
This is the **published version** of the bachelor thesis:

Pla Antigas, Mariona. *A Comprehensive Analysis of Cooperative Multi-Agent Reinforcement Learning*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026

This version is available at <https://ddd.uab.cat/record/336334>

under the terms of the  license.

A Comprehensive Analysis of Cooperative Multi-Agent Reinforcement Learning

Mariona Pla

February 6, 2026

Abstract

This project explores multi-agent reinforcement learning (MARL) by combining theoretical foundations with practical implementations. It examines key learning paradigms, including independent learning, centralised learning, and centralised training with decentralised execution, while addressing fundamental challenges such as non-stationarity and multi-agent credit assignment. A diverse set of representative MARL algorithms is analysed and evaluated to compare their performance, stability, and coordination properties. In addition, advanced strategies such as hierarchical reinforcement learning and explicit agent communication are explored. Particular attention is given to the trade-offs between centralised and decentralised approaches and their impact on cooperation. Overall, this project provides a comprehensive overview of recent advances and modern approaches in MARL.

Keywords: Reinforcement learning, multi-agent reinforcement learning, solution concepts, communication, cooperation

1 INTRODUCTION

Reinforcement learning (RL) is a subfield of machine learning in which an agent interacts dynamically with an environment and learns how to behave through trial and error. Unlike supervised learning, RL does not require prior knowledge in the form of labeled data. Instead, the agent makes decisions by selecting actions according to the current state of the environment. After each action, the environment responds with feedback in the form of a reward signal, which guides the agent toward learning an optimal strategy over time. However, rewards might be delayed; some actions may yield a lower immediate reward but lead to a higher cumulative reward over the long term.

One of the central challenges in RL is the balance between exploration and exploitation. Exploration allows the agent to gather new information about the environment, while exploitation uses the knowledge already acquired to maximise rewards. If the agent exploits too early, it may settle on a suboptimal policy because it has not explored enough alternatives. On the other hand, if it continues exploring for too long, it may waste time and resources instead of taking advantage of the optimal actions it has already discovered.

Multi-agent reinforcement learning (MARL) extends the RL paradigm to settings involving multiple agents that must coordinate or compete to achieve their objectives. In these scenarios, the environment's state evolves not only in response to a single agent's actions but as a function of

the collective actions of all agents. A central challenge in MARL is the non-stationarity of the environment, as each agent's learning process alters the dynamics perceived by other agents. Additional challenges include credit assignment, which involves determining the contribution of individual agents to overall outcomes, and scalability, as learning complexity grows with the number of agents. In cooperative settings, where agents share a common goal, agents can learn to communicate and collaborate effectively, thereby improving collective performance.

2 OBJECTIVES

This work aims to study MARL by combining theoretical analysis, algorithm implementation and experimental evaluation. The following objectives outline the main steps of the research.

- O1:** Study the theoretical foundations of MARL, including fundamental concepts, learning paradigms and key challenges.
- O2:** Analyse and evaluate a set of representative MARL algorithms, comparing their performance both qualitatively and quantitatively.
- O3:** Analyse how different learning paradigms influence the performance, stability and coordination outcomes in MARL.
- O4:** Investigate communication mechanisms in MARL and propose a novel or adapted communication strategy.

- Contact E-mail: Mariona.PlaA@autonoma.cat
- Supervised by: Jordi Casas Roma
- Academic Year 2025/26

- 05:** Experimentally assess the impact of communication on agent performance by analysing its effects on co-ordination, efficiency and qualitative aspects of agent behaviour.

3 METHODOLOGY AND WORK PLAN

3.1 Methodology

The tasks are organised in a *cascading* manner, meaning that each step builds upon the results of the previous one. This structured approach ensures a systematic progression from theoretical understanding to practical implementation and performance evaluation.

- **Definition of objectives:** Establish the research goals, including the study of MARL theoretical foundations, algorithm implementation, communication strategies and performance analysis.
- **Literature review:** Conduct a comprehensive review of existing literature to gather information on MARL theoretical foundations, main algorithms, learning paradigms, solution concepts and communication mechanisms among agents.
- **Environment selection and setup:** Explore suitable MARL environments and select the most appropriate one.
- **Algorithm selection:** Identify representative MARL algorithms for implementation based on the literature review and project objectives.
- **Algorithm adaptation and configuration:** Adapt, configure and extend existing MARL algorithm implementations to enable controlled experimentation, and communication mechanisms.
- **Agent training:** Train the agents using the implemented algorithms. Conduct modular training with variations in the number of agents and learning paradigms to evaluate performance and robustness.
- **Testing and evaluation:** Validate the correct functioning of agents and algorithms to ensure reproducibility and reliability and assess performance using quantitative metrics (e.g., cumulative rewards, convergence, stability) as well as qualitative aspects (e.g., coordination behaviours and communication effectiveness).
- **Research documentation:** Document methodology, results, analysis, and conclusions simultaneously for the final report.

3.2 Work plan

The project's work plan is outlined in Figure A1 (see Appendix), which details the main tasks and their allocated time frames in the Gantt chart.

4 PRELIMINARIES AND BACKGROUND

4.1 Main concepts

This section introduces the fundamental concepts underlying RL and its extension to MARL. The basic elements of RL are first reviewed, as they form the foundation for understanding how agents learn and act in dynamic environments. These concepts are then extended to MARL, where multiple agents interact and influence one another's outcomes, necessitating additional considerations.

4.1.1 Single agent RL

In RL, an agent learns to make decisions by interacting with an environment, receiving feedback in the form of rewards and adapting its behaviour to maximise the expected cumulative return. This framework relies on fundamental concepts, including states, actions, observations, rewards, returns, and policies.

- **Environment:** Physical or virtual world whose state s_t evolves over time and is influenced by the chosen actions a_{t-1} of the agents present in it.
- **Agent:** Entity that receives information about the state of the environment and influences it by choosing an action a_t according to its policy π .
- **State (s_t):** Current representation of the environment at time t . Can have a discrete or continuous representation.
- **Observation (o_t):** Information available to an agent about the environment at time t . In fully observable settings, the observation corresponds directly to the true state ($o_t = s_t$). In partially observable settings, observations provide only partial or noisy information about the underlying state.
- **Action (a_t):** Decision made by an agent at time t that influences the subsequent state of the environment. Actions can be discrete or continuous.
- **Reward (r_t):** Scalar feedback signal received after taking certain actions in certain states.
- **Return (G_t):** Cumulative discounted reward.

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$
- **Discount factor (γ):** Parameter controlling how future rewards are valued relative to immediate rewards.
- **Expected return:** The expectation of the return under a given policy π , representing the objective that the agent seeks to maximise.
- **Policy (π):** Function or distribution $\pi(a_t|s_t)$ used by the agent to select actions given the current state. Policies are goal-oriented.

- **State value function ($v_\pi(s)$):** Measures the expected return when starting in state s and subsequently following policy π .

$$v_\pi(s) = \mathbb{E}_\pi [G_t \mid S_t = s]$$

- **State-action value function** ($q_\pi(s, a)$): Measures the expected return when taking action a in state s and then continuing to act according to policy π . It provides a more fine-grained evaluation by distinguishing between different actions available in the same state.

$$q_\pi(s, a) = \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a]$$

The agent–environment interaction underlying these concepts is illustrated in Figure 1, showing the exchange of actions, observations, and rewards over time.

