordination with procedural coordination tasks, soft specialisation, communication, and controllable difficulty; **ii**) a shared evaluation interface for comparing zero-shot LLM agents with trained MARL policies in the same environment; and **iii**) an evaluation of 13 modern LLMs and 4 MARL baselines, showing that coordination remains a distinct bottleneck from base task competence and exposing failure modes across communication, memory, reasoning, and team composition.

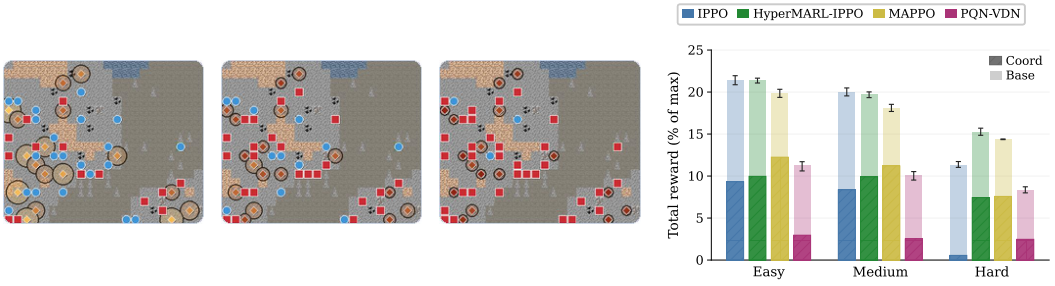
2 ALEM: An Open-Ended World for Multi-Agent Coordination

We introduce ALEM, a benchmark for studying coordination in open-ended multi-agent worlds. Built on Craftax-Coop (Omari et al., 2025), ALEM extends prior work along four axes: diverse *coordination types*, *procedural generation* of coordination tasks, *soft specialisation*, and an explicit *communication* channel. Across nine procedurally generated levels, agents explore, trade resources, craft tools, build structures, and fight mobs under controllable coordination demands. By scaling these demands while preserving the underlying world structure, ALEM enables controlled evaluation of multi-agent coordination. Information-theoretic diagnostics verify that its mechanics require genuine multi-agent interdependence (Tessera et al., 2026) (App. G.1), while its JAX implementation supports billion-step MARL training on a single GPU in under two days (App. H.3).

2.1 Coordination Types

Following Malone & Crowston (1994), we view coordination as *interdependence between activities*. In multi-agent systems, these activities correspond to agents' actions. We characterise the dependencies between coupled actions A_t^i and $A_{t'}^j$ by their *temporal separation*, $\Delta t = |t - t'|$, resulting in a *coordination spectrum* from long-range dependencies to simultaneous joint action (Fig. 2).

Existing MARL benchmarks typically isolate specific regions of this spectrum: SMAC (Samvelyan et al., 2019) emphasises synchronous focus-fire ($\Delta t \approx 0$), Overcooked (Carroll et al., 2019) tests short-window handovers (small Δt), and Melting Pot's Clean Up (Agapiou et al., 2022) targets long-range resource interdependence (large Δt). ALEM instantiates this entire spectrum within a single open-ended environment, using procedural generation to vary both the temporal structure of coordination and the number of agents involved (details App. G.2).



Easy (α=0.30) Medium (α=0.60) Hard (α=0.90)

Figure 3: **ALEM difficulty settings and MARL baselines. Left:** The same generated world under Easy, Medium, and Hard settings, with coordination sites marked as ● 2-agent sync, ■ all-agent sync, and ♦ handover. Increasing α preserves the layout and coordination opportunities, but tightens execution constraints through more all-agent requirements, shorter handover windows, and stronger specialisation. **Right:** Trained MARL baseline performance, reported as mean percentage of max achievable reward with 95% CIs and decomposed into coordination (dark) and base (light) reward.

2.2 Procedural Coordination Generation

While related environments procedurally generate spatial layouts (Omari et al., 2025; Matthews et al., 2024; Ye et al., 2025; Hafner, 2022), fixed coordination tasks still allow agents to overfit to interaction patterns. ALEM therefore procedurally generates the *coordination structure* itself. Eligible entities across tasks like mining, construction, combat, and crafting (Fig. 1) are independently assigned a coordination type (synchronous or handover) and a required agent count (2 or N). Synchronous tasks further adopt game-theoretic incentives: *hard* tasks act as pure coordination problems requiring joint action for any reward (Schelling, 1980), while *soft* tasks function as assurance games, balancing a safe solo payoff against a larger joint-action bonus (Skyrms, 2004) (details App. G.2).

2.3 Soft Specialisation and Difficulty Scaling

Instead of enforcing rigid division of labour via hard role gates (Agapiou et al., 2022; Omari et al., 2025), ALEM uses *soft specialisation*. Any agent can attempt any role-gated action, but specialists succeed deterministically ($\eta_{\text{spec}} = 1$) while non-specialists succeed with probability $\eta_{\text{non}} = 1 - \alpha$.

We scale overall coordination difficulty via this single scalar $\alpha \in [0, 1]$. Because world generation in JAX is deterministic, varying α preserves the spatial layout and the frequency of coordination opportunities (Fig. 3), cleanly isolating execution difficulty from world variation. Higher α tightens execution constraints: it increases the probability of N-agent requirements, shortens handover windows, increases solo failure rates on soft tasks, and lowers non-specialist efficiency. We evaluate on Easy ($\alpha = 0.3$), Medium ($\alpha = 0.6$), and Hard ($\alpha = 0.9$) settings (parameters detailed in App. G.3).

2.4 Environment Interface

ALEM exposes both symbolic and pixel observations of the underlying state, including inventory, teammate status, recent messages, and visible coordination constraints. Agents act through a discrete action space covering movement, crafting, combat, communication, resource requests, and teammate-directed interactions (e.g. reviving teammates or requesting resources). For LLMs, we provide a structured text interface that maps these observations and legal actions to a prompt, and parses each model response into actions, with optional reasoning, communication, and memory fields. Full details are given in App. B.1.

2.5 MARL Baselines

We train IPPO (De Witt et al., 2020), MAPPO (Yu et al., 2022), PQN-VDN (Gallici et al.), and HyperMARL-IPPO (Tessera et al., 2025) to provide reference points for zero-shot LLM performance and to verify that the difficulty settings meaningfully affect learned policies. Figure 3 reports 1B-step performance (hyperparameter sweeps detailed in App. I). Performance sharply degrades as coordination difficulty increases; even after 3B environment steps, the strongest baseline reaches only 18% of the maximum total reward on the Hard setting (App. H.4), confirming that ALEM provides a rigorous, unsaturated challenge for current MARL methods.

3 Experiments

We evaluate 13 modern LLMs in zero-shot homogeneous teams, alongside 4 MARL baselines, to isolate and measure coordination capabilities in open-ended, long-horizon environments. While the underlying open-ended world is inherently challenging, calibration against a single-agent baseline confirms that ALEM's multi-agent dynamics result in a statistically significant drop in performance, showing that coordination introduces a distinct layer of difficulty (App. J.1).

With this structural difficulty confirmed, our experiments ask two questions: **Q1:** How well do LLM agents coordinate zero-shot within homogeneous teams? **Q2:** How do communication, memory, and reasoning affect coordination? Additional heterogeneous-team results are reported in App. C.2.

3.1 Experimental Setup

Evaluation protocol. We follow a zero-shot evaluation protocol similar to BALROG (Paglieri et al., 2025). At each timestep, each agent receives the game rules, environment details, local observation, and legal actions in text form through the interface in Sec. B.1.2. By default, agents have scratchpad memory, broadcast communication, and a text history of the last eight observations and actions. Success in ALEM requires spatial reasoning, long-horizon planning, tracking crafting dependencies, memory, effective communication, and coordination under partial observability.

Metrics. Our primary metric is normalised episode return – following previous Craftax-based environments (Omari et al., 2025; Matthews et al., 2024), each agent's cumulative reward is divided by the maximum available reward for the relevant category and then averaged across agents, so 1.0 indicates that all rewards in that category were obtained (details in App. H.2).

Models. We evaluate 13 modern language models. Closed-source models are evaluated via their respective APIs (GPT-5.4 (OpenAI, 2026) and Gemini 3.1 Pro (Google DeepMind, 2026a)). Open-weight models are served with vLLM (Kwon et al., 2023) and include Gemma-4-E4B-it/26B-A4B-it/31B-it (Google DeepMind, 2026b), Llama-3.1-8B-Instruct and Llama-3.3-70B-Instruct (Grattafiori et al., 2024), Qwen-3.5-9B/27B/35B-A3B/122B-A10B (Team, 2026), and Qwen-3.6-27B/35B-A3B (Qwen Team, 2026). Where applicable, reasoning is enabled, with proprietary models configured to high-reasoning settings. We run 20 seeds per difficulty for open-weight models and 10 for closed-source models (due to API costs), reporting cross-seed means with 95% stratified bootstrap confidence intervals using the rliable library (Agarwal et al., 2021).

Agent harness. Language models are evaluated with the agent harness described in Sec. B.1.2 and App. H.5. The harness follows a reason-then-act paradigm similar to ReAct (Yao et al., 2023), but generates a new prompt at each step focusing evaluation on the reasoning and coordination capabilities of the model instead of confounding factors such as context length or compaction strategy.

Table 1: **Zero-shot coordination of homogeneous teams in ALEM.** Performance across Easy, Medium, and Hard difficulties, with trained MARL baselines as reference. Models are ordered descending by their Total% score in the Hard difficulty setting. We report mean episode return with 95% stratified bootstrap confidence intervals. Base%, Coord.%, and Total% are normalised independently by the maximum achievable reward for their respective categories (Total% is not a direct sum). MARL 100M and 1B denote environment training steps (not model parameters), reporting the best-performing algorithm for each compute budget.

	Easy			Medium			Hard		
Method	Base%	Coord.%	Total%	Base%	Coord.%	Total%	Base%	Coord.%	Total%
gemini-3.1-pro-high	18.1 [14.2,21.8]	18.0 [12.5,23.9]	18.1 [14.8,21.0]	15.1 [12.3,18.0]	23.4 [20.3,27.0]	18.6 [16.3,21.3]	13.8 [11.2,16.6]	17.5 [13.5,21.5]	15.4 [13.2,17.8]
gpt-5.4-high	14.1 [11.5,16.7]	7.0 [3.6,10.8]	11.1 [9.2,13.0]	10.0 [6.5,13.5]	4.0 [2.1,5.7]	7.4 [4.9,9.8]	11.8 [8.0,15.3]	4.2 [2.0,6.6]	8.6 [5.9,10.8]
gemma-4-31b-it	9.6 [7.4,11.9]	8.6 [6.4,11.0]	9.2 [7.6,10.8]	9.7 [8.0,11.3]	10.6 [8.3,13.0]	10.1 [9.0,11.1]	7.3 [5.9,8.9]	8.8 [6.9,10.9]	8.0 [6.7,9.1]
gemma-4-26b-a4b-it	4.3 [2.6,6.2]	7.3 [5.0,9.6]	5.6 [3.8,7.4]	4.1 [2.8,5.3]	6.1 [3.8,8.3]	4.9 [3.3,6.4]	5.9 [5.3,6.6]	9.5 [7.7,11.3]	7.4 [6.6,8.2]
qwen3.6-27b	5.1 [3.4,6.9]	4.7 [2.9,6.7]	4.9 [3.4,6.5]	7.0 [5.4,8.8]	6.1 [4.3,7.9]	6.6 [5.4,7.9]	7.5 [5.7,9.5]	7.2 [5.4,8.9]	7.4 [6.1,8.8]
qwen3.6-35b-a3b	2.8 [1.4,4.5]	4.4 [2.3,6.7]	3.5 [1.8,5.3]	6.3 [5.0,7.9]	9.7 [7.7,11.6]	7.7 [6.3,9.1]	6.3 [4.8,8.1]	8.7 [6.7,10.7]	7.3 [6.0,8.6]
qwen3.5-27b	4.8 [3.1,6.9]	7.0 [4.9,9.2]	5.7 [4.1,7.4]	4.1 [2.4,6.0]	6.5 [3.8,9.3]	5.1 [3.1,7.2]	5.1 [3.5,7.1]	8.1 [5.8,10.4]	6.4 [4.8,7.9]
qwen3.5-122b-a10b	5.4 [4.3,6.5]	8.4 [6.6,10.0]	6.6 [5.5,7.7]	5.0 [4.6,5.3]	9.6 [7.5,11.5]	6.9 [6.0,7.9]	4.7 [3.9,5.5]	8.3 [6.2,10.6]	6.2 [5.0,7.4]
qwen3.5-35b-a3b	1.9 [1.0,2.9]	4.2 [2.1,6.3]	2.9 [1.6,4.2]	2.2 [1.4,3.1]	4.7 [2.7,6.8]	3.3 [2.0,4.6]	2.9 [2.2,3.7]	5.3 [3.4,7.5]	4.0 [2.7,5.2]
qwen3.5-9b	2.7 [2.0,3.4]	7.0 [4.6,9.5]	4.6 [3.2,5.9]	2.7 [1.6,4.0]	4.7 [2.7,7.1]	3.6 [2.2,4.9]	2.6 [1.7,3.4]	5.0 [2.9,7.1]	3.6 [2.3,4.8]
gemma-4-e4b-it	2.4 [1.9,2.9]	4.6 [3.0,6.3]	3.3 [2.6,4.1]	2.4 [1.9,2.8]	3.1 [1.7,4.5]	2.7 [2.0,3.3]	2.3 [1.9,2.7]	3.2 [1.9,4.5]	2.7 [2.1,3.3]
llama-3.3-70b-instruct	1.6 [1.1,2.3]	5.0 [3.2,6.8]	3.1 [2.2,4.0]	1.1 [0.8,1.5]	2.9 [1.4,4.4]	1.9 [1.3,2.5]	1.4 [1.1,1.7]	4.2 [2.7,5.8]	2.6 [1.8,3.3]
llama-3.1-8b-instruct	1.4 [1.1,1.7]	0.0 [0.0,0.0]	0.8 [0.6,1.0]	1.4 [1.1,1.6]	0.0 [0.0,0.0]	0.8 [0.6,0.9]	1.3 [1.0,1.5]	0.3 [0.0,0.9]	0.9 [0.7,1.2]
Across LLM agents	5.7	6.6	6.1	5.5	7.0	6.1	5.6	6.9	6.2
MARL 100M	11.6 [11.1,12.1]	20.0 [19.0,21.4]	15.1 [14.6,15.6]	12.1 [11.5,12.8]	17.0 [16.2,18.0]	14.2 [13.5,15.0]	12.2 [11.6,12.8]	5.6 [5.1,6.2]	9.4 [8.9,9.9]
MARL 1B	21.0 [19.4,22.6]	22.0 [20.8,23.1]	21.4 [20.9,21.9]	20.1 [19.6,20.7]	19.8 [19.4,20.2]	19.9 [19.5,20.5]	13.6 [12.8,14.2]	17.6 [16.9,18.2]	15.3 [14.9,15.7]

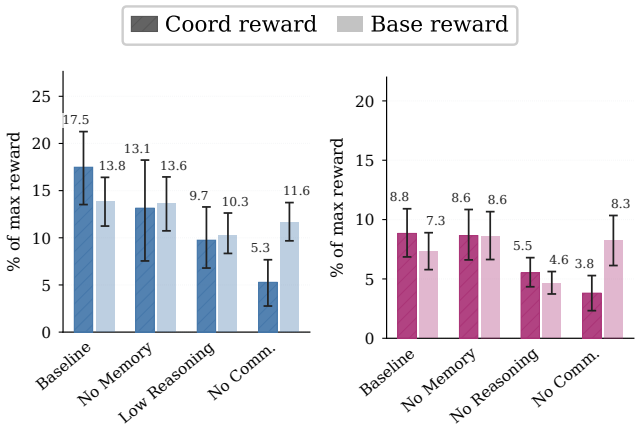
3.2 Results

3.2.1 Q1: How well do LLM agents coordinate zero-shot within homogeneous teams?

ALEM is unsolved but separates models. Table 1 shows that ALEM remains far from saturated by both zero-shot LLMs and trained MARL baselines, while still separating models across a broad performance range. On Easy, Total% spans from 0.8% for Llama-3.1-8B-Instruct to 18.1% for Gemini-3.1-Pro-High; on Hard, LLMs average only 6.2% Total%. Yet the top model, Gemini-3.1-Pro-High, reaches 17.5% Coord.% on Hard, comparable to the 17.6% Coord.% of MARL agents trained for one billion environment steps. Furthermore, many models also survive for long episodes despite low returns (Fig. 5c), suggesting that the bottleneck is not basic survival or interface parsing (App. H.5), but converting interaction time into coordinated progress.