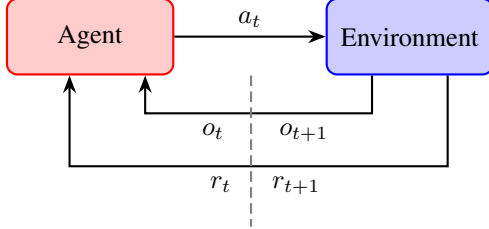


Fig. 1: Basic RL framework.

4.1.2 Extension to MARL

MARL extends single-agent RL to settings with multiple agents whose actions collectively influence the environment. This multi-agent context extends the RL framework with joint actions, joint policies and game-theoretic reward structures.

- **Joint action** ($\mathbf{a}_t = [a_t^1, a_t^2, \dots, a_t^n]$): Combination of all individual agents' actions at time t .
- **Joint policy** ($\pi = [\pi_1, \pi_2, \dots, \pi_n]$): Collection of n agents' individual policies.

The agent–environment interactions in a multi-agent setting, including joint action and individual rewards, are shown in Figure 2.

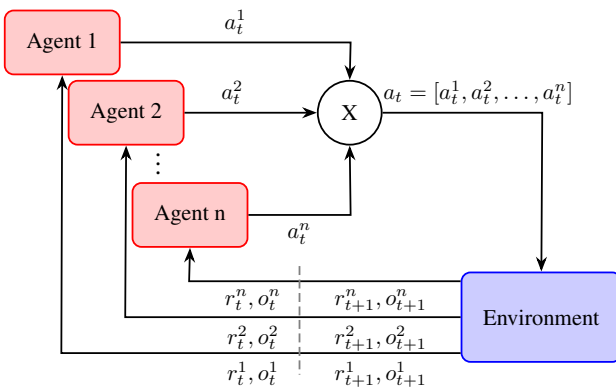


Fig. 2: Basic MARL framework.

Depending on how rewards are shared or opposed among agents, MARL settings can be grouped into the following game types:

- **Zero-sum games:** Agents' rewards always sum to zero. In two-player games, one agent's reward is the negative of the other's.
- **Common reward games:** All agents receive the same reward.

- **General sum games:** No restrictions on the relationship between agents' rewards.

4.2 Game-theoretic foundations

4.2.1 Game models

Multi-agent games can be organised into a hierarchy depending on the number of states and the degree of observability of those [1].

- **Normal-form games:** Represent a single strategic interaction between two or more agents. Each agent selects an action simultaneously, without knowledge of the others' choices. The outcome is determined by the joint action and each agent receives a reward according to the corresponding entry in the payoff matrix. Normal-form games capture one-shot, static decision-making scenarios.
- **Repeated normal-form games:** Extend the static normal-form setting where the same stage game is played repeatedly over time. After each round, agents may observe past actions or outcomes and adjust their policies accordingly. This repeated interaction allows for the emergence of long-term behaviours such as cooperation, punishment and reputation-building, which are not possible in one-shot games.
- **Stochastic games:** State-based environments in which the state evolves over time based on the agent's actions. At each stage, agents observe the current state and simultaneously select actions. The joint action determines both the immediate rewards for each agent and the probability distribution over the next state.
- **Partially observable stochastic games:** Extend stochastic games to settings where agents do not have full knowledge of the true underlying state of the environment. Instead, each agent receives a private observation that provides only partial information about the current state.

4.2.2 Solution concepts

Solution concepts [1] specify the desired properties of a joint policy in multi-agent systems. They formalise notions of optimality, stability and coordination. These concepts provide a theoretical lens for analysing learning dynamics and understanding the outcomes to which agents may converge.

- **Minimax:** A solution concept for two-player normal form games. Agent i uses a policy that is optimised for the worst-case opponent j that seeks to minimise agent i 's return.
- **Best-response:** Considering the policies of all agents except agent i , denoted as $\pi_{-i} = (\pi_1, \dots, \pi_{i-1}, \pi_{i+1}, \dots, \pi_n)$, the best response for agent i to π_{-i} is a policy π_i that maximises the expected return when playing against agents following π_{-i} . π_i might not be unique; there are usually multiple solutions.

- **Nash equilibrium:** A solution concept for general sum games. A joint policy π is a Nash equilibrium if no agent can unilaterally improve its expected return by deviating from its policy π_i , assuming that the policies of all other agents remain fixed.
- **ϵ -Nash equilibrium:** Given the computational cost of finding the exact Nash equilibrium in games with more than two players, the ϵ Nash equilibrium provides an approximate solution. A joint policy π is an ϵ -Nash equilibrium if no agent can increase its expected return by more than ϵ through unilateral deviation from its policy π_i .
- **Correlated equilibrium:** Generalises Nash equilibrium by allowing for correlation between agents' policies. Each agent receives a recommended action from π_i conditioned on the correlation variable ξ . Given ξ , no agent has an incentive to deviate from its recommended action. In coarse correlated equilibrium, agents need to accept or deviate from the action recommendation in advance.
- **Pareto optimality:** Aims to narrow down the space for equilibrium solutions. A joint policy π is Pareto-optimal if there is no other π that improves the expected return for at least one agent without reducing the expected return for any other agent.
- **Social welfare and fairness:** A solution concept for general sum games. A welfare-optimal joint policy maximises the sum of the agents' expected returns and a fairness-optimal joint policy maximises the product of the agents' expected returns, encouraging balanced outcomes among agents.
- **No regret:** Based on the difference between the rewards an agent received and the rewards it could have received if it followed a different policy. An agent has no-regret if in infinite episodes, the agent's average regret is at most zero, meaning the agent could not have consistently achieved higher rewards by deviating from its learned policy.

4.3 Learning Paradigms in MARL

The design of learning algorithms in MARL is influenced by how agents share information and coordinate their decision-making. Depending on whether policies and observations are centralised or decentralised during training and execution, different learning paradigms arise [1], each with distinct advantages and limitations.

4.3.1 Centralised learning

Centralised learning trains a single central policy π_c that controls all agents jointly. The training process has access to the local observations of all agents and outputs an individual action for each agent at every timestep. Despite the apparent simplicity of the concept, centralised learning involves several complex challenges.

First, the individual agent's reward must be aggregated into a single scalar value. In fully cooperative settings, this is straightforward since all agents share the same reward.

However, in zero-sum and general-sum stochastic games, reward scalarisation is neither simple nor uniquely defined. Second, the joint action space grows exponentially with the number of agents. Finally, the communication between the central policy and all the agents might not always be feasible; agents might be localized entities that are physically or virtually distributed.

4.3.2 Independent learning

Independent learning reduces a multi-agent problem into a set of single-agent problems from the perspective of each agent. Each agent i learns its policy π_i using only its own observations, actions and rewards while ignoring the existence of other agents. Although this decomposition avoids the exponential growth of the joint action space, each agent's environment becomes non-stationary due to the other agents' influence on the environment with their actions.

4.3.3 Centralised training with decentralised execution

The previous approaches can be combined into centralised training with decentralised execution (CTDE). In this framework, the training phase can exploit shared information from all agents to update policies, while during execution, each agent acts independently based only on its local observations.

4.4 Self-play and mixed-play

The division of a multi-agent problem into a set of single-agent problems allows agents to employ different learning algorithms [1].

The concept of **self-play** has two distinct interpretations. The first originates from zero-sum sequential games, where an agent's policy is trained against itself. This setup enables the agent to identify and overcome its own weaknesses through iterative improvement. The second interpretation arises in the context of independent learning, where all agents share the same learning algorithm.

In contrast, **mixed-play** refers to scenarios in which agents are trained using different learning algorithms. This setting more closely reflects real-world conditions, where agents often interact with others developed by different organisations and whose internal mechanisms are entirely unknown.