Base competence does not guarantee coordination. Decomposing performance reveals failures masked by aggregate scores. GPT-5.4-High achieves the second-highest Base% across difficulties ($14.1, 10.0, 11.8\%$), but lower Coord.% ($7.0, 4.0, 4.2\%$), falling behind smaller models such as Gemma-4-26B-A4B-it with non-overlapping 95% CIs on Medium and Hard. Its behaviour also shows that local cooperation need not translate into sustained coordination: GPT-5.4-High has the highest trade count (48.9) but few revives (3.4), while Gemini-3.1-Pro-High is more balanced (29.3 trades, 6.9 revives; Fig. 5b). Conversely, Qwen3.5-122B-A10B and Gemma-4-26B-A4B-it achieve higher Coord.% than Base%, indicating that agents can exploit coordination opportunities even when general environment competence is limited. Together, these results show that base-task competence does not guarantee coordination.

Most LLM agents do not exploit the difficulty axis. Unlike trained MARL agents, which degrade as coordination difficulty increases (Sec. 2.3), aggregate LLM Total% is nearly flat at $\approx 6\%$ across Easy, Medium, and Hard. This suggests that most LLM agents do not yet coordinate reliably enough for the difficulty parameter to strongly affect aggregate performance. This is visually corroborated by their stagnant achievement tier coverage (Fig. 10), which shows most LLMs failing to progress beyond the Basic tier achievements regardless of difficulty, whereas capable models (e.g., MARL 1B, Gemini-3.1-Pro-High) exhibit clear degradation at intermediate and advanced tiers.



(a) Gemini-3.1-Pro **(b)** Gemma-4-31B

Figure 4: **Harness ablations on Hard.** We ablate communication, scratchpad memory, and reasoning for Gemini-3.1-Pro-High and Gemma-4-31B-it. Bars show Coord.% and Base%, normalised by the maximum achievable reward in each category, with 95% bootstrap confidence intervals.

Coordination failures reflect task structure, priors, and scale. Appendix diagnostics show that coordination failures vary by task structure and are not explained by model scale alone. Handover tasks are comparatively more accessible because they allow temporal slack, while *Synchronous-Hard* tasks require agents to identify a shared target, reach compatible positions, and act in the same timestep (App. C.1, Fig. 5a).

Sparse *Construction* tasks are harder still as they require long-horizon resource collection, technology progression, convergence on a shared site, and synchronised execution. Coverage is $\leq 2\%$ for every LLM except Gemini-3.1-Pro-High (7%), and 0% for the MARL 1B reference, suggesting that these tasks are difficult to discover through trial-and-error at this compute budget, while pretrained priors can help to a limited extent (Fig. 5a). Across open-weight models, performance is also not monotonic in parameter count, suggesting that coordination depends on factors beyond raw scale, such as post-training, communication use, and planning behaviour (Fig. 5c).

3.2.2 Q2: How do communication, memory, and reasoning affect coordination?

We next ablate three components of the LLM agent harness: scratchpad memory, communication, and reasoning (Sec. 3.1). We run these ablations on the strongest closed-source and open-weight models from Sec. 3.2.1, Gemini-3.1-Pro-High and Gemma-4-31B-it, on the Hard setting. Across these two models, communication produces the largest coordination drop, followed by reasoning, then memory. Base reward is less sensitive to these ablations, except for reasoning, which consistently reduces both Base% and Coord.%. However, the mechanisms differ by model.

Communication is critical for coordination. Fig. 4 shows that removing communication produces the largest drop in coordination performance. Gemini-3.1-Pro-High falls from 17.5 to 5.3 Coord.%, while Gemma-4-31B-it falls from 8.8 to 3.8 Coord.%. The corresponding changes in Base% are smaller and less consistent (Gemini $13.8 \rightarrow 11.6$, Gemma $7.3 \rightarrow 8.3$), suggesting that communication primarily supports coordination rather than general environment progress. Message statistics are consistent with this interpretation: agents frequently address a specific teammate (Gemini 43.4%, Gemma 44.1%) and often use imperative verbs (Gemini 73.7%, Gemma 52.2%). This suggests that agents largely use the channel to broadcast intent, assign actions, and align timing. Without it, they can still make base progress but struggle to coordinate effectively (see qualitative analysis in App. N).

Scratchpad memory helps when it is used for planning. The memory ablation is model-dependent. Removing scratchpad memory lowers Gemini-3.1-Pro-High's mean Coord.% from 17.5 to 13.1, while leaving Base% nearly unchanged (13.8 → 13.6). For Gemma-4-31B-it, the mean changes are small in Coord.% (8.8 → 8.6) and Base% (7.3 → 8.6), with overlapping confidence intervals. This suggests that scratchpad memory is not uniformly useful; it depends on how models use it. Qualitative analysis gives one possible explanation (App. L). Gemini uses the scratchpad as a *forward-looking planner*: 71.5% of entries are multi-line, 75.3% contain future-oriented terms such as *will*, *next*, *then*, or *plan*, and 12.4% contain explicit turn-indexed action sequences such as `T9:Do`, `T10:Move West`, and `T11:Give wood to Agent 1`. In contrast, Gemma's scratchpads are mostly one-line state summaries (average 1.08 lines), with frequent coordinate references (98.8%) and status information that is often already present in the next observation. These patterns suggest that scratchpad memory appears more helpful when models use it to preserve plans or commitments across turns, and less helpful when it mostly restates the current state.

Reasoning supports both coordination and base progress. Reasoning is the only ablation that lowers both Base% and Coord.% for both models (Fig. 4). For Gemini-3.1-Pro-High, the ablation is partial because the API only allows us to reduce reasoning, not disable it. Even so, Coord.% falls from 17.5 to 9.7 and Base% from 13.8 to 10.3. For Gemma-4-31B-it, disabling reasoning reduces Coord.% from 8.8 to 5.5 and Base% from 7.3 to 4.6.

Qualitative traces suggest different failure modes (App. M). For Gemini, reduced reasoning removes much of the planner-like behaviour: scratchpad length falls from 207 to 69 characters, turn-indexed plans nearly disappear (12.4% → 0.1%), and messages become shorter (132 → 53 characters). Gemma instead writes longer scratchpads without reasoning (153 → 228 characters), often near the ∼1000-character cap, suggesting that some deliberation moves into the scratchpad. This compensation is incomplete, as both Base% and Coord.% still fall.

4 Related Work

Existing benchmarks cover parts of the setting studied here, but not the full combination of long-horizon interaction, open-ended progression, and explicit multi-agent coordination. Long-horizon LLM-agent benchmarks test reasoning, exploration, and sustained execution, but are primarily single-agent or have limited coordination demands (Kwa et al., 2025; Yamada et al., 2025; Chan et al., 2025; Paglieri et al., 2025; Zhang et al., 2026; Zhao et al., 2026; He et al., 2026). Multi-agent LLM benchmarks study collaboration, communication, and role specialisation (Hong et al., 2023; Du et al., 2023; Zhu et al., 2025; Sun et al., 2025; Grötschla et al., 2025), but typically use shorter horizons, highly specified tasks, or stronger role constraints. MARL benchmarks capture important coordination patterns, but usually isolate these patterns rather than embedding them in an open-ended agentic world. ALEM differs by combining these ingredients in a single procedurally generated benchmark: building on Craftax-Coop (Omari et al., 2025), it introduces procedurally sampled coordination tasks, soft specialisation, explicit communication and controllable coordination difficulty. We also use diagnostics (Tessera et al., 2026) to verify that our environment demands genuine multi-agent interdependence (App. G.1). Extended related work is provided in App. D.

5 Conclusions

We introduce ALEM, a JAX-based benchmark for long-horizon, open-ended multi-agent coordination. Our zero-shot evaluations of 13 LLMs and trained MARL baselines show that coordination remains a clear bottleneck for current agents, and that base-task competence does not guarantee collaborative success. By exposing failure modes across communication, memory, and reasoning, ALEM makes coordination measurable and provides a testbed for developing more collaborative agents.

A Background

We model the environment as a partially observable stochastic game (POSG) (Hansen et al., 2004), denoted as the tuple $\langle I, S, A, O, \Omega, T, R \rangle$. Here, $I = \{1, ..., n\}$ is the set of agents, S is the set of states, $A = A_1 \times \cdots \times A_n$ is the joint action space, and $O = O_1 \times \cdots \times O_n$ the joint observation space. At each state $s \in S$, an agent i receives an individual observation $o_i \in O_i$ drawn from $\Omega : S \to O$ and selects an action $a_i \in A_i$. The environment transitions according to $T : S \times A \to \Delta(S)$, where $\Delta(S)$ is the distribution over possible states. Each agent receives an individual reward based on the reward function $R_i : S \times A \times S \to \mathbb{R}$ for each i.

B Environment

B.1 Environment Interface

B.1.1 RL Interface

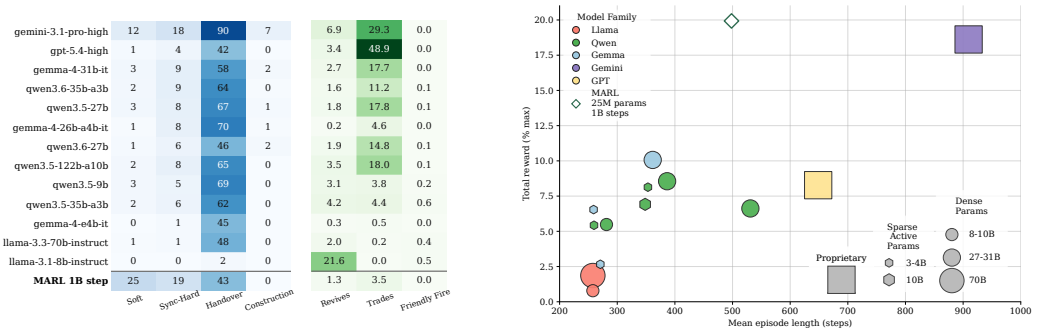
Observation space. ALEM exposes both pixel (Fig. 7) and symbolic observations of the underlying POSG (Hansen et al., 2004). The egocentric observation includes a local map, inventory, and a teammate dashboard (health, specialisation, requested resources, off-screen direction, and recent communications). Beyond Craftax-Coop, ALEM explicitly embeds communication messages, coordination states, including soft/hard requirements, handover countdowns, and multi-agent mob targets.

Action space. Agents navigate a 60-dimensional discrete action space encompassing environment manipulation, crafting, exploration, and combat. To facilitate teamwork, agents can issue persistent resource requests (active for up to 10 timesteps), execute teammate-targeted giving, and broadcast discrete messages. The primary DO action is contextual, it supports both environment interaction and direct agent-to-agent mechanics, such as reviving downed teammates or triggering friendly fire. We apply conservative action masking to filter strictly infeasible choices.

Achievements and rewards. Following prior environments (Matthews et al., 2024; Omari et al., 2025), ALEM pairs a dense health reward with a sparse, first-time reward for completing a hierarchy of achievements. We introduce 27 novel coordination achievements spanning synchronised mining, temporal handovers, cooperative construction, and coordinated combat. Rewards are individual by default; crucially, coordination bonuses are awarded exclusively to participating agents, naturally enforcing strict multi-agent credit assignment (details in App. H.6).

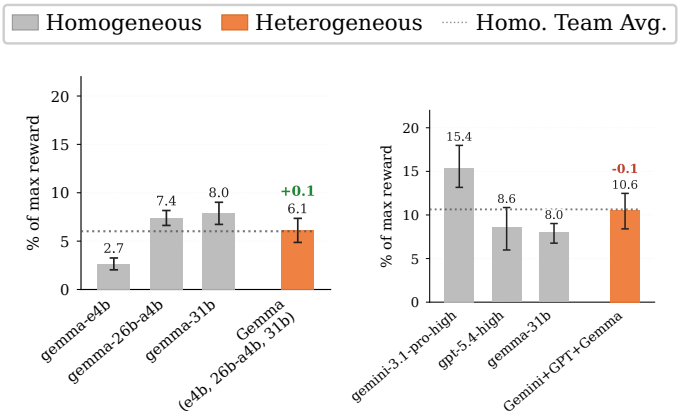
B.1.2 Text-based Language Interface

To evaluate language agents, we expose a structured text interface mapped directly to the underlying environment (App. H.5). At each timestep, models receive a comprehensive textual state—including recent observation-action history, teammate messages, a private scratchpad, and visible coordination constraints—alongside explicit action affordances. Models must return a structured response containing exactly one valid action and, optionally, reasoning, communication, and scratchpad fields. This parsed action is then mapped to the corresponding discrete environment step.



(a) Coordination breakdown **(b)** Cooperative events **(c)** Survival vs. reward

Figure 5: **Diagnostic breakdown of zero-shot homogeneous team failures in ALEM.** (**A**) Coordination reward coverage by coordination type, averaged across difficulties. Cell values report the percentage of the maximum attainable reward within each coordination category. (**B**) Local cooperative event counts per evaluation across all difficulties. (**C**) Mean episode length versus Total% reward on Medium difficulty, showing that longer survival does not necessarily translate into progression or coordinated achievements. Marker size and shape denote model parameter count and architecture type, respectively. MARL 1B is shown as a separated reference row in (A,B) and as a diamond in (C).



(a) Same-family team **(b)** Cross-family team

Figure 6: **Heterogeneous-team performance on Hard.** We compare heterogeneous teams against their constituent homogeneous baselines. Bars show mean Total% with 95% stratified bootstrap confidence intervals. The dashed line marks the average homogeneous-team baseline, and the annotation reports the difference relative to that baseline.

C Results

C.1 Additional Diagnostics

C.2 Heterogeneous Teams

C.2.1 How well do agents coordinate in teams composed of different models?

We evaluate heterogeneous teams on Hard in two settings: a same-family Gemma team composed of Gemma-4-E4B-it, Gemma-4-26B-A4B-it, and Gemma-4-31B-it, and a cross-family team composed of the three strongest LLM agents: Gemma-4-31B-it, GPT-5.4-High, and Gemini-3.1-Pro-High.

Heterogeneous teams perform near the homogeneous-team average. Fig. 6 shows that heterogeneous teams perform similarly to the average of their corresponding homogeneous teams. The same-family Gemma team is slightly above this average ($+0.1$ Total%), while the cross-family team is slightly below it (-0.1 Total%). Thus, heterogeneous teams neither collapse to the weakest member nor inherit the performance of the strongest one.

This suggests that adding a stronger model to a team is not enough to lift team performance. Mixed teams may face an additional coordination problem as agents must align their communication conventions and planning horizons, not only their environment actions.