4.5 MARL challenges

MARL poses challenges that go beyond those encountered in single-agent settings, arising from the interactions between multiple learning agents. The most notable challenges [1] are summarized below.

- **Non-stationarity:** Since agents continuously update their policies during training, the environment becomes non-stationary from the perspective of each individual agent. In independent learning, the policies of other agents become part of the environment's state transition dynamics. As these policies evolve, the transition dynamics perceived by agent i also change. This

continual co-adaptation can lead to cyclic dynamics, where agents repeatedly adjust their policies in response to one another’s adaptations, making convergence difficult.

- **Equilibrium selection:** Many multi-agent games admit multiple equilibrium solutions, each potentially leading to different expected returns for the agents. Among these, a *reward-dominant* equilibrium yields higher collective payoffs if all agents coordinate successfully, whereas a *risk-dominant* equilibrium offers greater safety under uncertainty about others’ actions, by guaranteeing higher minimum reward. Consequently, learning algorithms may converge to suboptimal equilibria depending on factors such as initialization, exploration strategies, or learning dynamics. To mitigate this issue, one can reduce the solution space by imposing additional selection criteria such as Pareto optimality, welfare or fairness considerations or by incorporating agent modeling and coordination mechanisms to guide agents toward more desirable equilibria.
- **Multi-agent credit assignment:** In RL, the credit assignment problem concerns determining which past actions contributed to the received rewards. In multi-agent settings, this challenge becomes more complex, as it also requires identifying which agents’ actions were responsible for the observed outcomes. Addressing this problem requires each agent to disentangle the effects of its own actions from those of others.
- **Scalability:** In centralised learning, the joint state and action spaces grow exponentially with the number of agents, making training computationally expensive and often infeasible for large systems. In contrast, independent learning scales more easily since each agent learns its own policy. However, as the number of agents increases, the degree of non-stationarity in the environment also rises.

5 STATE OF THE ART

5.1 Formal foundations of RL

This section briefly reviews the key foundations of RL, including the standard formalization and value estimation methods [2].

Markov Decision Process (MDP): RL problems are typically formalized as an MDP, defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $P(s'|s, a)$ defines the transition probabilities, $R(s, a)$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The **Markov property** assumes that the future state depends only on the current state and action, not on past history:

$$P(R_{t+1} = r, S_{t+1} = s' \mid S_t, A_t) = P(R_{t+1} = r, S_{t+1} = s' \mid S_0, A_0, R_1 \dots, S_{t-1}, A_{t-1}, R_t, S_t, A_t) \quad (1)$$

In practice, however, the environment is often unknown, and computing or storing the full transition model P is computationally expensive. For this reason, many RL algorithms estimate value functions or policies directly from interaction data rather than relying on an explicit model.

Dynamic programming (DP): DP methods assume full knowledge of the MDP transition and reward functions. Given this perfect model, they compute optimal value functions and derive optimal policies through iterative procedures such as policy evaluation, policy iteration, and value iteration.

Monte Carlo (MC): MC methods estimate value functions by averaging the returns obtained from complete episodes of experience. The sample average converges to the value function without requiring any knowledge of the environment’s transition dynamics. However, MC methods can only update value estimates after an episode terminates, which can make them inefficient or impractical for tasks with very long or continuous episodes.

Temporal Difference (TD) learning: TD methods combine ideas from MC and DP. Instead of waiting until the end of an episode, TD updates the value function incrementally at each time step using the observed reward and the estimated value of the next state:

$$V(s_t) = V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (2)$$

where α is the learning rate. TD methods can learn online and handle continuing tasks efficiently.

5.2 Algorithm taxonomies

RL algorithms can be classified according to how they use experience and the data source [2].

On-Policy algorithms: Evaluate or improve the current policy used to make decisions. To reduce correlation between consecutive experiences, multiple independent instances of the environment can run in parallel. At each t , the agent receives a batch of states and rewards from all environments and outputs an action for each one. This setup ensures more diverse and uncorrelated experiences, which stabilizes learning.

Off-Policy algorithms: Evaluate or improve a policy different from the one used to generate the data. Experiences are typically stored in a replay buffer, which enables the algorithm to reuse past interactions for learning. The buffer usually has a fixed capacity and follows a first-in-first-out strategy: once full, the oldest experiences are replaced by new ones. Because the stored experiences were generated by earlier policies, the data is considered off-policy, allowing more diverse and uncorrelated training batches.

While the previous distinction focuses on how data is collected, algorithms can also be classified based on what they actually represent and optimize.

Value-based: Learn a parameterized value function and the agent follows a policy derived from this value function.

Policy-based: Directly learn a parameterised policy without explicitly learning a value function.

Actor-critic (AC): Learn both a parameterised policy (actor) and a parameterized value function (critic).

5.3 Algorithms

MARL algorithms differ in how they represent and optimize policies, handle exploration, and process experience. This section introduces several state-of-the-art algorithms, summarizing their key characteristics in Table 1. The classification highlights whether each algorithm uses independent

learning or CTDE, whether it is on-policy or off-policy, and the type of representation it employs (value-based, policy-based, or AC).

Algorithm	Ind./CTDE	On/Off policy	Val./Policy/AC
IQL	Indep	Off	Value
IA2C	Indep	On	AC
IPPO	Indep	On	AC
MADDPG	CTDE	Off	AC
COMA	CTDE	On	AC
MAA2C	CTDE	On	AC
MAPPO	CTDE	On	AC
VDN	CTDE	Off	Value
QMIX	CTDE	Off	Value

TABLE 1: MARL algorithm classification.

5.3.1 Independent Q-learning (IQL)

IQL [3] treats each agent as an independent DQN learner, each maintaining its own state-action value function and a corresponding target network, $Q_i(o_t^i, a_t^i)$. Action selection is handled by the multi-agent controller (MAC), which computes the per-agent Q-values at each timestep and applies an ϵ -greedy exploration strategy: with probability $1-\epsilon$ the agent selects $\arg \max_a Q_i(o_t^i, a)$, and with probability ϵ it selects a random available action. During training, experiences are sampled off-policy from the replay buffer, and the target MAC produces the next-step estimates used in the TD target:

$$y_t^i = r_t^i + \gamma \max_{a'} Q_i(o_{t+1}^i, a') \quad (3)$$

The learner then computes the TD error, $\delta_t^i = Q_i(o_t^i, a_t^i) - y_t^i$, and minimizes the squared TD loss over the masked batch. To reduce overestimation bias, Double Q-learning is optionally applied: the online network selects the maximizing action at the next state, while the target network evaluates it.

5.3.2 Independent synchronous Advantage Actor-Critic (IA2C)

IA2C is a variant of the Advantage Actor-Critic (A2C) algorithm [4, 5] and gives each agent its own actor and critic. The actor outputs a stochastic policy over the agent's action set $\pi(a | o_t^i)$, while the critic estimates a state-value function $V(o_t^i)$ that evaluates how good the agent's current observation is under its policy.

During training, the critic uses a target critic network to compute bootstrapped n -step returns,

$$G_t^i = \sum_{k=0}^{n-1} \gamma^k r_{t+k}^i + \gamma^n V_{\text{target}}(o_{t+n}^i) \quad (4)$$

which provide more stable learning targets. The critic is trained to minimise the TD error, $\delta_t^i = G_t^i - V(o_t^i)$ and the advantage is defined simply as $A_t^i = \delta_t^i$, capturing how much better or worse the taken action was compared to the critic's expected value.