D Extended Related Work

Long-horizon and agentic evaluation. LLMs are increasingly used for long-horizon, open-ended tasks (Malone & Crowston, 1994; Du et al., 2023; Kwa et al., 2025; Yamada et al., 2025; Chan et al., 2025). To effectively evaluate these capabilities, recent benchmarks test models in complex environments that demand spatial reasoning and exploration (Paglieri et al., 2025; Zhang et al., 2026; Zhao et al., 2026; He et al., 2026). Procedurally generated tasks are especially useful for such evaluations because they reduce memorisation, mitigate contamination, and support scalable variation across tasks (Paglieri et al., 2025; Shojaee et al., 2026). ALEM brings these principles to a multi-agent setting with explicit coordination demands.

Multi-Agent LLM collaboration. The domains of code generation and advanced reasoning have already shown that employing multiple agents helps tackle complex tasks Hong et al. (2023); Du et al. (2023). This phenomenon has also been investigated from an RL perspective, and specialised benchmarks have been made or adapted to evaluate these capabilities in LLMs (Zhu et al., 2025; Sun et al., 2025; Grötschla et al., 2025). While these frameworks test communication and role specialisation, they typically enforce division of labour through hard role gates (Omari et al., 2025) and are restricted to shorter horizons and highly specified tasks. ALEM loosens the specialisation and organisation constraints and adapts these ideas to open-ended problems.

MARL and coordination. Existing MARL benchmarks typically isolate narrow segments of this spectrum: SMAC (Samvelyan et al., 2019) emphasises synchronous focus-fire, Overcooked (Carroll et al., 2019) tests short-window handovers, and Melting Pot (Agapiou et al., 2022) isolates long-range resource interdependence. ALEM builds on Craftax-Coop (Omari et al., 2025), but makes coordination explicit, procedural, and controllable: tasks are sampled across a coordination spectrum (Fig. 2); agents use soft rather than hard specialisation; communication is explicit; and difficulty can be scaled. Furthermore, we leverage information-theoretic diagnostics (Tessera et al., 2026) to verify that our environment demands genuine multi-agent interdependence (App. G.1).

E Limitations and Future Work.

Our evaluation focuses on text-based LLM agents, leaving Vision-Language Models (VLMs) and Claude-based LLMs for future work. While API costs limit evaluations of proprietary models (GPT, Gemini) to 10 seeds per difficulty, we will release the collected trajectories to support further analysis. Future work can use ALEM to study stronger agent harnesses, cross-episode memory, and lifelong learning in interactive worlds.

References

John P Agapiou, Alexander Sasha Vezhnevets, Edgar A Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, et al. Melting pot 2.0. *arXiv preprint arXiv:2211.13746*, 2022.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Saaket Agashe, Yue Fan, Anthony Reyna, and Xin Eric Wang. Llm-coordination: evaluating and analyzing multi-agent coordination abilities in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 8038–8057, 2025.

Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47: 253–279, 2013.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134 (1-2):57–83, 2002.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*, 2023.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:37567–37593, 2023.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations*.

Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. Overcookedv2: Rethinking overcooked for zero-shot coordination. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=hlvLM3GX8R.

Google DeepMind. Gemini 3.1 pro model card. https://deepmind.google/models/model-cards/gemini-3-1-pro/, February 2026a.

Google DeepMind. Gemma 4 model card. https://ai.google.dev/gemma/docs/core/model_card_4, April 2026b. Accessed: 2026-04-25.

Thomas Grady, Kip Parker, Iliyan Zarov, Henry Course, Anthony Hartshorn, Chengxi Taylor, and Ross Taylor. Kellybench: Can language models beat the market? 2026. URL https://openreward.ai/GeneralReasoning/KellyBench.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Florian Grötschla, Luis Müller, Jan Tönshoff, Mikhail Galkin, and Bryan Perozzi. Agentsnet: Coordination and collaborative reasoning in multi-agent llms. *arXiv preprint arXiv:2507.08616*, 2025.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: a survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 8048–8057, 2024.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1W0z96MFEoH.

Lewis Hammond, Alan Chan, Jesse Clifton, Jason Hoelscher-Obermaier, Akbir Khan, Euan McLean, Chandler Smith, Wolfram Barfuss, Jakob Foerster, Tomáš Gavenčiak, et al. Multi-agent risks from advanced ai. *arXiv preprint arXiv:2502.14143*, 2025.

Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pp. 709–715, 2004.

Muyu He, Adit Jain, Anand Kumar, Vincent Tu, Soumyadeep Bakshi, Sachin Patro, and Nazneen Rajani. $yc - bench$: Benchmarking ai agents for long-term planning and consistent execution. *arXiv preprint arXiv:2604.01212*, 2026.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Edward Hughes, Michael D Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge Shi, Tom Schaul, and Tim Rocktäschel. Position: Open-endedness is essential for artificial superhuman intelligence. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20597–20616. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/hughes24a.html.

Saman Kazemkhani, Aarav Pandya, Daphne Cornelisse, Brennan Shacklett, and Eugene Vinitsky. Gpudrive: Data-driven, multi-agent driving simulation at 1 million fps. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2408.01584.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, et al. Measuring ai ability to complete long tasks. *arXiv preprint arXiv:2503.14499*, 352, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

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

Dianbo Liu, Vedant Shah, Oussama Boussif, Cristian Meo, Anirudh Goyal, Tianmin Shu, Michael Curtis Mozer, Nicolas Heess, and Yoshua Bengio. Stateful active facilitator: Coordination and environmental heterogeneity in cooperative multi-agent reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=B4maZQLLW0_.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026.

Thomas W Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)*, 26(1):87–119, 1994.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2024.

Mariana Meireles, Niklas Lauffer, Rupali Bhati, and Cameron Allen. The influence of scaffolds on coordination scaling laws in LLM agents. In *Workshop on Scaling Environments for Agents*, 2025. URL https://openreview.net/forum?id=E9whrbtgUA.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Bassel Al Omari, Michael Matthews, Alexander Rutherford, and Jakob Nicolaus Foerster. Multi-agent craftax: Benchmarking open-ended multi-agent reinforcement learning at the hyperscale. *arXiv preprint arXiv:2511.04904*, 2025.

OpenAI. Introducing gpt-5.4. https://openai.com/index/introducing-gpt-5-4/, March 2026.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=fp6t3F669F.

Qwen Team. Qwen3.6-Plus: Towards real world agents, April 2026. URL https://qwen.ai/blog?id=qwen3.6.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian S de Witt, et al. Jaxmarl: Multi-agent rl environments and algorithms in jax. *Advances in Neural Information Processing Systems*, 37:50925–50951, 2024.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pp. 2186–2188, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems.

Thomas C Schelling. *The Strategy of Conflict: with a new Preface by the Author*. Harvard university press, 1980.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Parshin Shojaee, Seyed Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=YghiOusmvw.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The illusion of diminishing returns: Measuring long horizon execution in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3lm8lWYxiq.

Brian Skyrms. *The stag hunt and the evolution of social structure*. Cambridge University Press, 2004.

Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents. *arXiv preprint arXiv:1903.00784*, 2019.

Joseph Suarez, David Bloomin, Kyoung Whan Choe, Hao Xiang Li, Ryan Sullivan, Nishaanth Kanna, Daniel Scott, Rose Shuman, Herbie Bradley, Louis Castricato, et al. Neural mmo 2.0: a massively multi-task addition to massively multi-agent learning. *Advances in Neural Information Processing Systems*, 36:50094–50104, 2023.

Haochen Sun, Shuwen Zhang, Lujie Niu, Lei Ren, Hao Xu, Hao Fu, Fangkun Zhao, Caixia Yuan, and Xiaojie Wang. Collab-overcooked: Benchmarking and evaluating large language models as collaborative agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4922–4951, 2025.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Kale-ab Abebe Tessera, Arrasy Rahman, Amos Storkey, and Stefano V Albrecht. Hypermarl: Adaptive hypernetworks for multi-agent rl. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=56CgYnf9Dr.

Kale-ab Abebe Tessera, Leonard Hinckeldey, Riccardo Zamboni, David Abel, and Amos Storkey. Probing dec-POMDP reasoning in cooperative MARL. In *The 25th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Oral*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Wei Wang, Dan Zhang, Tao Feng, Boyan Wang, and Jie Tang. Battleagentbench: A benchmark for evaluating cooperation and competition capabilities of language models in multi-agent systems. *arXiv preprint arXiv:2408.15971*, 2024b.

Weixuan Wang, Dongge Han, Daniel Madrigal Diaz, Jin Xu, Victor Rühle, and Saravan Rajmohan. Odysseybench: Evaluating llm agents on long-horizon complex office application workflows. *arXiv preprint arXiv:2508.09124*, 2025.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Hanqing Yang, Jingdi Chen, Marie Siew, Tania Lorido Botran, and Carlee Joe-Wong. LLM-powered decentralized generative agents with adaptive hierarchical knowledge graph for cooperative planning. In *The First MARW: Multi-Agent AI in the Real World Workshop at AAAI 2025*, 2025. URL https://openreview.net/forum?id=l9QUw0oUTa.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Eric Ye, Ren Tao, and Natasha Jaques. An efficient open world environment for multi-agent social learning. *arXiv preprint arXiv:2508.15679*, 2025.

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

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, Chenxu Lv, and Junyang Lin. Deepplanning: Benchmarking long-horizon agentic planning with verifiable constraints. *arXiv preprint arXiv:2601.18137*, 2026.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, et al. Ama-bench: Evaluating long-horizon memory for agentic applications. *arXiv preprint arXiv:2602.22769*, 2026.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Daisy Zhe Wang, Zhenhailong Wang, Cheng Qian, Robert Tang, Heng Ji, et al. Multiagentbench: Evaluating the collaboration and competition of llm agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8580–8622, 2025.

Supplementary Materials

The following content was not necessarily subject to peer review.

F ALEM's Place Among Existing Environments

Table 2 compares ALEM with representative interactive multi-agent environments. We compare environments using four criteria. *Long horizon* means that success requires maintaining progress over extended trajectories, with rewards delayed by T timesteps where T is typically in the hundreds or thousands. *Bounded open-endedness* means that the environment supports a diverse but finite set of achievable goals $G = \{g_1, \ldots, g_n\}$. This differs from fully open-ended systems, where n can grow over time through the generation of novel and learnable goals (Hughes et al., 2024). *Explicit coordination requirements* means that the environment contains identifiable multi-agent task structures, such as synchronous actions, handovers, role allocation, or long-range interdependence. *Coordination difficulty control* means that the benchmark exposes an explicit parameter or protocol for scaling coordination difficulty.

Environment	Long horizon	Bounded open-endedness	Explicit coord. requirements	Coord. difficulty control
SMAC / SMACv2/ SMAX (?Ellis et al., 2023; Rutherford et al., 2024)	55	55	∼	55
Overcooked / Overcookedv2 (Carroll et al., 2019; Gessler et al., 2025)	55	55	∼	55
Hanabi (Bard et al., 2020)	55	55	∼	55
Melting Pot (Agapiou et al., 2022; Leibo et al., 2021)	∼	55	∼	55
HECOGrid (Liu et al., 2023)	55	55	51	51
GPUDrive (Kazemkhani et al., 2025)	55	55	55	55
Neural MMO (Suarez et al., 2019; 2023)	51	51	∼	55
Multi-Agent Craftax v1 (Ye et al., 2025)	51	51	∼	55
Craftax-Coop (Omari et al., 2025)	51	51	∼	55
ALEM	51	51	51	51

Table 2: **Comparison with representative multi-agent interactive environments.** 51indicates that the property is a central design feature, 55indicates that it is absent or not a primary design goal, and ∼indicates restricted support. For *long horizon*, restricted support means that only some scenarios require extended trajectories or delayed outcomes. For *explicit coordination requirements*, restricted support means that coordination-relevant interactions exist, but are fixed, implicit, narrow in type, or not procedurally generated as explicit task requirements. *Coordination difficulty control* requires an explicit parameter or protocol for scaling coordination difficulty, rather than only changing maps, scenarios, opponents, or task instances.

Existing environments cover important parts of this space, but not the full combination targeted by ALEM. Classic MARL benchmarks such as SMAC, Overcooked, and Hanabi isolate useful coordination motifs, but are short-horizon and task-specific. Open-world environments such as Neural MMO and Craftax-Coop provide longer horizons and bounded open-endedness, but do not expose explicit controls for coordination difficulty. ALEM combines these ingredients by adding procedurally generated coordination tasks, soft specialisation, communication, and a difficulty parameter that scales coordination demands while preserving the underlying world structure.

The closest environments are Multi-Agent Craftax v1 (Ye et al., 2025) and Craftax-Coop (Omari et al., 2025). Both inherit Craftax's long-horizon, bounded open-ended structure, but their coordination demands are mostly implicit. Multi-Agent Craftax v1 places multiple agents in the same open world, but does not introduce explicit coordination tasks, a controllable coordination-difficulty axis, or the full nine-level progression used in Craftax and ALEM (it is implemented only on the overworld). Craftax-Coop adds roles, creating long-range interdependence through specialisation. However, it does not procedurally generate coordination requirements over entities, require synchronous or handover coordination, provide an explicit communication channel, or expose a parameter that scales coordination difficulty while preserving the same world. ALEM adds these mechanisms directly: coordination tasks are sampled each episode, span synchronous and handover structures, vary the required number of agents, and are controlled by a single difficulty parameter.

G Environment Details

Figure 7: Example pixel-based observation in ALEM. The top bar shows teammate status (health, role icon), the centre is the agent's local view of the world, and the bottom panel is the agent's inventory and attributes (resources, tools, armour, potions, enchantments). ALEM also provides a symbolic observation (a structured array of nearby blocks, teammate states, and inventory tensors) and a text-based observation (natural-language descriptions of surroundings, status, teammates, and inventory) used by the LLM-agent baselines.

G.1 Probing Environment Changes

We use the information-theoretic diagnostics proposed in (Tessera et al., 2026) to check whether the design changes in ALEM introduce the intended behavioural demands. We roll out trained RL policies under the same four settings used in the environment calibration: a single-agent setting, a multi-agent setting without role pressure or coordination tasks, a multi-agent setting with soft specialisation, and full ALEM (EASY). The diagnostics are reported as behavioural questions in Table 3, with the corresponding metric acronym shown in parentheses.

All four diagnostic signals increase as multi-agent structure is added. This suggests that full ALEM induces stronger demands for memory use, hidden teammate information, synchronous coordination, and temporal coordination, rather than simply making the base game harder. We report diagnostic magnitudes rather than Wilcoxon tests because this is a single-environment calibration across settings, not a suite-level comparison across independent scenarios.

G.2 Coordination Tasks

G.2.1 Coordination Task Categories

We instantiate coordination across four activity categories:

- **Mining.** Resource blocks (including trees, ores, gems) are sampled for coordination during world generation. Synchronous mining requires the assigned number of agents to act on the same block in the same step. Handover mining requires a second agent to mine the block within $\Delta t \in [\Delta t_{\min}, \Delta t_{\max}]$ steps of the first agent.

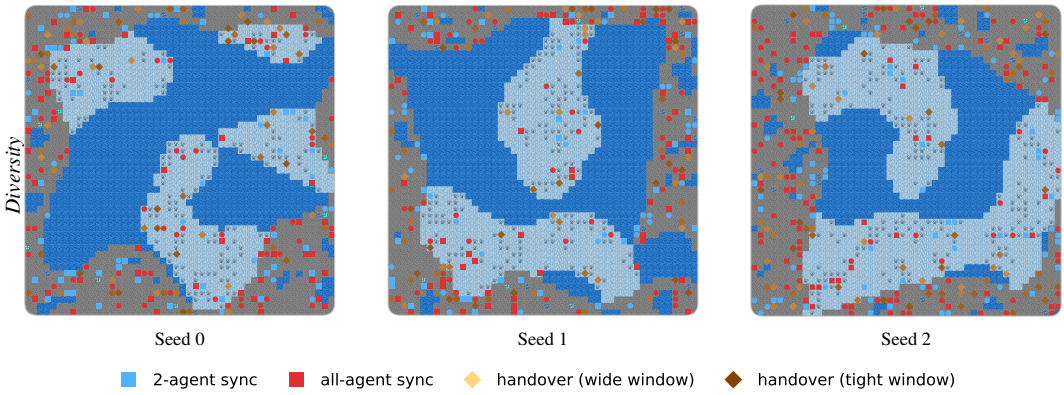


Figure 8: **ALEM procedurally generates diverse worlds and coordination layouts.** Three independently sampled episodes at the same medium difficulty show variation in both terrain layout and the spatial placement of coordination tasks (■ 2-agent sync, ■ all-agent sync, ♦ handover) vary across episodes.

Table 3: **Behavioural diagnostics across environment settings.** We use the diagnostics of (Tessera et al., 2026) on the same four settings used in the environment calibration: single-agent, multi-agent without role pressure or coordination tasks, multi-agent with soft specialisation, and full ALEM (EASY). Signals for memory use, hidden teammate information, synchronous coordination, and temporal coordination increase as these components are added. Higher values indicate stronger statistical dependence between agents' actions and the corresponding information source or coordination relationship, not higher return. Values are mean ± standard deviation over seeds.

Behavioural question	Single-agent	Multiple agent	MA + Soft spec.	ALEM
Do agents benefit from memory? (HAR)	0.355 ± 0.142	0.372 ± 0.022	0.437 ± 0.123	0.641 ± 0.064
Do agents use hidden teammate information? (PIF)	–	0.314 ± 0.036	0.384 ± 0.125	0.609 ± 0.148
Does synchronous coordination emerge? (AA)	–	0.017 ± 0.004	0.032 ± 0.011	0.061 ± 0.022
Does temporal coordination emerge? (DAI)	–	0.319 ± 0.057	0.406 ± 0.128	0.613 ± 0.165

- **Construction.** Construction sites for shelters, forges, and beacons are placed on valid overworld and mining-floor tiles, independently of mining blocks. Each site receives the same synchronous or handover structure as mining, together with a soft or hard synchronous variant. Building also requires the appropriate resources for the chosen structure. In handover construction, one agent initiates a specific structure, and a second agent must complete that structure type before the deadline, otherwise the site reverts and materials are refunded to the initiator.
- **Combat.** Mob slots are pre-labelled during world generation as standard or coordinated mobs. Coordinated mobs appear as either elite melee mobs, elite ranged mobs, or large passive mobs, and each is marked as soft or hard and assigned the required agent count.
- **Crafting.** Diamond pickaxes, swords, and armour require synchronous coordination, i.e. the craft succeeds only when the required number of agents attempt it at the same Epic Forge in the same step. This is a deliberately restricted variant, it is hard coordination only and tied to a specific tool station, and tests whether agents can converge on a shared location for a one-shot joint action.

G.3 Coordination Difficulty

Table 4 and 5 describe what exactly changes across different difficulty levels.

Table 4: Coordination difficulty parameters controlled by the scalar α. Easy, Medium, and Hard correspond to $\alpha \in \{0.3, 0.6, 0.9\}$. Larger α makes coordination harder to execute, but does not change how many coordination opportunities are sampled.

Parameter	Formula	Easy	Medium	Hard
$p_{\max}$	α	0.30	0.60	0.90
handover_window_min	$\max(3, \lceil 12(1-\alpha) \rceil)$	9	5	3
handover_window_max	$\max(6, \lceil 24(1-\alpha) \rceil)$	17	10	6
soft_solo_fail_prob	α	0.30	0.60	0.90
non_specialist_efficiency	$1-\alpha$	0.70	0.40	0.10
mob_health_multiplier†	$1+\alpha/3$	1.10	1.20	1.30
starting_resource_multiplier†	$1-\alpha/3$	0.90	0.80	0.70

† Only used when `scale_base_difficulty=True`. In the main benchmark, coordination difficulty can be scaled independently of base survival difficulty.

Table 5: Coordination opportunity parameters held fixed across difficulty levels. These control how much coordination exists in the world, rather than how hard it is to execute.

Parameter	Value
coordination_enabled	True
coordination_probability	0.25
soft_coordination_ratio	0.50
handover_ratio	0.25
hard_mob_probability	0.50
elite_mob_probability	0.15
large_passive_probability	0.20
construction_enabled	True
num_construction_sites	8
num_mining_construction_sites	4
soft_construction_ratio	0.50
crafting_coordination_enabled	True
diamond_crafting_agents_required	2
soft_specialization	True

H Detailed Experimental Results

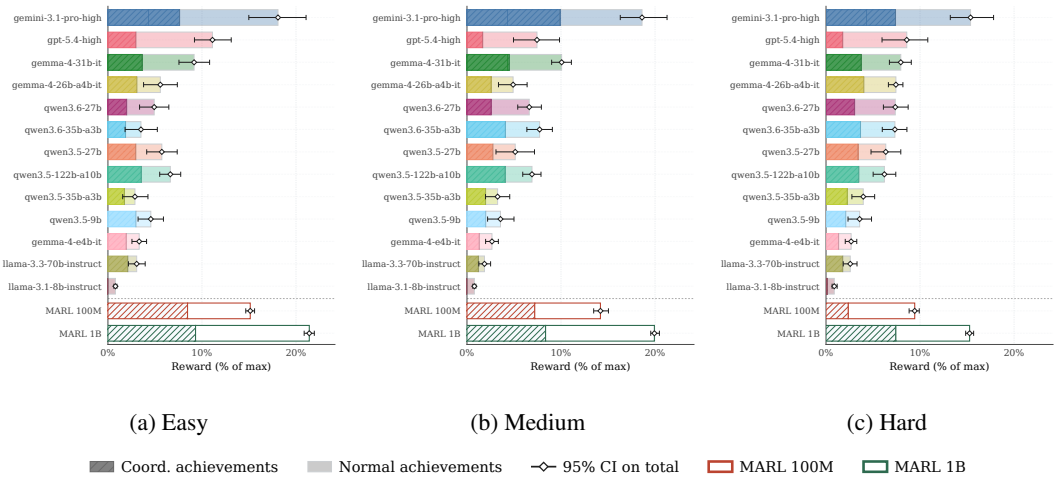


Figure 9: Zero-shot homogeneous coordination in ALEM. We evaluate modern LLMs on Easy, Medium, and Hard difficulties, and compare them against RL baselines at 100M and 1B environment steps. Solid bars denote base reward, dashed bars denote coordination reward, and diamonds denote total reward. Values are reported as percentages of the corresponding achievable maximum for each metric and difficulty, with 95% confidence intervals. Total% is separately normalized and should therefore be interpreted independently of Base% and Coord.%.

H.1 Algorithm Hyperparameters

The hyperparameters for the IPPO, MAPPO, HyperMARL, and PQN-VDN algorithms are summarized in Table 6. These parameters were standardized where applicable to ensure comparable evaluation across multi-agent reinforcement learning baselines.

Table 6: Hyperparameters for Multi-Agent Reinforcement Learning Algorithms.

Parameter	**IPPO**	**MAPPO**	**HyperMARL**	**PQN-VDN**
		Training & Environment		
Total Timesteps	10^9	10^9	10^9	10^9
Number of Environments	1024	512	1024	512
Steps per Environment	64	64	64	128
Number of Epochs	4	4	4	4
Number of Minibatches	8	8	8	16 ($4^\dagger$)
Learning Rate	2×10^{-4}	2×10^{-4}	1×10^{-4}	5×10^{-5}
Max Gradient Norm	1.0	1.0	1.0	0.5
Discount Factor (γ)	0.99	0.99	0.99	0.99
		Algorithm-Specific Optimization		
GAE / Eligibility λ	0.8	0.8	0.8	0.9
PPO Clip Ratio (ϵ)	0.2	0.2	0.2	–
Entropy Coefficient	0.01	0.01	0.01	–
Value Function Coefficient	0.5	0.5	0.5	–
ϵ-greedy Start	–	–	–	1.0
ϵ-greedy Finish	–	–	–	0.01 ($0.005^\dagger$)
ϵ-greedy Decay	–	–	–	0.1
		Network Architecture		
RNN Hidden Dimension	512	512	512	1024
FC Layer Dimension	128	128	128	–
Activation Function	`tanh`	`tanh`	`relu`	–
Normalization Type	–	–	–	Layer Norm
		HyperMARL-Specific		
Hypernet Embedding Dim	–	–	64	–
Hypernet Hidden Dims	–	–	[64]	–
Hypernet Init Scale	–	–	$\sqrt{2}$	–

† Denotes alternative configurations used in specific cooperative PQN variants.

Hyperparameter sweeps. To ensure a fair comparison, IPPO, MAPPO, and HyperMARL were tuned over an identical grid (Table 7, top), with 5 seeds per configuration. PQN-VDN was tuned separately over a normalisation, eligibility-trace, and learning-rate grid with 3 seeds per configuration (Table 7, bottom). All sweeps used `Alem-Coop-Symbolic` (PPO-family on the *Debug* variant at *hard* difficulty; PQN-VDN at *easy*), non-shared rewards, soft specialisation, and action masking. The configurations selected (in bold) are those used for the main results in Table 6.

Table 7: Hyperparameter sweep grids and selected values (**bold**). The PPO-family grid is shared across IPPO, MAPPO, and HyperMARL.

Hyperparameter	**Sweep range**
PPO-family (IPPO / MAPPO / HyperMARL), 5 *seeds per config*	
Learning rate IPPO/MAPPO - $\mathbf{2\times10^{-4}}$; HyperMARL - $\mathbf{1\times10^{-4}}$	$\{4\times10^{-3},\ 1\times10^{-3},\ 3\times10^{-4},\ 1\times10^{-4}\}$
PPO clip ratio ϵ	$\{0.1,\ \mathbf{0.2}\}$
Update epochs	$\{2,\ \mathbf{4}\}$
Max gradient norm	$\{0.5,\ \mathbf{1.0}\}$
PQN-VDN, 3 *seeds per config*	
Learning rate	$\{3\times10^{-4},\ 1\times10^{-4},\ \mathbf{5\times10^{-5}},\ 3\times10^{-5},\ 1\times10^{-5}\}$
Number of minibatches	$\{4,\ 8,\ \mathbf{16}\}$
Eligibility trace λ	$\{0.5,\ 0.7,\ \mathbf{0.9}\}$
Normalization type	{layer_norm, batch_norm}
Normalize inputs	{true, **false**}
ϵ-greedy finish	$\{\mathbf{0.01},\ 0.005\}$
Number of epochs	$\{\mathbf{4},$ others$\}$
Recurrent cell	{**LSTM**, GRU}

The PQN-VDN sweep additionally revealed two non-default choices worth flagging: (i) although the original PureJaxQL Craftax recipe enables input normalization, this consistently degraded performance in our setting and was disabled; and (ii) GRU was substantially worse than LSTM as the recurrent cell, despite GRU being used by all PPO-family baselines. The cooperative PQN variant († in Table 6) used 4 minibatches and ϵ-finish $= 0.005$.

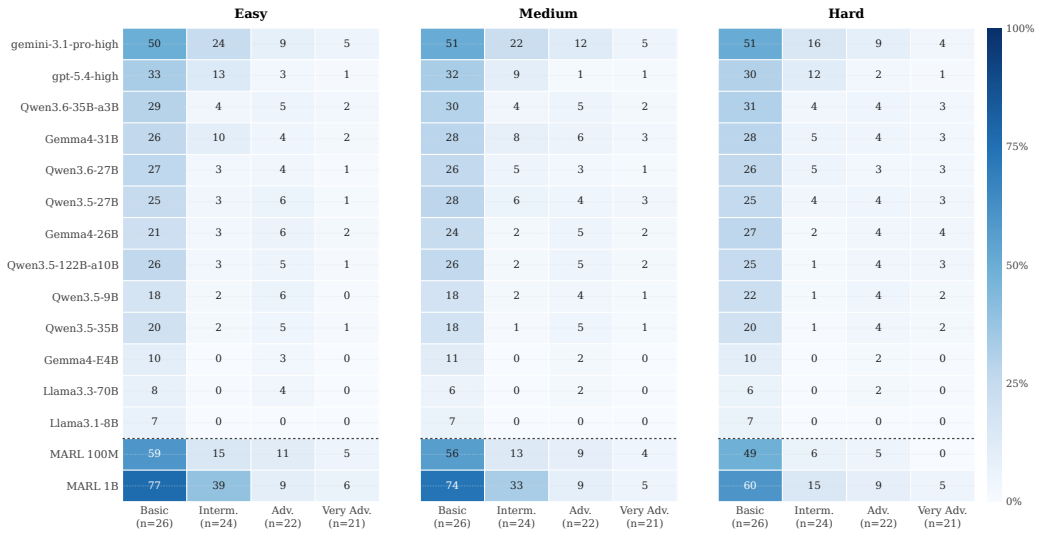


Figure 10: Achievement-tier coverage across coordination difficulty. Each cell reports the mean fraction of achievements completed within a tier by a method; columns group achievement tiers and panels separate coordination settings. Cell values are percentages and colour encodes coverage from 0% to 100%. MARL baselines are separated by a dashed rule, and methods are ordered by their total achievement count on the hard coordination setting.

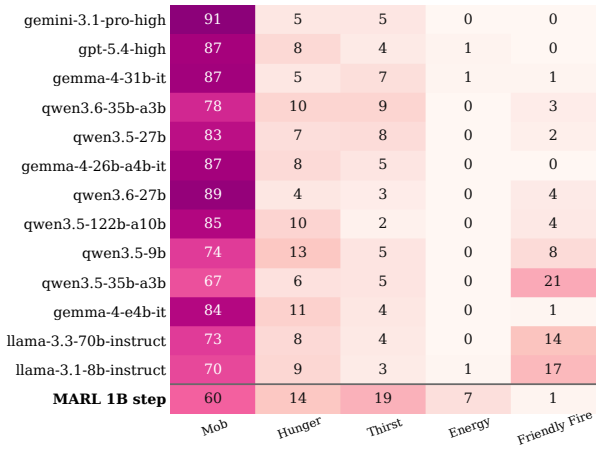


Figure 11: Different causes of death for different agents, averaged across easy, medium and hard.

H.2 Additional Metrics

Our primary metric is normalised episode return. For each agent, we divide cumulative reward by the maximum reward available in the relevant reward category, then average across agents. We report coordination return normalised by $R_{\max}^{\text{coord}} = 159$, base return by $R_{\max}^{\text{base}} = 217$, and total return by $R_{\max}^{\text{total}} = 376$.

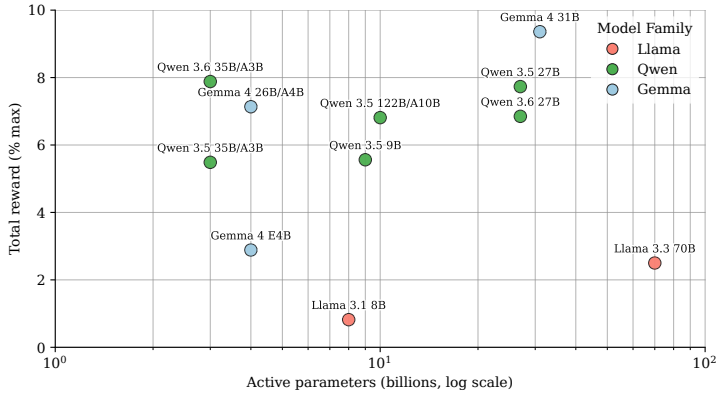


Figure 12: Performance vs active model size.