The actor is updated using a policy-gradient objective, where each agent minimises:

$$\mathcal{L}_{\text{actor}}^i = -A_t^i \log \pi(a_t^i | o_t^i) - \beta \mathcal{H}(\pi(\cdot | o_t^i)) \quad (5)$$

that is, the advantage-weighted log-probability of the chosen action combined with an entropy regularisation term encouraging exploration.

5.3.3 Independent Proximal Policy Optimisation (IPPO)

IPPO is a variant of the Proximal Policy Optimization (PPO) [6] algorithm. It is closely related to IA2C in that each agent maintains its own decentralised actor and critic; however, it replaces the single-step policy-gradient update of A2C with multiple small, clipped policy updates that improve stability and prevent overshooting. For each update, the agent compares the new policy to an older snapshot using the probability ratio $r_t^i = \frac{\pi_{\theta}(a_t^i | o_t^i)}{\pi_{\theta_{\text{old}}}(a_t^i | o_t^i)}$ and optimises the clipped surrogate objective:

$$\mathcal{L}_{\text{actor}}^i = - \left(\min \left(r_t^i A_t^i, \text{clip}(r_t^i, 1 - \epsilon, 1 + \epsilon) A_t^i \right) + \beta \mathcal{H}(\pi(\cdot | o_t^i)) \right) \quad (6)$$

which limits how far the policy can move on each update.

5.3.4 Multi-Agent DDPG (MADDPG)

MADDPG [7] is a variation of the DDPG [8] algorithm for MARL. It extends the CTDE paradigm by equipping each agent with a decentralised actor and a centralised critic that conditions on the global state and joint action. The critic estimates the joint Q-value for all agents, allowing coordination through the gradient signal during training. MADDPG uses a state-action value function instead of a state-value function, updates actors directly via the deterministic policy gradient and uses Gumbel-Softmax to handle discrete action spaces.

5.3.5 Counterfactual Multi-Agent (COMA)

COMA [9] follows the CTDE paradigm using a centralised critic and decentralised actors. Like MADDPG, the critic estimates a state-action value function $Q(s_t, a_t^i, a_t^{-i})$ that conditions on the full global state and the joint actions. However, unlike MAA2C, COMA computes an agent-specific counterfactual advantage that isolates the contribution of each agent's action to the global outcome, which requires the *common-reward* setting. For each agent i , the baseline is the soft counterfactual expectation of its Q-values under its current policy while holding the other agents' actions fixed:

$$b_t^i = \sum_{a'} \pi(a' | o_t^i) Q(s_t, a', a_t^{-i}) \quad (7)$$

The advantage is then $A_t^i = Q(s_t, a_t^i, a_t^{-i}) - b_t^i$. This advantage measures how much better the selected action was compared to the agent's expected value over its alternative actions, given that teammates' actions are held constant. The counterfactual baseline reduces variance and addresses the multi-agent credit-assignment problem.

5.3.6 Multi-Agent A2C (MAA2C)

MAA2C extends IA2C by replacing the independent critics with a centralised critic that conditions on global information. Each agent still maintains its own decentralised actor,

$\pi(a_t^i | o_t^i)$, but the critic now estimates a joint value function that depends on the global state $V_{\text{central}}(s_t)$. The critic computes n -step bootstrapped targets and TD errors analogously to IA2C, and each actor is still updated independently using the same advantage-weighted policy-gradient objective. Training is centralised, but execution remains fully decentralised, as each agent only requires its own policy at test time.

5.3.7 Multi-Agent PPO (MAPPO)

MAPPO [10] is very similar to MAA2C in that it employs decentralised actors for each agent and a centralised critic that conditions on global information to compute joint value estimates. The key distinction is that, instead of using a single-step policy-gradient update like MAA2C, MAPPO performs multiple small, clipped policy updates per batch, following the IPPO objective. This clipping constrains how far the policy can change at each update, improving stability and preventing large destructive updates.

5.3.8 Value Decomposition Networks (VDN)

VDN [11] implements the central idea of decomposing the joint state-action value function into the sum of individual agent value functions: $Q_{\text{tot}}(o_t, a_t) = \sum_{i=1}^N Q_i(o_t^i, a_t^i)$. Each agent maintains its own Q-network, and the joint Q-value is computed centrally during training using the standard DQN update. VDN requires a *common reward* setting, since the linear sum assumes all agents share the same reward signal. During training, the TD loss of the joint Q-value is backpropagated to each agent's network, enabling coordinated learning. At execution time, agents act independently using ϵ -greedy policies with a shared replay buffer, consistent with the CTDE paradigm.

5.3.9 QMIX

QMIX [12] extends VDN by representing a more flexible decomposition of the joint Q-function using a parameterized mixing network that conditions on the global state and joint action. Like VDN, QMIX requires a *common reward* setting. The mixing network produces the joint Q-value, Q_{tot} , from individual agent Q-values, while enforcing a monotonicity constraint: the derivative of Q_{tot} with respect to each individual Q_i is non-negative, ensuring that an increase in any agent's Q-value cannot decrease the total Q-value. This constraint guarantees that the joint action that maximises Q_{tot} corresponds to the set of individual greedy actions from the agent Q-functions, preserving decentralised execution while enabling more complex coordination during centralised training.

5.4 Hierarchical reinforcement learning

Complex tasks are often difficult to solve using flat RL, as they require long-horizon planning and reasoning at multiple levels of abstraction. Inspired by the structure of human and animal decision-making, hierarchical reinforcement learning (HRL) decomposes a large task into a set of smaller, interdependent subtasks [13].

- **Temporal abstractions:** Reasoning happens over multiple time scales. Higher-level actions may span

multiple time steps, allowing the agent to plan using long-term behaviours.

- **State abstractions:** The agent simplifies the state representation by focusing only on the aspects relevant for decision-making at a given level of the hierarchy.

In **Feudal HRL**, decision-making is organised into multiple levels: managers have sub-managers, who work for them, and super-managers, for whom they work [14]. A high-level manager policy operates at a coarse temporal resolution and selects subgoals. Lower-level worker policies operate at a finer timescale and execute primitive actions to fulfill the subgoals provided by the manager. This recursive formulation continues until the lowest level interacts directly with the environment.

Higher-level managers do not need access to detailed state information; instead, they rely on abstracted representations of the environment. In contrast, lower-level controllers handle short-term behaviour and environment-specific dynamics [14, 15]. This hierarchical organisation induces temporal abstraction: decisions made at higher levels persist over multiple lower-level time steps.

- **Reward hiding:** A sub-manager is rewarded only for achieving its designated subgoal, regardless of whether this ultimately contributes to the higher-level manager's objective. Conversely, if it fails at its subgoal, it receives no reward even if the overall task succeeds. This separation allows each component of the hierarchy to learn robustly, even when higher-level subgoal choices are imperfect. This allows sub-managers to learn to achieve their corresponding subgoals even when the manager was mistaken in setting the sub-goals.
- **Information hiding:** Managers only need to know the state of the system at the granularity of their own choices of tasks.

Feudal HRL introduces additional structural properties:

- **Reward hiding:** Sub-managers are rewarded solely for achieving their assigned subgoals, independent of the overall task outcome. This separation enables robust learning even when higher-level subgoal assignments are imperfect [14].
- **Information hiding:** Managers operate on abstracted state representations relevant only to their level of control, reducing the complexity of decision-making [14].

5.5 Communication and coordination mechanisms

In coordination games, agents typically share a common goal and must work together to obtain the desired outcome. Communication plays a crucial role in this process by allowing agents to exchange information about their observations, intentions and future plans [1]. It can be understood as a type of action that can be observed by other agents but which does not affect the environment. However, this is not straightforward; agents must learn how to interpret the messages they receive.