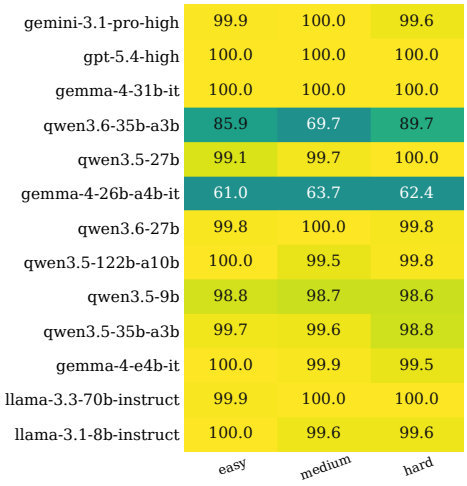


Figure 13: Percentage success rate for parsing actions, averaged across easy, medium and hard.

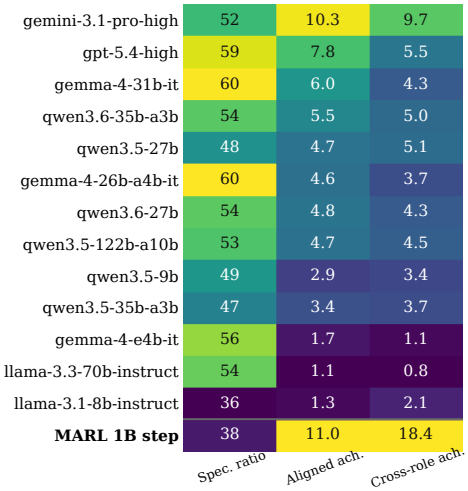


Figure 14: Specialisation metrics, averaged across easy, medium and hard.

H.3 Environment Speed

We benchmark ALEM's simulation speed against two baselines: Craftax-Coop (Omari et al., 2025), the multi-agent environment we extend, and Craftax (Matthews et al., 2024), the original single-agent JAX environment. Measurements report steps per second (SPS) for a full IPPO training iteration (rollouts and updates, excluding logging) on a single NVIDIA L40S (48 GB) GPU using JAX 0.4.38. ALEM is evaluated under its default training configuration (Easy difficulty, four communication channels).

Figure 15 shows throughput scaling efficiently across 1 to 8 agents. The throughput reduction compared to Craftax-Coop directly reflects ALEM's expanded mechanics: a richer observation space (coordination tracking and messages), coordination dynamics, additional tasks (building shelter, new passive mobs), soft specialiation, larger action space, and coordination reward logic. Crucially, the environment remains fully JIT-compiled; a 1B-timestep multi-agent training run completes in under two days on a single GPU, ensuring that large-scale experimentation remains highly accessible.

For researchers prioritising execution speed, we also provide ALEM-LITE. This provides the exact same coordination logic, but limits the environment to one level (the overworld), as opposed to the full nine levels. This offers the highest speed across all tested environments.

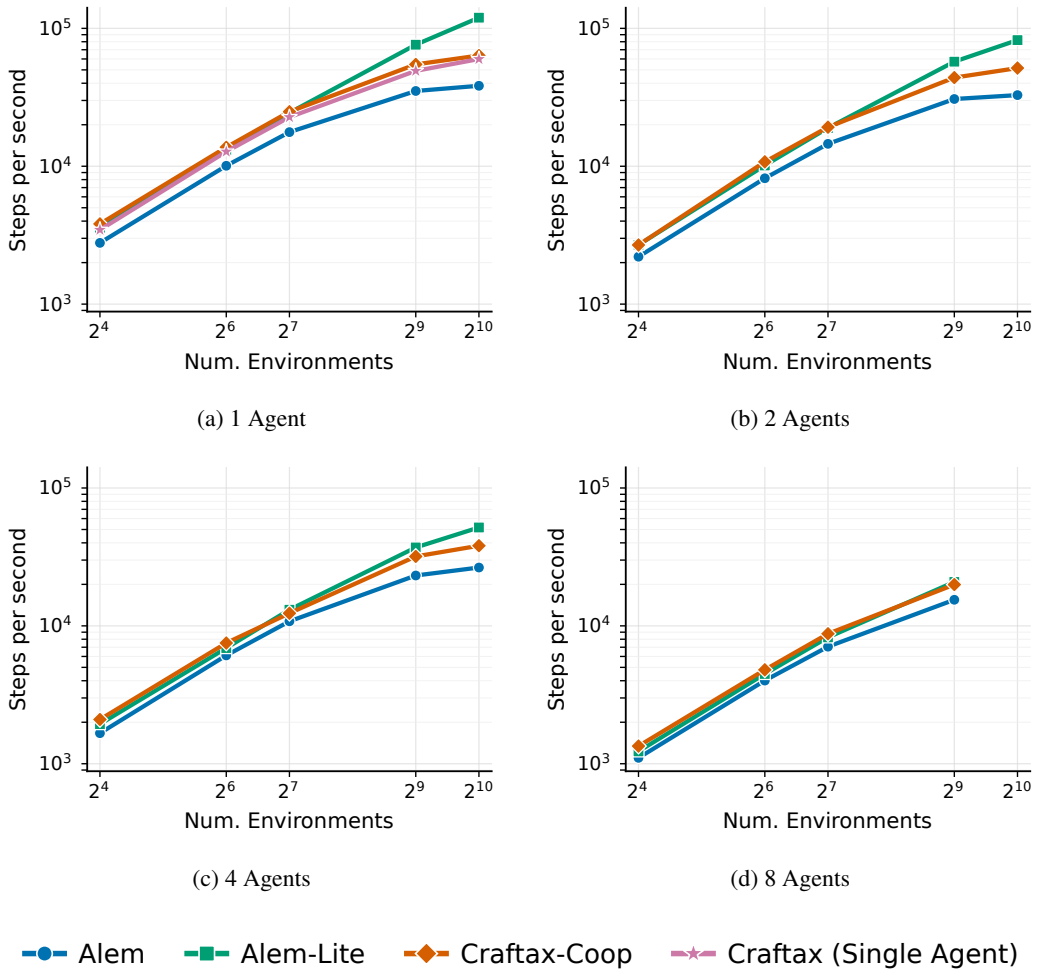
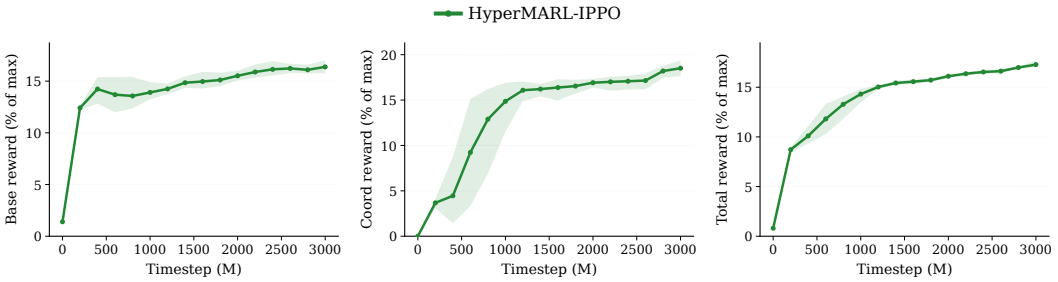


Figure 15: Simulation throughput (SPS) during a full IPPO training step across 1–8 agents. The single-agent Craftax baseline is included in all panels as a global reference.

Our LLM experiments were carried out on 3 NVIDIA A100 80GB devices and took between 10 and 30 hours per 20 seeds per difficulty level for each model.



(a) Base reward (b) Coordination reward (c) Total reward

Figure 16: Extended MARL ablation on hard ALEM. HyperMARL-IPPO is trained for 3B environment steps across five seeds. Curves show seed means and shaded bands show 95% bootstrap confidence intervals. Performance does not saturate: coordination and total reward continue improving late in training, and total reward remains well below the maximum achievable score.

H.4 Extended MARL Training

To test whether ALEM's hard coordination setting is solved simply by running MARL for longer, we extend the strongest RL baseline, HyperMARL-IPPO, to 3B environment steps with five seeds. Figure 16 shows that the environment is not saturated: base reward improves early and then plateaus, while coordination and total reward continue to climb steadily through the end of training. Even after 3B steps, total reward remains far from the maximum achievable score, indicating that the hard setting retains substantial headroom for stronger learning algorithms.

H.5 Text-based Language Interface Details

Prompt Structure. At each timestep t, the prompt for agent i is assembled from three components: a system prompt, a history of recent observations and actions, and the current observation with instructions.

1. **System prompt:** The system prompt is shared across all time steps for an agent and defines the agent identity and role, the team size, the objective, and the rules needed to act in the environment. By default it also optionally includes the full action catalogue with natural-language descriptions, game and coordination mechanics.
2. **Observation and Action History:** The observation and action history is a rolling sequence of recent messages from the environment and the agent itself. The observation messages contain previous observations, labelled as "Observation from k step(s) ago", and the action messages contain the action previously taken by the agent. When memory and communication is enabled, the observation history also includes scratchpad notes and teammate messages, each labelled with the corresponding timestep.
3. **Current Observation and Action Space:** The current observation message contains textual descriptions of: achievement progress, current level, nearby terrain, items and enemies, visible coordination opportunities and requirements, teammate status, agent stats, vitals and inventory. The agent is then prompted to respond with an action. By default, the call to action prompt also includes current legal action affordances, instructions to think step-by-step before acting, to optionally broadcast a message, and to optionally write a private scratchpad note.

The length of observation, communication, and memory history is independently configurable, and the system prompt can be ablated to control how much information about the game and coordination mechanics is disclosed to the model. The default setting includes all information in the system prompt and current observation with communication and scratchpad history enabled, but we ablate these components to probe their effects on LLM performance in 3.2.2.

Prompt Disclosure Levels. In order to reduce the amount of information given to the model, we only reveal specific gameplay information as the team reaches the relevant level.

In order to probe the effect of environment information and coordination cues on LLM performance, we ablate the amount of game and coordination information given to the model. This is distinct from the underlying environment coordination difficulty. The default setting is as described in the Prompt Structure section above. The first ablation level removes the coordination-specific system prompt information and the current coordination cues from the observation, leaving only the general game mechanics and rules. The second ablation level builds on the first level and additionally removes all environment-specific information from the system prompt such as game rules and mechanics, leaving only the task introduction, action descriptions, and achievement list.

Coordination Information. The current observation includes explicit natural language descriptions of the current coordination targets and requirements. For synchronous targets, the prompt states whether the target works solo with a bonus or requires N agents simultaneously, what the required action is, and which side of the target the acting agent currently occupies. The location and vitals of teammates are includes as well as their previous messages.

Action Parsing and Execution. The model response is instructed to respond in a structured format containing exactly one action tag, optionally preceded by a think tag for reasoning, communication tags for teammate messages, and scratchpad tags for private notes. The required action is parsed from the response and must match one of the legal actions. If parsing fails, the agent is re-prompted with a format error. If the model still fails to produce a valid action after the configured retry budget is exhausted, the action defaults to NOOP. The final parsed action is mapped back to the underlying discrete action index before stepping the environment.

H.6 Rewards and Achievements

Table 8: Complete list of all 93 achievements available in the environment, sorted alphabetically.

Achievement	**Achievement**	**Achievement**	**Achievement**
Cast Spell	Coord Mine Coal Hard	Defeat Necromancer	Handover Complete
Collect Coal	Coord Mine Coal Soft	Defeat Orc Mage	Learn Spell
Collect Diamond	Coord Mine Diamond Hard	Defeat Orc Solider	Make Arrow
Collect Drink	Coord Mine Diamond Soft	Defeat Pigman	Make Diamond Armour
Collect Food	Coord Mine Handover	Defeat Skeleton	Make Diamond Pickaxe
Collect Iron	Coord Mine Iron Hard	Defeat Troll	Make Diamond Sword
Collect Ruby	Coord Mine Iron Soft	Defeat Zombie	Make Iron Armour
Collect Sapling	Coord Mine Ruby Hard	Drink Potion	Make Iron Pickaxe
Collect Sapphire	Coord Mine Ruby Soft	Eat Bat	Make Iron Sword
Collect Stone	Coord Mine Sapphire Hard	Eat Cow	Make Stone Pickaxe
Collect Wood	Coord Mine Sapphire Soft	Eat Plant	Make Stone Sword
Coord 2 Agents Hard	Coord Mine Stone Hard	Eat Snail	Make Torch
Coord 2 Agents Soft	Coord Mine Stone Soft	Enchant Armour	Make Wood Pickaxe
Coord 3 Agents Hard	Damage Necromancer	Enchant Sword	Make Wood Sword
Coord 3 Agents Soft	Defeat Archer	Enter Dungeon	Open Chest
Coord Build Beacon	Defeat Deep Thing	Enter Fire Realm	Place Furnace
Coord Build Forge	Defeat Fire Elemental	Enter Gnomish Mines	Place Plant
Coord Build Shelter	Defeat Frost Troll	Enter Graveyard	Place Stone
Coord Diamond Armour	Defeat Gnome Archer	Enter Ice Realm	Place Table
Coord Diamond Pickaxe	Defeat Gnome Warrior	Enter Sewers	Place Torch
Coord Diamond Sword	Defeat Ice Elemental	Enter Troll Mines	Wake Up
Coord Elite Melee Kill	Defeat Knight	Enter Vault	
Coord Elite Ranged Kill	Defeat Kobold	Find Bow	
Coord Large Passive Kill	Defeat Lizard	Fire Bow	

H.7 Prompt Examples

System prompt given to the LLM

```
system

You are Agent 1 (forager) in a 3-agent cooperative survival game. Your goal is to gather
     resources, craft gear, fight monsters, and descend through 9 dungeon levels, while
     coordinating with teammates. You must survive -- if your health reaches zero, you die
     , and if all agents die the game ends. Maximize the number of achievements while
     staying alive.

<game_rules>

## How to play
- Each turn, choose exactly one action.
- **Movement** uses absolute directions: north, south, east, and west. Any move attempt
     changes your facing to that direction, even if the move is blocked and you stay in
     place. A move is blocked if the target tile is solid, including trees, stone, ore
     veins, walls, crafting stations, chests, and plants, or if it contains water, lava, a
      mob, or another player. If repeated move attempts do not change your position, that
     direction is blocked. You can also use a blocked move to turn in place, for example
     to face an adjacent tree.
- **Facing**: your facing direction is set by your last movement action and persists until
      you move again. **Do** always targets the tile in your current facing direction.
- **Do** is your main interaction: face a tile and use the **Do** action on exactly that
     tile to chop trees, mine ore, attack creatures, drink water, open chests, or revive a
      downed teammate. If the faced tile contains a downed teammate, Do revives them. If
     the faced tile contains a living teammate, Do targets that teammate instead and can
     cause friendly fire. For synchronous-style coordination, all required agents must
     stand next to the same target tile, face it, and act together.
- **Crafting**: stand next to (including diagonally) the required station and use the
     craft action; you do NOT need to face it. Diamond items always require an adjacent
     epic forge, not a table.
- **Placing**: face the target tile, then use the place action. Tables and furnaces need
     an empty non-solid tile that is not water or lava; stone can also be placed into
     water (costs 1 stone). Place Plant puts a sapling on the faced tile. Place Torch
     lights dark areas.
- **Ranged combat**: use Shoot Arrow while facing a creature (requires a bow + arrows).
     Bows are found in dungeon chests.
- **Elite mobs** are tougher and deal more damage; coordinating with teammates (multiple
     agents attacking together) makes them much easier to defeat.
- **Request/Give**: use Request [Resource] to broadcast a resource request to teammates
     for 10 turns; teammates can use Give to Agent X to transfer one unit of the requested
      resource directly -- no adjacency required, works at any distance. Give only appears
      as an available action when a teammate has an active Request.

## Survival stats
Food, drink, and energy deplete gradually over time -- roughly every 20-30 steps you lose
     1 point of each (dexterity slows this rate). When food or drink reaches 0, your
     health starts dropping. When energy reaches 0, you automatically fall asleep and
     cannot act until energy is full. While sleeping, you take 2.5x damage from all
     sources. Mana does NOT decay -- it is only spent by casting spells or enchanting.
     Mana slowly regenerates over time (faster while sleeping).
- **Sleep**: choose this voluntarily to recover energy at 2x the passive rate. Ends
     automatically when energy is full.
- **Rest**: choose this to recover health gradually. Requires food, drink, and energy all
     > 0; ends when health is full or a stat runs out.

## Roles
Role-restricted actions succeed with reduced probability for non-specialists. Depending on
      the difficulty configuration, non-specialist success rates are 10%, 40%, or 70%.
     Specialist success rate is always 100%.

- **Forager**: collecting water, saplings, eating passive mobs (e.g. cows/bats/snails).
     Also has 3x base food and drink capacity.
- **Miner**: crafting pickaxes/torches, placing stone.
- **Warrior**: crafting swords and arrows. Also deals 2x melee damage, and specializes in
     enchanting swords and bows.
- No role restriction: Place Table, Place Furnace, Wood Sword, Iron Armour, Diamond Armour

## Coordination
Some tasks, tiles, creatures, and structures require multiple agents. When collaboration
     is possible the observation will include a short coordination hint. Follow the rules
     below to coordinate safely and efficiently.

- **Sync**: N agents must each stand on a different tile adjacent to the shared target,
     each facing it, and all choose the **Do** action on the same turn. Approach from
```

```
    different sides so every agent targets the shared tile directly. If a teammate is
    between you and the target, your **Do** will hit the teammate instead and can cause
    friendly fire.
- **Handover**: one agent starts the task with **Do** (pays the resources required), and
    another agent finishes it with **Do** within a small time window shown in the
    observation. If no agent completes it in time, the site resets and materials are
    refunded to the initiator. Follow the exact handover timing when specified.
- **Construction**: Construction sites (shelters, forges, beacons) may require either sync
     or handover. Always follow the coordination rule shown in the observation for that
    site.
- **Elite mobs**: stronger enemies may benefit from or require coordinated attacks. Attack
     from different sides, avoid standing between a teammate and the mob, and avoid
    blocking another agent's attack.
- **Revive**: Face a downed teammate and use **Do** to revive them.
- **Epic forge / Diamond crafting**: Diamond-tier items require multiple agents to craft
    simultaneously at an adjacent epic forge. All required agents must choose the
    crafting action for the same item on the same turn while adjacent to the forge.

## Resource chain
Trees -> wood (no tool required) -> Stone/Coal (needs wood pickaxe) -> Iron (needs stone
    pickaxe) -> Diamond (iron pickaxe) -> Ruby/Sapphire (diamond pickaxe)

## Crafting recipes
All recipes consume the listed materials.
Stations: Table (2 wood), Furnace (1 stone)
- Wood pickaxe/sword: table + 1 wood
- Stone pickaxe/sword: table + 1 wood + 1 stone
- Iron pickaxe/sword: table + furnace + 1 wood + 1 stone + 1 iron + 1 coal
- Iron armour: table + furnace + 3 iron + 3 coal
- Diamond pickaxe: epic forge + 1 wood + 3 diamond + enough agents crafting the same item
    there on the same turn
- Diamond sword: epic forge + 1 wood + 2 diamond + enough agents crafting the same item
    there on the same turn
- Diamond armour: epic forge + 3 diamond + enough agents crafting the same item there on
    the same turn
- Arrows: table + 1 wood + 1 stone (yields 2)
- Torch: table + 1 wood + 1 coal (yields 4)

Construction (at a construction site, face it and use Build action):
- Build Shelter: needs 10 wood + 5 stone. Shelters result in +50% energy regeneration
    while resting (all agents).
- Build Forge: needs 10 stone + 3 iron + 2 coal. Creates an epic forge, which enables
    diamond gear crafting.
- Build Beacon: needs 3 iron + 2 coal. Expands the lit area on this level.

## Attributes
Gain 1 XP each time you descend to a new floor. Spend XP with Level Up actions.
- **Strength**: max health = 8 + strength
- **Dexterity**: max food = 7 + 2*dexterity (+2 extra for foragers); max drink = same; max
     energy = 7 + 2*dexterity
- **Intelligence**: max mana = 6 + 3*intelligence; enchantment damage +5% per point above
    1.

## Progression
1. Gather wood -> place a table -> craft a wood pickaxe; craft a wood sword early if
    combat is likely.
2. Mine stone and coal -> place a furnace -> craft iron tools and iron armour.
3. To descend: stand on the `ladder_down` tile (visible in your observation when close)
    and use the Descend action. The ladder only becomes usable after enough monsters on
    that level have been killed. Only one agent needs to use Descend/Ascend -- all
    teammates are teleported with them.
</game_rules>

<achievements>
## Achievements
Collect Wood
Place Table
Eat Cow
Collect Sapling
Collect Drink
Collect Food
Make Wood Pickaxe
Make Wood Sword
Place Plant
Defeat Zombie
Collect Stone
Place Stone
Eat Plant
Defeat Skeleton
```

```
Make Stone Pickaxe
Make Stone Sword
Wake Up
Place Furnace
Collect Coal
Collect Iron
Collect Diamond
Make Iron Pickaxe
Make Iron Sword
Make Arrow
Make Torch
Place Torch
Make Diamond Pickaxe
Make Diamond Sword
Make Iron Armour
Make Diamond Armour
Enter Gnomish Mines
Coord 2 Agents Soft
Coord Large Passive Kill
Coord Mine Stone Soft
Coord Mine Stone Hard
Coord Mine Coal Soft
Coord Mine Coal Hard
Coord 2 Agents Hard
Coord 3 Agents Soft
Handover Complete
Coord Mine Handover
Coord Mine Iron Soft
Coord Mine Iron Hard
Coord Mine Diamond Soft
Coord Mine Diamond Hard
Coord Build Shelter
Coord 3 Agents Hard
Coord Build Forge
Coord Build Beacon
Coord Diamond Pickaxe
Coord Diamond Sword
Coord Diamond Armour
Coord Elite Melee Kill
Coord Elite Ranged Kill
</achievements>

<output_format>
Each turn you receive an observation showing what you see, your inventory, teammates, and
     available actions.
Think first, then output strictly in the following format:
1. (Required) Exactly one action from the available action list:
<action>YOUR_CHOSEN_ACTION</action>
2. (Optional) Broadcast to teammates, up to 400 chars. Teammates can only act on what you
     tell them. Be specific (e.g. 'Dig on tree next turn', 'Ladder at 5NE', 'Need 2 wood')
     . Reply to teammates' requests.
<communication>YOUR_MESSAGE</communication>
3. (Optional) Private notes, up to 1000 chars -- not shared with teammates. Your context
     resets each turn -- this is your only memory. Don't repeat what's already in your
     observation; store what you'll need later. Record teammates' plans and any facts you'
     ll need after they scroll out of view.
<scratchpad>YOUR_NOTES</scratchpad>
Important: every tag you open must be closed (e.g. <communication>...</communication>).
Token budget: 8192 tokens for your full response (including reasoning). Keep reasoning
     concise and stop thinking early enough to emit every required tag -- if you exhaust
     the budget mid-reasoning, no action is produced and your turn fails.
</output_format>
```

Example observation given to the LLM

```
The LLM receives the last 8 observation-action turns as context. Below we show the current
      observation from one turn.

Step: 0/10000 (10000 remaining, ends early if all agents die)
Current Observation:
Position: (x=25, y=24)
Role: forager
Location: Overworld (surface)
Achievements: 0/93 (39 unlock later)

Level info:
```

```
- Light: bright (0.80)
- Level: cleared -- you can find the ladder down tile and use Descend to go deeper.

You see:
- tree 3 steps north (x=25, y=21)
- construction_site 1 step north (x=25, y=23)
- water 5 steps east (x=30, y=24)

Facing: north.
Do target: construction_site (x=25, y=23).

Coordination:
- construction_site 1 step north (x=25, y=23): requires 2 agents to use a Build action
    simultaneously (fails alone). You are on the south side, adjacent (facing target).
- construction_site 1 step west (x=24, y=24): works solo but grants a bonus when 3 agents
    use a Build action simultaneously. You are on the east side, adjacent (facing north).
- tree 1 step south and 2 steps west (x=23, y=25): works solo but grants a bonus when 2
    agents select Do simultaneously. You are on the north-east side, 3 steps away.
- tree 2 steps north and 2 steps east (x=27, y=22): requires 3 agents to select Do
    simultaneously (fails alone). You are on the south-west side, 4 steps away.

Teammates:
Agent 0 (warrior): 1 step west (x=24, y=24), health=9
Agent 2 (miner): 1 step south and 1 step west (x=24, y=25), health=9

Your status:
- health: 9
- food: 9
- drink: 9
- energy: 9
- mana: 9
- xp: 0

Attributes:
- dexterity: 1
- strength: 1
- intelligence: 1

You have nothing in your inventory.

Available actions:
 - Noop
 - Move West
 - Move East
 - Move North
 - Move South
 - Do
 - Sleep
 - Rest
 - Request Food
 - Request Drink
 - Request Wood
 - Request Stone
 - Request Iron
 - Request Coal
 - Request Diamond
 - Request Ruby
 - Request Sapphire

---
Response format (in this order): <action> , <communication> , <scratchpad>. Close every
    tag you open.
```

I MARL Baselines

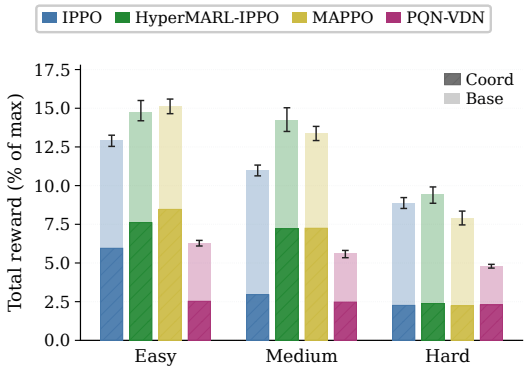
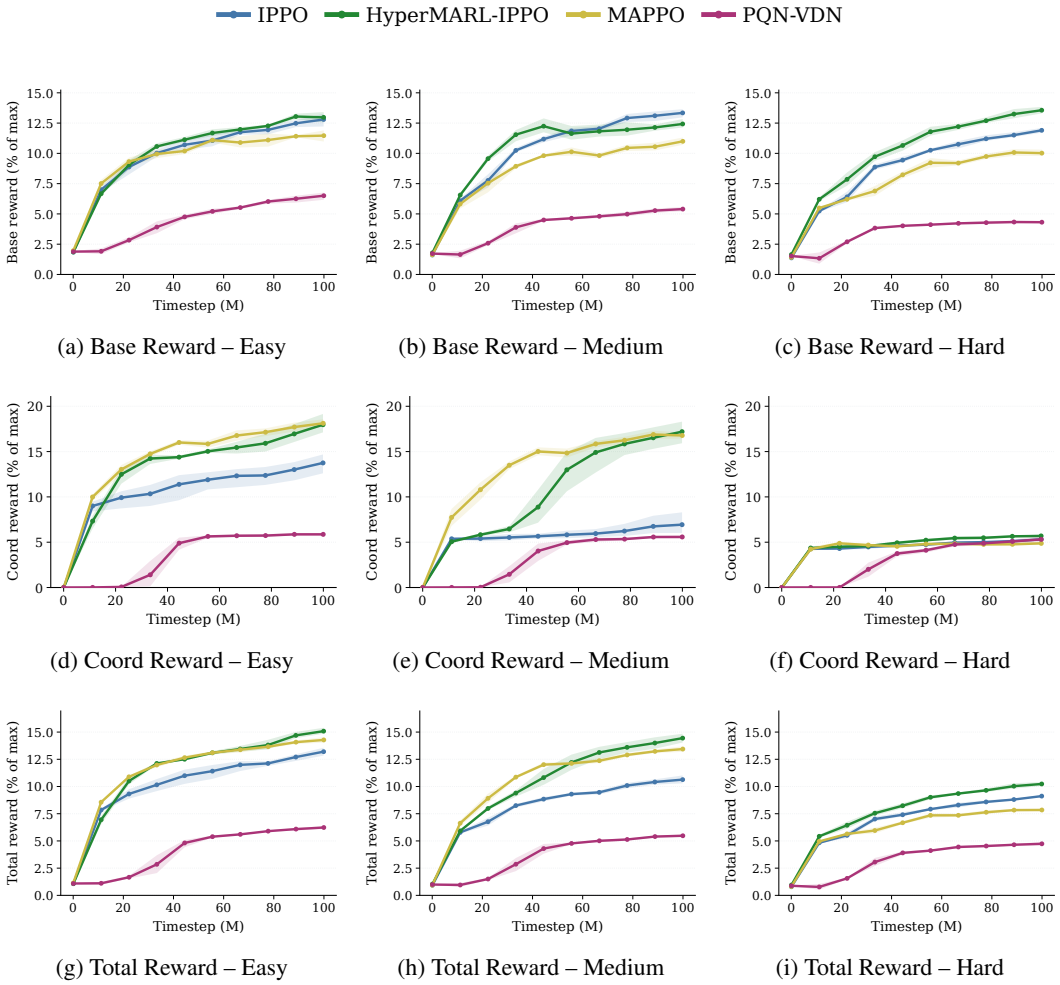


Figure 17: **MARL 100m Results.** MARL baseline performance, reported as mean percentage of max achievable reward with 95% CIs and decomposed into coordination (dark) and base (light) reward.