Through communication, agents coordinate by aligning their decisions with the inferred intentions of others. Effective coordination therefore, depends on both the quality of the messages exchanged and how these messages are interpreted and integrated.

Messages should contain relevant information and avoid redundancy in their content. In order for message passing to be efficient, representation and encoding methods are necessary.

Message integration [16] refers to how incoming messages are combined before being internalized by the agent. There are different approaches for doing that:

- **Message concatenation:** All incoming messages are joined into a single vector without weighting or filtering. This keeps all information intact but increases input size, which can be inefficient as the number of agents grows.
- **Equally valued message integration:** Messages are combined using operations like averaging or summation, treating all agents equally and producing a fixed-size representation.
- **Weighted message prioritization:** Agents assign different importance levels to incoming messages, either manually or through learned attention mechanisms. This allows agents to focus on the most relevant information while filtering out noise.
- **Neural network-based integration:** Neural networks automatically learn which messages are most useful.

Each of these integration strategies offers a trade-off between informativeness, efficiency and scalability.

6 IMPLEMENTATION DETAILS AND EXPERIMENTAL SETUP

6.1 Multi-agent environments

Level-Based Foraging (LBF)¹ is a grid-based multi-agent environment in which agents move across a map to collect food items. Although the task may appear simple, the dynamics of food collection introduce complex coordination challenges. Each agent and each food item is assigned a level, and a food item can only be collected if the sum of the levels of all participating agents is greater than or equal to the level of that food. When a collection occurs, all agents that contributed to the required level receive a reward proportional to their own level.

The environment supports multiple configurations for the grid size, number of agents, number of apples, cooperative setting and partial observability. In the standard setting, an agent may collect food individually if its level is high enough, which creates competitive incentives. In contrast, the cooperative setting disallows individual collection, forcing agents to coordinate in order to achieve any reward. In the standard setting, the environment has full observability, which means that $o_i = s$ for each one of the agents. With the partial observability setting, each agent's view is limited to a patch of 5x5 cells; two cells in every direction.

¹<https://github.com/semitable/lb-foraging>

Agents have six discrete actions: *do nothing*, *load*, and *move* in one of the four cardinal directions (up, down, left, right). Each agent receives observations that encode both food and agent information. For every food item and every agent (including itself), the observation includes the x -position, y -position, and level. Thus, an agent's observation vector consists of the positions and levels of all food items, the agent's own information, and the information of all other agents.

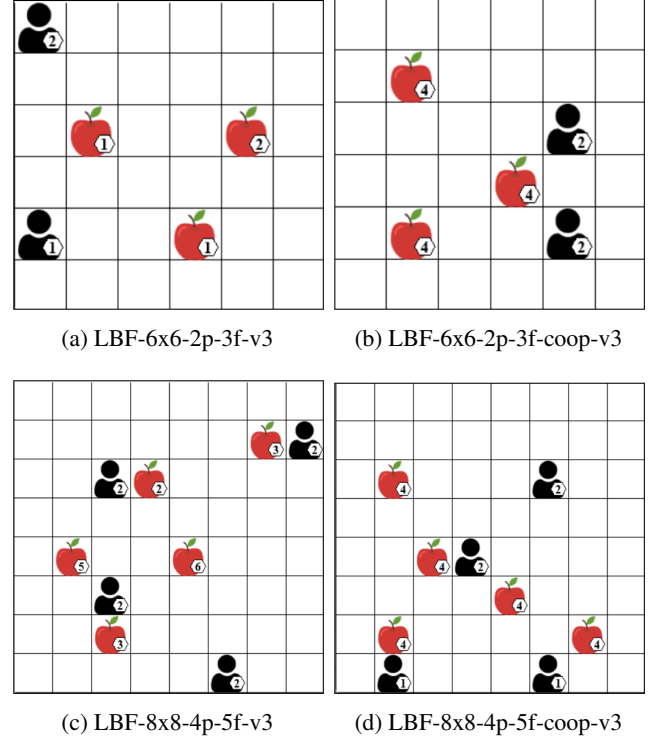


Fig. 3: LBF configuration examples. Environment configuration follows the format *lbforaging:Foraging-(grid size)x(grid size)-(sight)s-(players)p-(food)f-(coop)-v3*.

6.2 Hyperparameter grid search

To identify well-performing and stable training configurations, a hyperparameter grid search with the most important parameters was conducted for each algorithm (see Table A1 in Appendix). Key parameters include the learning rate, which controls the step size of network updates; the batch size and buffer size, which affect the variance of gradient estimates and the reuse of past experiences in off-policy algorithms; epsilon values, which determine the exploration-exploitation balance; n -step returns, which influence temporal credit assignment; and the entropy coefficient, which encourages exploration in policy-based methods. Algorithm-specific parameters such as reward scalarization for common reward settings and target update frequencies were also tuned to improve stability and learning performance.

Due to the large number of algorithms and hyperparameters evaluated, the grid search was performed on a single environment configuration, namely *Foraging-10x10-2p-2f-v3*. The hyperparameters selected through this process are subsequently reused across all environment configurations, acknowledging that they may not be optimal for every set-

ting. The same values were applied to both configurations, with and without parameter sharing.

6.3 Experiments

All the experiments are based on the EPyMARL² [17] algorithm implementation.

After conducting the hyperparameter grid search, Table 2 summarizes the best hyperparameters selected for each algorithm. Off-policy algorithms are trained for 4M timesteps, while on-policy algorithms are trained for 40M timesteps using 10 parallel environments. This difference reflects the inherent data efficiency of the two algorithm classes: off-policy methods can reuse past experiences through a replay buffer, allowing effective learning with fewer environment interactions, whereas on-policy methods require fresh trajectories for each update, necessitating more timesteps to achieve stable convergence.

All algorithms can be trained with or without *parameter sharing*. Without parameter sharing, each agent maintains its own network parameters. With parameter sharing, all agents use the same network, and their identities are encoded as one-hot vectors appended to the network inputs. The shared loss is computed across all agents and backpropagated into the shared parameters.

For the experiments, sixteen different environment configurations are used for each algorithm. The set of configurations includes variations in grid size, number of agents, and number of food items. For each variation, four different settings are considered: default, cooperative, partial observability, and cooperative with partial observability. To ensure robust evaluation, two random seeds are applied to each configuration.

To specifically study the impact of communication, additional experiments are conducted only for IQL, IPPO and MAPPO with 15x15 environment configurations under parameter sharing with two simple different mechanisms:

- **Target Apple:** Agent 0 is designated as the leader, and the first available apple in its observation is set as a shared target for other agents.
- **Last Action:** Each agent observes the previous action of all other agents.

7 RESULTS

7.1 Quantitative outcomes

Results are presented in Table 3 for algorithms with parameter sharing and Table 4 for those without parameter sharing. Across all LBF configurations, the results show a clear and consistent hierarchy between algorithm families, environment difficulty, observability, and the use of parameter sharing.

In small fully observable settings such as 10x10 with two players and two foods, most AC and policy-gradient methods with parameter sharing, including IA2C, IPPO, COMA, MAA2C, and MAPPO, reliably reach near-optimal performance with very low variance. This indicates that

coordination is relatively trivial in low-complexity grids. In contrast, value decomposition methods like VDN and QMIX show higher variance even in these simple cases, while MADDPG performs substantially worse, suggesting its continuous-control architecture is poorly suited to the discrete, coordination-heavy nature of LBF.

As the environments scale in size and number of agents, performance gaps widen, but independent training methods, particularly IPPO remains highly competitive and often achieves the top or near-top performance in 18x18 configurations, particularly with parameter sharing. In 12x12 and 15x15 settings, parameter-sharing actor-critic methods, especially MAA2C and MAPPO, generally achieve the highest returns and show robust scalability, but IPPO often matches or closely follows their performance, demonstrating that independent policy learning can scale effectively to larger environments. IA2C also remains competitive in many cases, though generally slightly below IPPO and the top centralised actor-critic methods. In contrast, independent value-based learning (IQL) and centralised value decomposition methods (VDN, QMIX) degrade rapidly as the number of agents grows, often collapsing to near-zero performance in larger grids, which highlights their sensitivity to non-stationarity and sparse coordination signals in complex cooperative tasks.

Partial observability, denoted by the 2s configurations, reduces performance across the board, though the impact is tiered. In small environments, the drop is moderate, and several methods, including IA2C, IPPO, MAA2C, and MAPPO, continue to achieve strong performance. In larger environments, performance under partial observability declines more noticeably, but MAA2C and MAPPO still lead, while IA2C and IPPO remain competitive and often outperform other baselines, showing that independent actor-critic methods are not entirely crippled by partial observability. Value-based approaches and MADDPG, however, frequently show larger performance drops, reflecting their challenges in adapting to limited local observations combined with multi-agent coordination demands.

Cooperative variants of the environments amplify these trends. While cooperation simplifies learning in small grids, in larger and partially observable cooperative settings only a subset of algorithms, mainly IA2C, IPPO, MAA2C, and MAPPO with parameter sharing, maintain meaningful performance. Many other methods either fail outright or exhibit extremely low returns.

Finally, comparing parameter sharing to non-sharing variants reveals that parameter sharing provides a systematic advantage. In almost every environment, shared-parameter versions achieve higher mean returns and lower variance than their non-sharing counterparts. This effect becomes more pronounced as task difficulty increases, suggesting that parameter sharing helps mitigate non-stationarity, improves sample efficiency, and enforces a useful inductive bias toward symmetric cooperative behavior. Without parameter sharing, many algorithms that perform well in small environments fail to scale or show extreme instability in larger or partially observable settings.

Overall, the experiments demonstrate that actor-critic methods with centralised critics and parameter sharing are the most reliable choice for LBF, but independent training approaches can also remain competitive.

²EPyMARL: Framework for multi-agent reinforcement learning. Available at: <https://github.com/ue-agents/epymarl>

Algorithm	lr	Hidden	Batch	Buffer	ϵ_{start}	ϵ_{finish}	Anneal	ϵ_{eval}	H coef.	Steps	Epochs	r scal.	Target upd.	t
IQL	3e-4	128	32	5k	1	0.05	200k	0.0	–	–	–	–	200	4M
IQL _{ns}	3e-4	128	32	5k	1	0.05	50k	0.0	–	–	–	–	200	4M
IA2C	5e-4	64	10	–	–	–	–	–	0.001	10	–	–	0.01	40M
IA2C _{ns}	5e-4	64	10	–	–	–	–	–	0.01	5	–	–	200	40M
IPPO	3e-4	64	10	–	–	–	–	–	0.001	5	2	–	0.01	40M
IPPO _{ns}	5e-4	64	10	–	–	–	–	–	0.01	5	2	–	200	40M
MADDPG	3e-4	64	64	10k	–	–	–	–	–	–	–	–	200	4M
MADDPG _{ns}	5e-4	64	64	10k	–	–	–	–	–	–	–	–	0.01	4M
COMA	3e-4	128	10	–	–	–	–	–	0.01	5	–	sum	200	40M
COMA _{ns}	3e-4	128	10	–	–	–	–	–	0.001	5	–	sum	200	40M
MAA2C	3e-4	128	10	–	–	–	–	–	0.01	5	–	–	200	40M
MAA2C _{ns}	3e-4	128	10	–	–	–	–	–	0.01	10	–	–	200	40M
MAPPO	5e-4	64	10	–	–	–	–	–	0.01	5	2	–	0.01	40M
MAPPO _{ns}	3e-4	128	10	–	–	–	–	–	0.01	5	2	–	0.01	40M
VDN	3e-4	64	64	1k	1	0.05	50k	0.05	–	–	–	sum	200	4M
VDN _{ns}	3e-4	64	64	5k	1	0.01	50k	0.05	–	–	–	sum	200	4M
QMIX	3e-4	64	64	1k	1	0.05	50k	0.05	–	–	–	sum	200	4M
QMIX _{ns}	3e-4	64	64	10k	1	0.05	50k	0.05	–	–	–	sum	200	4M

TABLE 2: Hyperparameters used for each algorithm. _{ns} denotes no parameter sharing; ‘–’ indicates not applicable.

Environment configuration	IQL	IA2C	IPPO	MADDPG	COMA	MAA2C	MAPPO	VDN	QMIX
10x10-2p-2f-v3	0.97±0.17	1.00 ±0.00	1.00 ±0.00	0.38±0.36	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	0.91±0.27	0.74±0.41
10x10-2p-2f-coop-v3	1.00 ±0.05	1.00 ±0.00	1.00 ±0.00	0.44±0.16	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	0.94±0.21	0.93±0.25
10x10-2s-2p-2f-v3	0.91±0.25	0.97±0.16	0.99 ±0.10	0.42±0.37	0.78±0.33	0.97±0.16	0.99 ±0.11	0.58±0.42	0.47±0.41
10x10-2s-2p-2f-coop-v3	0.92±0.21	0.96±0.15	0.99±0.11	0.63±0.35	0.74±0.51	0.99±0.09	1.00 ±0.05	0.47±0.45	0.21±0.35
12x12-3p-3f-v3	0.62±0.42	1.00 ±0.00	0.97±0.13	0.09±0.14	0.41±0.38	1.00 ±0.05	0.94±0.22	0.14±0.21	0.04±0.11
12x12-3p-3f-coop-v3	–	–	–	–	–	–	–	–	–
12x12-2s-3p-3f-v3	0.47±0.34	0.70±0.35	0.70±0.36	0.14±0.20	0.31±0.30	0.67±0.31	0.74 ±0.32	0.29±0.29	0.15±0.19
12x12-2s-3p-3f-coop-v3	0.39±0.29	0.72±0.25	0.62±0.33	0.03±0.12	–	0.78 ±0.24	–	0.05±0.14	0.01±0.05
15x15-4p-4f-v3	0.59±0.40	0.99 ±0.07	0.93±0.21	0.05±0.11	0.13±0.16	0.93±0.22	0.82±0.31	0.07±0.11	0.02±0.06
15x15-4p-4f-coop-v3	–	0.92±0.17	0.61±0.27	0.00±0.03	0.00±0.03	0.93 ±0.22	0.01±0.04	0.00±0.04	–
15x15-2s-4p-4f-v3	0.56±0.29	0.70±0.32	0.67±0.30	0.13±0.15	0.26±0.23	0.69±0.31	0.75 ±0.30	0.29±0.25	0.22±0.24
15x15-2s-4p-4f-coop-v3	0.26±0.23	0.42±0.27	0.37±0.26	0.01±0.05	0.06±0.13	0.45 ±0.30	0.38±0.30	0.03±0.13	0.03±0.09
18x18-5p-4f-v3	0.57±0.29	0.83±0.26	0.93 ±0.20	0.05±0.10	0.12±0.15	0.88±0.28	0.89±0.25	0.08±0.13	0.04±0.04
18x18-5p-4f-coop-v3	–	0.80±0.28	0.84 ±0.22	0.01±0.05	0.02±0.06	0.78±0.36	0.00±0.02	0.01±0.05	–
18x18-2s-5p-4f-v3	0.65±0.28	0.75±0.27	0.77 ±0.29	0.14±0.18	0.31±0.25	0.76±0.28	0.72±0.26	0.36±0.28	0.31±0.28
18x18-2s-5p-4f-coop-v3	0.31±0.26	0.42±0.29	0.49 ±0.29	0.00±0.02	0.08±0.15	0.47±0.27	0.47±0.27	0.11±0.17	0.14±0.21

TABLE 3: Maximum test return mean and standard deviation over two seeds. Algorithms with parameter sharing.