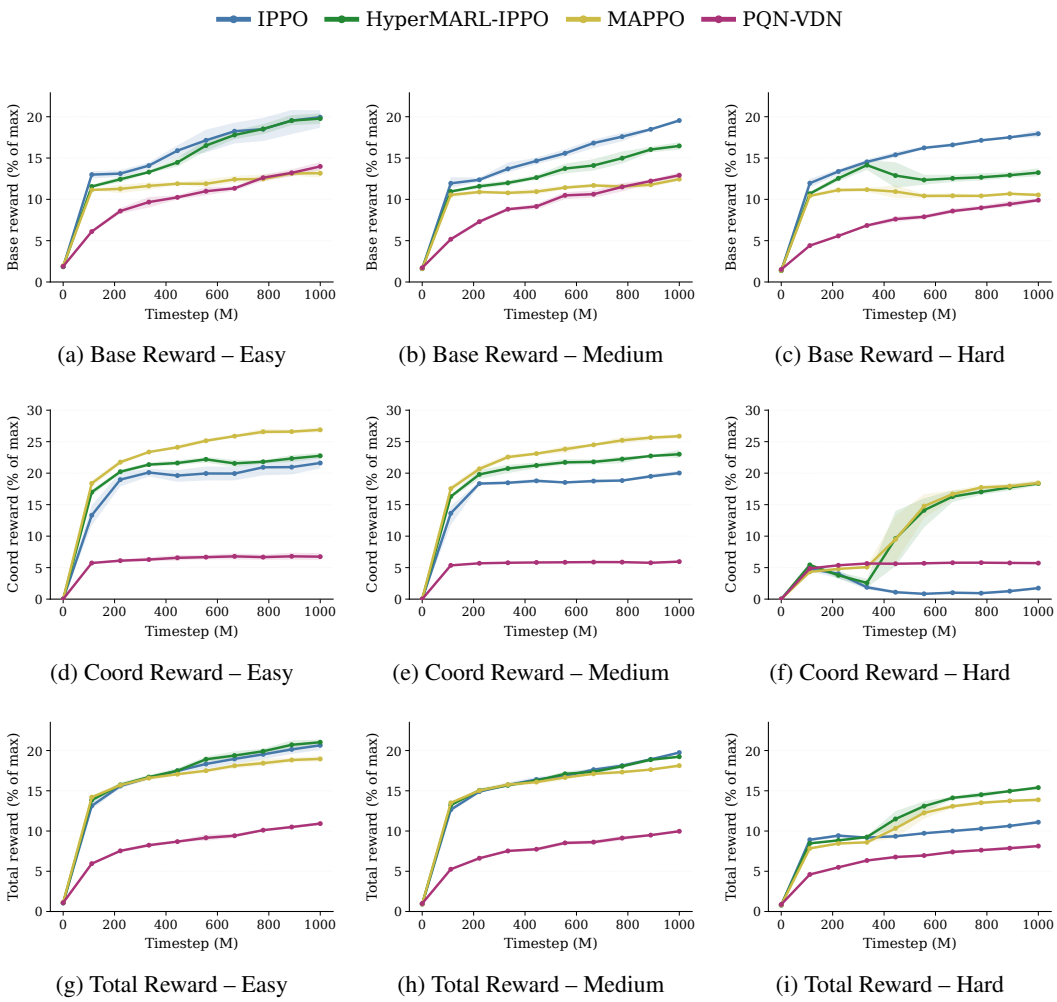


(a) Base Reward – Easy (b) Base Reward – Medium (c) Base Reward – Hard

(d) Coord Reward – Easy (e) Coord Reward – Medium (f) Coord Reward – Hard

(g) Total Reward – Easy (h) Total Reward – Medium (i) Total Reward – Hard

Figure 18: **Learning curves during training for MARL baselines across coordination difficulty, trained for 100m steps.** Individual panels (a)-(i) break down performance by environment difficulty (Easy, Medium, Hard) and reward type (Base, Coordination, Total). Curves show the mean across independent seeds; shaded regions are 95% bootstrap confidence intervals. Base and coordination scores are normalised by their category-specific achievable maxima, while total reward is normalised by the full achievable reward.



(a) Base Reward – Easy (b) Base Reward – Medium (c) Base Reward – Hard

(d) Coord Reward – Easy (e) Coord Reward – Medium (f) Coord Reward – Hard

(g) Total Reward – Easy (h) Total Reward – Medium (i) Total Reward – Hard

Figure 19: **Learning curves during training for MARL baselines across coordination difficulty, trained for one billion steps.** Individual panels (a)-(i) break down performance by environment difficulty (Easy, Medium, Hard) and reward type (Base, Coordination, Total). Curves show the mean across independent seeds; shaded regions are 95% bootstrap confidence intervals. Base and coordination scores are normalised by their category-specific achievable maxima, while total reward is normalised by the full achievable reward.

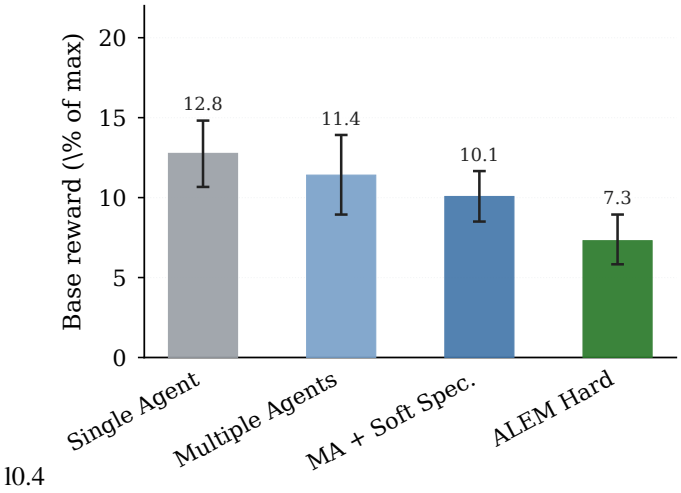


Figure 20: **Environment calibration.** Base reward for Gemma-4-31B-it across settings sharing the same underlying world. Cooperative achievements are excluded so all settings are directly comparable. Error bars show 95% bootstrap CIs.

J Additional LLM Experiments and Ablations

J.1 Environment calibration: multi-agent structure adds difficulty beyond the base game.

Before evaluating full ALEM, we isolate how much difficulty is introduced by the multi-agent structure beyond the underlying open-ended environment dynamics. We evaluate Gemma-4-31B-it in four settings that share the same base world: a single-agent setting, a multi-agent setting without role pressure or coordination tasks, a multi-agent setting with soft specialisation, and full ALEM at $\alpha = 0.9$ (HARD).

Fig. 20 shows a steady decline in Base reward, the only reward category directly comparable across all settings: 12.8% in the single-agent setting, 11.4% with multiple agents, 10.1% with soft specialisation, and 7.3% in full ALEM. The key comparison is the single-agent setting versus full ALEM: a 43% relative drop on the same underlying world. This calibration suggests that ALEM's difficulty is not reducible to the base game. Adding agents changes the dynamics as each agent's transition depends on the actions of the others, soft specialisation adds role-allocation pressure, and full ALEM adds explicit coordination requirements. We do not over-interpret the intermediate steps, since adjacent confidence intervals overlap and this calibration uses one open-weight model. Instead, the result serves as a sanity check that full ALEM introduces multi-agent failure modes absent from single-agent progression.

K Qualitative Analysis

Our qualitative analysis of the LLM agents' memory, reasoning and communication can be found under sections L.3, M.3 and N.3 respectively.

L Memory entry usage: detailed analysis

L.1 Setup

We compare how Gemini 3.1 Pro and Gemma use the persistent memory field across episodes. This is the same trajectory set used in the communication analysis: 10 Gemini episodes (34,080 agent-step records) and 20 Gemma episodes (24,747 records). Each record contains the persisted

`scratchpad` string, which we refer to as the agent's *memory entry*, together with the parsed action and local observation. Parse failures are negligible, under 0.5% for both models.

Lexical detectors. Structural features, such as character counts and line counts, are computed directly. The remaining features in Table 9 are simple regular-expression detectors over the raw memory-entry text. These detectors are intended as interpretable proxies rather than full discourse parsers. We define them as follows:

- **Multi-line memory entry (≥ 2 lines).** The memory entry contains at least one newline.
- **Turn-indexed plan.** The memory entry has at least two lines whose first non-whitespace tokens match "`T` N `:`" or "`Turn` N `:`" for an integer N. This captures explicit turn-by-turn plans, such as "*T9: Do (completes handover . . .)*".
- **Compact state summary.** The memory entry contains at least two "`Field: value`" fields drawn from a fixed vocabulary of status labels (*Pos, Position, HP, Health, Inv, Inventory, Energy, Hunger, Thirst, A1, A2, Agent*). This captures compact one-line entries that mainly record position, resources, health, or teammate state.
- **Plan / future-tense word.** *will, going to, gonna, plan, planning, next, then, after, once, when*, plus future-tense modal forms (*i'll, we'll, i will, we will*). "Plan terms / entry" counts every match, while "% of entries" fires if at least one match is present.
- **Past-tense / memory word.** *tried, failed, was, were, did, didn't, couldn't, earlier, before, already, just, noticed, saw*.
- **Coordinate reference.** A token matching `\(?\s*-?\d+\s*,\s*-?\d+\s*\)?`, i.e. a pair of integers, possibly wrapped in parentheses, such as "$25, 23$" or "$(0, -1)$".
- **Teammate reference.** An "`Agent` X" or "`A`X" token for $X \in \{0, 1, 2\}$, optionally followed by a comma or a status colon.
- **Step-to-step turnover.** For every consecutive pair of controlled memory entries, we compute (a) the SequenceMatcher character similarity ratio and (b) Jaccard similarity over lower-cased tokens. Higher values indicate more overlap between consecutive entries.

The absolute values depend on the exact detector definitions, but the qualitative differences below are stable across seeds and robust to reasonable variants of the regexes.

L.2 Quantitative results

Table 9 shows that the two models use the memory field in different ways. Gemini tends to write longer, multi-line entries with explicit future plans. Gemma writes shorter entries that more often summarise the current state.

Three patterns stand out.

Gemini writes plans; Gemma writes state summaries. The structural differences are large. Gemini writes 3.86 lines per memory entry on average, compared with 1.08 for Gemma. Similarly, 71.5% of Gemini entries have at least two lines, compared with 2.0% for Gemma. Gemini also uses explicit turn-indexed plans in 12.4% of entries, while Gemma almost never uses this format (0.2%). Reading the entries gives the same picture: Gemini often lays out the next few turns line by line, whereas Gemma usually writes a compact one-line summary of position, recent obstacles, next movement, and teammate state.

Neither model uses the pad as memory. Both models use the memory entry more for immediate planning or state tracking than for recording past failures or lessons. Past-tense or retrospective tokens, such as *tried*, *failed*, *earlier*, *before*, *already*, and *noticed*, appear in only 9.0% of Gemini

Table 9: Memory entry usage on controlled steps (mean ± std across runs; Gemini $n = 10$, Gemma $n = 20$). Top: structural style. Middle: content. Bottom: step-to-step turnover from one controlled memory entry to the next.

	Gemini 3.1 Pro	Gemma
Structural style		
entry characters	206.7 ± 16.4	153.0 ± 9.6
entry lines	3.86 ± 0.72	1.08 ± 0.06
multi-line entry (≥ 2 lines), % of entries	71.5 ± 7.2	2.0 ± 1.3
turn-indexed plan (≥ 2 lines), % of entries	12.4 ± 9.4	0.2 ± 0.2
compact state summary (≥ 2 status fields), %	4.7 ± 1.8	16.1 ± 4.3
Content		
plan / future-tense word, % of entries	75.3 ± 2.5	36.6 ± 5.2
past-tense / memory word, % of entries	9.0 ± 1.1	6.7 ± 1.4
coordinate reference, % of entries	90.7 ± 3.1	98.8 ± 1.1
teammate reference, % of entries	68.0 ± 7.0	70.8 ± 7.1
coord references / entry	3.52 ± 0.71	4.01 ± 0.49
plan terms / entry	1.60 ± 0.11	0.52 ± 0.08
Step-to-step turnover (controlled → next controlled entry)		
char similarity (sequence ratio)	0.44 ± 0.02	0.58 ± 0.04
Jaccard on tokens	0.37 ± 0.01	0.47 ± 0.04

entries and 6.7% of Gemma entries. This is notable because the memory entry is one of the few persistent within-episode channels available to the model. It could be used to record facts such as a failed craft attempt, a previous handover, or an already-tested route, but the models rarely use it in this way. This could also be due to the history (last 8 observations and actions) agents receive in each observation.

Both models actively rewrite their memory entries. The memory entries are not simply copied forward unchanged. Consecutive controlled entries have token Jaccard similarity of 0.37 for Gemini and 0.47 for Gemma, with character-level similarity of 0.44 and 0.58, respectively. This indicates substantial rewriting from one controlled step to the next. Gemini rewrites more than Gemma, consistent with its planning-oriented usage: it often recomputes the next few actions rather than incrementally updating a fixed state summary.

L.3 Qualitative examples

Example selection. The examples below are illustrative rather than randomly sampled. We hold the role fixed across models by showing agent 0, the warrior, so that role-gated penalties and action affordances are comparable. We only show controlled steps, since forced `Noop` steps copy the previous memory entry. The Gemini example is a representative turn-indexed plan from run 02, step 9. The two Gemma examples, from run 00 step 0 and run 06 step 473, illustrate that the compact state-summary format appears both early and later in the episode. We did not search for unusually coherent or unusually successful entries. In our sample, we found no Gemma entries with the turn-indexed planning pattern.

Gemini 3.1 Pro: run 02, agent 0 (warrior), step 9, action: `Do`

T9: Do (completes handover at 26,23)
T10: Move West (towards 21,23)
T11: Give wood to Agent 1 (since A1 will Request Wood on T10)
T12+: Move West to table, then Make Wood Pickaxe.
Since I'm a warrior, pickaxe crafting has a chance to fail (40% or whatever the config is), so I'll keep trying if it does.

Gemma 4: agent 0 (warrior), runs 00 & 06

Run 00, step 0 action: `Move North`

Starting resource collection. Agent 1 (Forager) is best for wood (25, 23). Agent 2 (Miner) is best for stone (20, 23). I will move to (24, 23) to assist with wood gathering.

Run 06, step 473 action: `Move North`

Current (0, 19). Hit stone at (0, 20) and edge at x=-1. Moving North to (0, 18) to reposition and start searching East. A2 is searching North. A1 still sleeping.

The examples illustrate the quantitative differences in Table 9. The Gemini entry is an explicit turn-indexed plan: it records the current coordinated action, predicts a teammate request, schedules a resource transfer, and includes a contingency for the warrior crafting penalty. The Gemma entries instead compress the current situation into a short state summary: position, recent obstacle, next movement, and teammate status. These examples reflect the broader pattern: Gemini tends to use memory entries to maintain near-future plans, while Gemma tends to use them to restate the current state.

M Reasoning ablation: detailed analysis

M.1 Setup

The reasoning ablation differs across models because of API constraints. Gemini 3.1 Pro does not allow reasoning to be disabled, so we compare the high-reasoning baseline to a low-reasoning setting, denoted *Gemini-Low*. Gemma allows the reasoning trace to be disabled entirely, denoted *Gemma-No-Reasoning*. Gemini-Low still records 879 ± 44 characters and 198 ± 20 tokens of `raw_reasoning` per controlled step on average, so it should be interpreted as a partial ablation.

We collected 10 Gemini-Low episodes and 18 Gemma-No-Reasoning episodes. We reuse the same metrics as in App. L for memory entries on controlled steps, and App. N for messages across all records. The lexical detectors are exactly those defined in the corresponding appendices: turn-indexed plans, plan-word and past-word lexicons, coordinate references, imperative verbs, and related markers. The baseline columns in Table 10 are copied from Tables 9 and 11.

M.2 Quantitative results

Gemini loses much of its planner-like behaviour. Under low reasoning, the memory-entry features that characterised Gemini in the baseline largely disappear. Entry length falls from 207 to 69 characters, multi-line entries from 71.5% to 2.1%, and turn-indexed plans from 12.4% to 0.1%. The message channel changes in the same direction: messages fall from 132 to 53 characters, from 3.22 to 1.55 sentences, from 73.7% to 38.7% imperative verb usage, and from 56.5% to 25.6% references to coordinates. In short, the low-reasoning setting appears to removes much of the outward planning and directive communication observed in the baseline.

Gemma shifts more deliberation into memory entries. Gemma shows a different pattern. With reasoning disabled, memory entries become longer rather than shorter ($153 \rightarrow 228$ characters), often approaching the harness's ~ 1000-character cap. Their content also becomes more plan-like: plan-word rate rises from 36.6% to 73.9%, plan terms per entry from 0.52 to 1.28, and past-tense or reflective tokens from 6.7% to 18.3%. The entries still average roughly one line (1.10 vs. 1.08), so the change is not toward Gemini-style turn-indexed plans. Instead, Gemma writes longer single-line deliberations inside the memory field. The messages shift modestly in the same direction, with higher imperative usage ($52.2\% \rightarrow 62.2\%$) and more plan words ($27.0\% \rightarrow 37.6\%$).