7.1.1 Communication impact

Table 5 summarises the maximum test return for IQL, IPPO, and MAPPO across different 15×15 environment configurations. Overall, the results indicate that simple communication mechanisms can provide modest but inconsistent benefits, with the impact strongly depending on both the learning algorithm and the task structure.

For IQL, the *Last Action* mechanism consistently yields the best or comparable performance across most configurations, suggesting that observing teammates’ recent actions helps mitigate non-stationarity and improves implicit coordination. In contrast, the *Target Apple* mechanism does not consistently outperform the no-communication baseline, likely due to its reliance on a fixed leader and a potentially suboptimal shared target.

In the case of IPPO, communication provides limited and inconsistent gains. The most notable improvement occurs in the cooperative setting with full observability, where the *Target Apple* mechanism yields a measurable boost in performance. Across the remaining environment configurations, the impact of communication is minor, indicating that IPPO’s learning process is largely able to capture the necessary coordination without relying on explicit messages.

For MAPPO, communication shows mixed effects across environments. During training, MAPPO employs a centralised critic with access to the global state of the envi-

ronment, which already provides rich coordination signals to the agents. As a result, the benefit of additional explicit communication during execution is reduced and inconsistent. In the default setting, the *Target Apple* mechanism leads to a slight performance improvement, indicating that a shared high-level objective can still be beneficial. However, in cooperative environments and under partial observability, communication does not consistently improve performance and can even degrade it. This behavior is likely due to redundancy with the centralised training signal and the added complexity introduced at execution time, which may interfere with the policies learned under centralised supervision.

In summary, these results suggest that simple, low-bandwidth communication mechanisms are not universally beneficial. Their effectiveness depends on the underlying algorithm and task complexity. While action-based communication can support coordination in decentralised methods like IQL, more advanced learners such as IPPO and MAPPO already capture much of the required coordination implicitly, limiting the marginal utility of explicit communication.

7.2 Qualitative assessment

A deeper qualitative analysis based on rendered episodes was conducted for all 15×15 environment configurations.

Environment configuration	IQL _{ns}	IA2C _{ns}	IPPO _{ns}	MADDPG _{ns}	COMA _{ns}	MAA2C _{ns}	MAPPO _{ns}	VDN _{ns}	QMIX _{ns}
10x10-2p-2f-v3	0.78±0.40	0.99±0.10	0.99±0.11	0.41±0.31	1.00±0.00	1.00±0.00	1.00±0.00	0.95±0.19	0.63±0.42
10x10-2p-2f-coop-v3	0.43±0.17	1.00±0.05	1.00±0.00	0.44±0.17	1.00±0.00	1.00±0.00	1.00±0.00	0.98±0.12	–
10x10-2s-2p-2f-v3	0.35±0.39	0.90±0.25	0.85±0.31	0.33±0.35	0.87±0.29	0.95±0.19	0.96±0.16	0.51±0.41	0.56±0.43
10x10-2s-2p-2f-coop-v3	0.42±0.42	0.78±0.31	0.87±0.30	0.51±0.37	0.76±0.36	0.94±0.20	0.84±0.27	0.52±0.44	0.24±0.33
12x12-3p-3f-v3	0.25±0.31	0.87±0.31	0.89±0.27	0.27±0.24	0.12±0.17	0.93±0.21	0.95±0.18	0.05±0.15	0.01±0.03
12x12-3p-3f-coop-v3	–	–	–	–	–	–	–	–	–
12x12-2s-3p-3f-v3	0.19±0.24	0.45±0.31	0.54±0.37	0.21±0.21	0.24±0.28	0.67±0.32	0.60±0.37	0.25±0.31	0.15±0.19
12x12-2s-3p-3f-coop-v3	–	0.51±0.38	0.56±0.37	–	–	–	–	0.03±0.11	0.02±0.09
15x15-4p-4f-v3	0.06±0.11	0.75±0.35	0.87±0.29	0.10±0.15	0.07±0.12	0.86±0.25	0.83±0.31	0.05±0.08	0.00±0.02
15x15-4p-4f-coop-v3	–	0.01±0.05	0.55±0.28	0.00±0.03	0.00±0.03	0.62±0.24	–	–	–
15x15-2s-4p-4f-v3	0.21±0.24	0.51±0.29	0.59±0.31	0.24±0.26	0.11±0.15	0.52±0.30	0.54±0.28	0.25±0.25	0.20±0.23
15x15-2s-4p-4f-coop-v3	0.04±0.07	0.32±0.27	0.26±0.28	0.02±0.08	0.03±0.08	0.32±0.27	0.31±0.25	0.04±0.1	0.01±0.04
18x18-5p-4f-v3	0.04±0.08	0.85±0.24	0.86±0.28	0.06±0.12	0.09±0.14	0.78±0.27	0.93±0.19	0.05±0.09	0.01±0.05
18x18-5p-4f-coop-v3	0.02±0.02	0.29±0.20	0.00±0.02	0.00±0.02	0.01±0.03	0.59±0.21	–	–	–
18x18-2s-5p-4f-v3	0.28±0.26	0.52±0.30	0.56±0.30	0.26±0.25	0.10±0.15	0.51±0.33	0.56±0.33	0.29±0.26	0.13±0.21
18x18-2s-5p-4f-coop-v3	0.09±0.16	0.25±0.23	0.30±0.29	0.04±0.09	0.02±0.07	0.33±0.27	0.30±0.27	0.08±0.15	0.03±0.08

TABLE 4: Maximum test return mean and standard deviation over two seeds. Algorithms without parameter sharing.

Environment config.	TT	TF	FT	FF
IQL				
15x15-4p-4f-v3	0.78±0.34	0.57±0.40	0.79±0.36	0.59±0.40
15x15-4p-4f-coop-v3	–	–	–	–
15x15-2s-4p-4f-v3	0.70±0.28	0.57±0.30	0.71±0.26	0.59±0.30
15x15-2s-4p-4f-coop-v3	0.41±0.25	0.21±0.21	0.43±0.28	0.27±0.26
IPPO				
15x15-4p-4f-v3	0.97±0.13	0.93±0.21	0.92±0.20	0.86±0.29
15x15-4p-4f-coop-v3	0.82±0.29	0.84±0.29	0.61±0.25	0.61±0.25
15x15-2s-4p-4f-v3	0.66±0.31	0.66±0.32	0.64±0.31	0.62±0.30
15x15-2s-4p-4f-coop-v3	0.36±0.26	0.32±0.30	0.32±0.28	0.26±0.24
MAPPO				
15x15-4p-4f-v3	0.85±0.29	0.94±0.19	0.87±0.29	0.85±0.27
15x15-4p-4f-coop-v3	–	–	–	–
15x15-2s-4p-4f-v3	0.72±0.29	0.67±0.32	0.70±0.30	0.66±0.31
15x15-2s-4p-4f-coop-v3	0.39±0.28	0.33±0.25	0.34±0.29	0.42±0.28

TABLE 5: Maximum test return mean and standard deviation over two seeds. Columns denote communication configurations: Target Apple + Last Action (TT), Target Apple (TF), Last Action (FT), and no communication (FF)

Algorithms using parameter sharing exhibit faster coordination behaviours, with agents tending to move together early in the episode. As a result, these agents typically complete episodes in fewer timesteps compared to their non-parameter-sharing counterparts.