This compensation is incomplete. Although Gemma appears to move some deliberation into the memory and message fields, performance still falls in the main ablation results: Coord.% drops by 3.3 points and Base% by 2.7 points. Gemini, which does not show the same expansion of

Table 10: Effect of reducing reasoning on memory-entry and message style (mean ± std across runs). Baseline columns are from Tables 9–11; ablated columns are the new runs ($n = 10$ Gemini-Low, $n = 18$ Gemma-No-Reasoning). The memory-entry block uses controlled steps only; the message block uses all records.

	Gemini 3.1 Pro		Gemma	
	Baseline	Low-Reason	Baseline	No-Reason
Memory entries (controlled steps only)				
entry characters	206.7 ± 16.4	$\mathbf{69.2 \pm 1.9}$	153.0 ± 9.6	$\mathbf{228.5 \pm 22.4}$
entry lines	3.86 ± 0.72	$\mathbf{1.04 \pm 0.01}$	1.08 ± 0.06	1.10 ± 0.12
multi-line entry (≥ 2 lines), %	71.5 ± 7.2	$\mathbf{2.1 \pm 0.5}$	2.0 ± 1.3	1.9 ± 2.3
turn-indexed plan, %	12.4 ± 9.4	$\mathbf{0.1 \pm 0.0}$	0.2 ± 0.2	0.2 ± 0.1
plan / future word, %	75.3 ± 2.5	$\mathbf{35.5 \pm 2.5}$	36.6 ± 5.2	$\mathbf{73.9 \pm 5.7}$
past / memory word, %	9.0 ± 1.1	2.9 ± 0.5	6.7 ± 1.4	$\mathbf{18.3 \pm 2.8}$
coordinate ref, %	90.7 ± 3.1	$\mathbf{45.5 \pm 5.4}$	98.8 ± 1.1	96.1 ± 2.5
teammate ref, %	68.0 ± 7.0	$\mathbf{27.8 \pm 6.4}$	70.8 ± 7.1	77.2 ± 7.2
plan terms / entry	1.60 ± 0.11	$\mathbf{0.49 \pm 0.04}$	0.52 ± 0.08	$\mathbf{1.28 \pm 0.16}$
Messages (all records)				
msg characters	132.4 ± 16.0	$\mathbf{52.9 \pm 3.4}$	104.1 ± 7.9	98.3 ± 13.1
sentences / msg	3.22 ± 0.40	$\mathbf{1.55 \pm 0.09}$	2.60 ± 0.16	2.28 ± 0.18
multi-sentence (≥ 2), %	93.7 ± 2.0	$\mathbf{47.3 \pm 6.7}$	87.9 ± 4.5	88.7 ± 4.9
imperative verb, %	73.7 ± 6.6	$\mathbf{38.7 \pm 5.4}$	52.2 ± 7.3	$\mathbf{62.2 \pm 10.8}$
plan / future word, %	59.1 ± 5.3	$\mathbf{22.1 \pm 3.4}$	27.0 ± 3.8	$\mathbf{37.6 \pm 7.2}$
contains coordinate, %	56.5 ± 10.0	$\mathbf{25.6 \pm 6.8}$	66.8 ± 10.7	67.6 ± 8.7
politeness (*please/thanks*), %	14.0 ± 3.1	10.4 ± 3.0	20.9 ± 4.8	$\mathbf{27.1 \pm 8.8}$
self-refs / msg	1.65 ± 0.18	$\mathbf{0.67 \pm 0.12}$	0.92 ± 0.17	1.16 ± 0.22

memory entries, loses more Coord.% (-7.8 points) and also loses Base% (-3.5 points). Thus, the behavioural traces match the quantitative ablation: reducing reasoning changes how models use the remaining harness channels, but those changes do not recover the lost performance.

M.3 Qualitative examples

Example selection. The examples below are illustrative rather than randomly sampled. For Gemini-Low, we show a typical one-line memory entry and a rare surviving turn-indexed plan. For Gemma-No-Reasoning, we show a long single-line deliberation and a reflective debugging entry. We hold the agent role fixed where possible, using agent 0 as the warrior in three of the four excerpts. We use agent 2 for the long Gemma deliberation example because that excerpt better illustrates the near-cap memory-entry behaviour. Within these constraints, we selected matching examples from fixed runs rather than searching across runs for the most extreme cases.

Gemini-Low: run 00, agent 0 (warrior)

Step 1 action: `Move East` *(typical one-line memory entry)*

Memory entry: *"Moving East to reach tree at 27,23 to coordinate with agents."*
Message: *"Moving to tree at 27,23 for 3-agent gather."*

Step 60 action: `Move South` *(rare surviving turn-indexed plan)*

Pathing to 29,28 to attack cow at 29,27.
Current: 27,26
T60: Move South (to 27,27)
T61: Move South (to 27,28)
T62: Move East (to 28,28)
T63: Move East (to 29,28)
T64: Do

Gemma-No-Reasoning: agent 0 (warrior) & agent 2

Run 00, agent 2, step 9 action: `Move East` *(long single-line deliberation)*

Current position (25, 25). Target tree (27, 23).
Path: (25, 25) -> (26, 25) -> (26, 24) [facing North] -> Do.
Agent 0 is at (26, 23) facing North/West? (Wait, Agent 0 is at 26,23, tree is at 27,23, so Agent 0 is West of tree and facing East). Agent 1 is at (27, 24) facing North. I will move to (26, 24) and face North (wait, if I'm at 26, 24 and tree is 27, 23, I should face East). Correct target: Tree (27, 23). My current pos: (25, 25). Action: Move East to (26, 25). Next: Move North to (26, 24), then Move East to face (27, 23)...Actually, if I move to (26, 23) I'd be on top of Agent 0. I'll go (26, 25) -> (26, 24) -> face East to target (27, 24)? No, tree is at (27, 23). [...]

Run 02, agent 0, step 211 action: `Move North` *(reflective debugging)*

[...] Wait, I just tried Move West several times and stayed at (25, 22). Let me check why. The construction site is at (24, 22). Construction sites are solid. I must go around. [...] Actually, looking at the observation, my last 5 moves were Move West and I stayed at (25, 22). This confirms (24, 22) is blocked.

The examples reflect the quantitative pattern in Table 10. Gemini-Low usually writes short memory entries and short messages; the turn-indexed format remains possible, but is rare. Gemma-No-Reasoning, by contrast, keeps the one-line memory format but fills it with path enumeration, corrections, and retrospective debugging. This suggests that disabling the reasoning trace changes not only performance, but also where each model places its intermediate deliberation.

N Inter-agent communication style

N.1 Setup

We analyse the per-step `message_sent` field, which is the open communication channel available to agents during an episode. We use the same trajectories as in the scratchpad analysis: 10 Gemini 3.1 Pro episodes and 20 Gemma 4 episodes, resulting 34,080 and 24,747 agent-step records respectively.

Lexical detectors. To characterise how agents use the channel, we compute a small set of lexical features from the raw message text. These are deliberately simple probes rather than discourse parsers. Their purpose is to capture stable differences in communication style, such as whether a message gives an instruction, names a teammate, asks a question, or records a plan. Table 11 reports the resulting rates. For completeness, the detectors are defined as follows:

- **Imperative verb.** A sentence-initial base-form verb from a fixed action lexicon (*move, go, do, place, make, drink, eat, attack, give, request, mine, chop, gather, face, wait, stay, build, kill, head, bring, get, take, drop, equip, craft, sleep, rest, hold, return, follow, stop, switch, prioritize, focus, defend*), optionally preceded by an addressee ("*Agent X,*"). We also count the same verbs when they immediately follow a teammate addressee elsewhere in the message. "Imperatives per message" counts all matches; "contains an imperative" is binary.
- **Broadcast phrasing.** A leading first-person plural cohortative: *let's, lets, let us, we should, we will, we need to, we must, we'll*.
- **Addresses specific teammate.** A token of the form "*Agent X,*" or "*Agent_X,*" followed by a clause, where $X \in \{0, 1, 2\}$.
- **Acknowledgement.** *ok, okay, got it, copy, acknowledged, agreed, sounds good, on it, will do, roger*.
- **Question.** A question mark outside coordinate tokens.
- **Politeness.** *please, thanks, thank you, ty*.
- **Urgency.** *now, urgent, urgently, immediately, asap, critical, critically, hurry, quickly, fast, right now*.

Table 11: Inter-agent message style across all records (mean ± std across runs; Gemini $n = 10$, Gemma $n = 20$). Top: structural style. Middle: speech-act and pragmatic markers. Bottom: content and turnover.

	Gemini 3.1 Pro	Gemma
Structural style		
message characters	132.4 ± 16.0	104.1 ± 7.9
message words	28.2 ± 3.8	20.9 ± 1.5
sentences / message	3.22 ± 0.40	2.60 ± 0.16
multi-sentence (≥ 2), % of msgs	93.7 ± 2.0	87.9 ± 4.5
Speech-act and pragmatic markers		
contains imperative verb, % of msgs	73.7 ± 6.6	52.2 ± 7.3
imperative verbs / message	1.80 ± 0.41	1.06 ± 0.20
contains broadcast phrasing (let's / we), %	12.8 ± 4.0	9.7 ± 3.7
addresses specific teammate ("Agent X,"), %	43.4 ± 7.0	44.1 ± 8.4
contains acknowledgement, %	5.3 ± 1.9	6.1 ± 2.2
contains a question, %	1.6 ± 0.4	1.7 ± 0.7
contains an exclamation mark, %	57.7 ± 7.1	43.8 ± 7.9
contains *please/thanks*, %	14.0 ± 3.1	20.9 ± 4.8
contains urgency word (*now, urgent, ...*), %	14.1 ± 2.5	21.2 ± 3.2
Content and turnover		
contains coordinate, % of msgs	56.5 ± 10.0	66.8 ± 10.7
contains future-tense / plan word, % of msgs	59.1 ± 5.3	27.0 ± 3.8
contains past-tense / memory word, % of msgs	8.5 ± 1.4	6.7 ± 1.7
self-references (*I, my, me*) / message	1.65 ± 0.18	0.92 ± 0.17
char similarity vs. previous message	0.39 ± 0.02	0.48 ± 0.02
Jaccard vs. previous message	0.27 ± 0.01	0.32 ± 0.03
verbatim same as previous message, %	0.5 ± 1.0	2.0 ± 2.3

- **Plan / future-tense word.** *will, going to, gonna, plan, planning, next, then, after, once, when*, plus future-tense modal forms (*i'll, we'll, i will, we will*).
- **Past-tense / memory word.** *tried, failed, was, were, did, didn't, couldn't, earlier, before, already, just, noticed, saw*.
- **Coordinate.** A token matching `\(?\s*-?\d+\s*,\s*-?\d+\s*\)?`, i.e. a pair of integers possibly wrapped in parentheses, such as "$25, 23$" or "$(0, -1)$".
- **Self-reference density.** Count of *i, my, me, i'm, i'll, mine* tokens, normalised per message.

The absolute values should be interpreted as proxy measurements, since they depend on the chosen lexicons. The main comparisons below are more important: the gaps are stable across seeds and robust to reasonable variants of the detectors.

N.2 Quantitative results

Table 11 groups the results into structural style, speech-act and pragmatic markers, and content turnover between consecutive messages.

Three patterns emerge.

Gemini communicates more like a planner. Gemini sends longer and more directive messages than Gemma. Its messages contain about 27% more characters, about 24% more sentences, and more imperative verbs per message (1.80 vs. 1.06). They also contain more self-references (1.65 vs. 0.92 per message). In the trajectories, this often looks like a model maintaining a team plan: Gemini

states what it is doing, assigns actions to particular teammates, and marks when the next coordinated step should happen. Gemma's messages are shorter and more often read as local status updates.

The channel is mostly used to broadcast intent, not to deliberate. The two models differ in tone. Gemini uses direct commands and exclamation marks more often. Gemma more often uses polite or urgent language: *please/thanks* appears in 20.9% of Gemma messages compared with 14.0% for Gemini, and urgency words appear in 21.2% compared with 14.1%. The shared pattern is equally important. Both models address a specific teammate at similar rates (43.4% vs. 44.1%), while questions are rare for both (1.6% and 1.7%). Thus, agents mainly use communication to announce intent, request help, and assign actions, rather than to conduct back-and-forth discussion.

Communication style mirrors scratchpad style. The message channel reflects the same model-specific tendencies seen in the scratchpads. Planning words appear in 59.1% of Gemini messages but only 27.0% of Gemma messages; in scratchpads, the corresponding rates are 75.3% and 36.6%. The channels are not redundant: Gemma's messages are more coordinate-heavy than its scratchpads, while Gemini's messages are less coordinate-heavy than its scratchpads. Still, the overall pattern is consistent. Gemini uses its scratchpad as a private plan and its messages as outward-facing instructions. Gemma uses its scratchpad more like a state snapshot and its messages more like status reports or requests for help.

N.3 Qualitative examples

How we sampled the examples. The excerpts below are illustrative, but we sample them with fixed criteria. We compare the same role in both models, agent 0 (the warrior), so available actions, crafting penalties, and team responsibilities are held constant. We also distinguish between controlled steps, where the agent has a real action choice, and forced `Noop` steps, where the agent is asleep, resting, or dead, but the harness still asks for a message.

Given these criteria, we take the first qualifying examples in chronological order from Gemini run 00 and Gemma run 00, rather than choosing messages that best fit the table. The only exception is the late HP-critical Gemma message: run 00 does not contain such a forced step, so we use the first matching example from Gemma run 06 at step 700. The Gemini box contains two controlled steps and one forced step; the Gemma box contains one controlled step and two forced steps.

Gemini 3.1 Pro: run 00, agent 0 (warrior)

Step 1 `ctrl` action: `Move East`
"I chopped 25,23. Let's do the 3-agent tree at 27,23! I'm moving to its West side (26,23). Agent 0, go to its North side (27,22). Agent 2, go to its South side (27,24). Face the tree and wait, we will 'Do' together when everyone is ready."

Step 10 `ctrl` action: `Make Wood Pickaxe`
"My wood pickaxe craft failed due to the warrior penalty! Trying again this turn. Once I get it, I will make a wood sword and then join you for the 3-agent stone."

Step 230 `forced` action: `Noop` hp 0
"Agent 2, your health is 1! You are about to die! Drink a potion IMMEDIATELY to heal if you have one! Place stone to block the orc!"

Gemma 4: runs 00 & 06, agent 0 (warrior)

Run 00, step 0 `ctrl` action: `Move North`
"Agent 1: face North and Do the tree at (25, 23). Agent 2: move to the stone at (20, 23). I'm moving North to (24, 23) to help gather wood."

Run 00, step 100 `forced` hp 5
"Almost there, thanks for waiting."

Run 06, step 700 `forced` hp 3
"Agent 1, please prioritize me for food as well, I am at 0 and my health is critical (3). Staying put."

These examples make the lexical patterns easier to interpret. Gemini's opening message contains a complete coordination proposal: it names the target, assigns positions to each teammate, and specifies when the joint `Do` action should happen. Its later messages keep this outward-facing style, either explaining the next step in its own plan or giving urgent instructions to another agent. Gemma's opening message also assigns roles and locations, showing that it can use the channel for coordination. However, its forced-step messages better reflect its aggregate style: concise status updates, polite acknowledgements, and requests tied to local health or resource needs.