Despite these temporal differences, no substantial differences are observed in the high-level strategies learned by the agents. Across both independent learning and CTDE approaches, agents generally converge to similar coordination patterns, forming pairs or moving as a group toward food items. Consequently, variations in agents’ returns are not driven by competitive interactions but are primarily explained by differences in agent levels and task allocation. These findings suggest that parameter sharing mainly improves learning efficiency and coordination speed, rather than inducing fundamentally different cooperative strategies.

Qualitative inspection of algorithms that fail to achieve high performance reveals two recurring failure modes. In some cases, agents exhibit near-static behaviour, frequently selecting the *None* action and remaining inactive for extended periods. In other cases, agents approach a food item

but are unable to collect it due to insufficient combined levels, leading them to wait indefinitely for additional agents instead of exploring alternative targets.

Finally, in partially observable configurations, agents tend to display more repetitive and oscillatory movement patterns, reflecting increased uncertainty about the environment state and the positions of other agents.

8 CONCLUSIONS

This study provides a comprehensive evaluation of MARL algorithms across a range of cooperative environment configurations, investigating factors such as observability, environment size, agent count, and the use of parameter sharing and communication mechanisms. The objectives of comparing different algorithm families and understanding the impact of architectural choices have been broadly addressed.

The findings indicate that actor-critic methods with centralised critics and parameter sharing generally offer the most reliable performance, while independent policy learning approaches can remain competitive. Simple communication mechanisms provide limited and task-dependent benefits, highlighting the challenges of coordination in partially observable and complex multi-agent settings.

Despite these insights, several important areas remain open for further research. HRL and more sophisticated communication strategies are largely unexplored and may offer substantial improvements for coordination and efficiency in complex tasks. Future work could also explore scalable architectures, adaptive learning mechanisms, and richer interaction protocols to advance multi-agent learning in realistic environments.

Overall, this work contributes to a better understanding of multi-agent learning dynamics, highlights the importance of design choices and identifies promising directions for ongoing research in hierarchical and communicative MARL.

REFERENCES

- [1] S. V.-Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and*

- Modern Approaches*. MIT Press, 2024.
- [2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.
 - [3] M. Tan, “Multi-agent reinforcement learning: Independent versus cooperative agents,” in *International Conference on Machine Learning*, 1997.
 - [4] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 1928–1937.
 - [5] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, “Deep reinforcement learning that matters.” AAAI Press, 2018.
 - [6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017.
 - [7] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” *CoRR*, vol. abs/1706.02275, 2017.
 - [8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv*, 2015.
 - [9] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in *AAAI Conference on Artificial Intelligence*, 2017.
 - [10] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu, “The surprising effectiveness of MAPPO in cooperative, multi-agent games,” *CoRR*, vol. abs/2103.01955, 2021.
 - [11] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, “Value-decomposition networks for cooperative multi-agent learning based on team reward.” International Foundation for Autonomous Agents and Multiagent Systems, 2018.
 - [12] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, “QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning,” in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 4295–4304.
 - [13] M. Hutsebaut-Buyse, K. Mets, and S. Latré, “Hierarchical reinforcement learning: A survey and open research challenges,” *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 172–221, 2022.
 - [14] P. Dayan and G. E. Hinton, “Feudal reinforcement learning,” *Neural Information Processing Systems*, 1992.
 - [15] A. S. Vezhnevets, S. Osindero, T. Schaul, N. M. O. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, “Feudal networks for hierarchical reinforcement learning,” in *International Conference on Machine Learning*, 2017.
 - [16] J. Chen, H. Yang, and Z. Liu, “The five ws of multi-agent communication: Who talks to whom, when, what, and why - a survey from marl to emergent language and llms,” *TechRxiv*, 2025.
 - [17] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, “Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

APPENDIX

A GANTT CHART

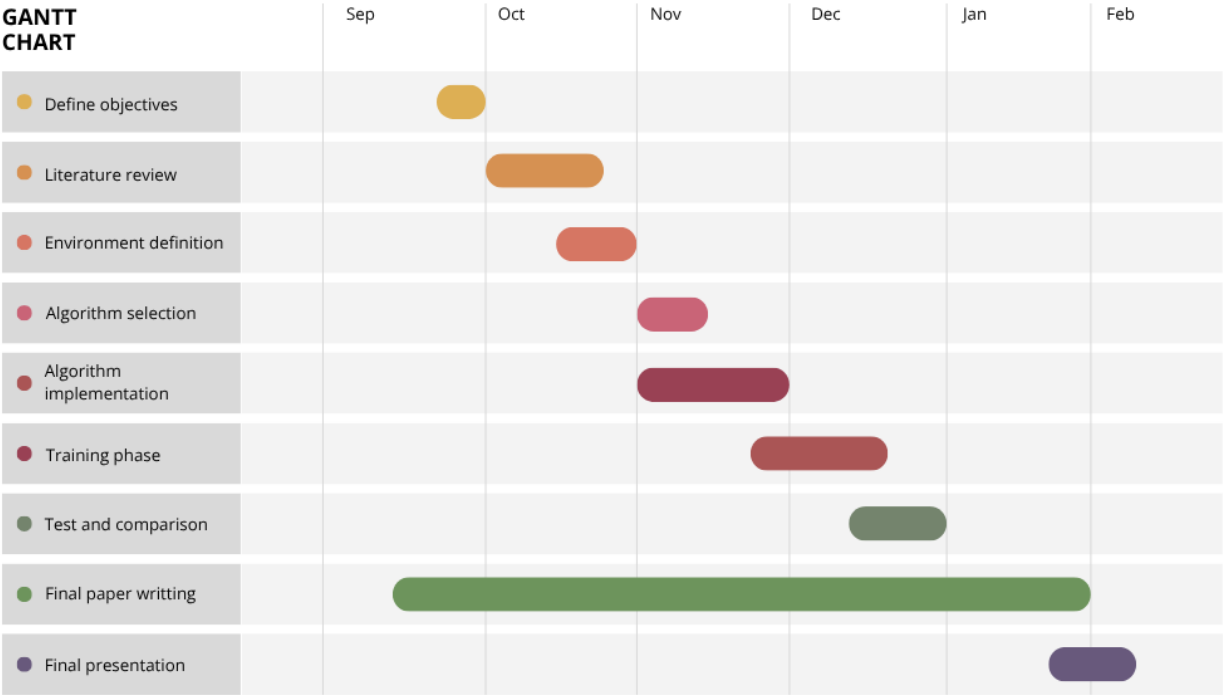


Fig. A1: Gantt Chart

B HYPERPARAMETERS

Parameter	IQL	IA2C	IPPO	MADDPG	COMA	MAA2C	MAPPO	VDN	QMIX
Learning rate (lr)	0.0001	0.0003	0.0003	0.0003	0.0001	0.0003	0.0003	0.0001	0.0001
	0.0003	0.0005	0.0005	0.0005	0.0003	0.0005	0.0005	0.0003	0.0003
Buffer size	1000			10000				1000	1000
	5000	—	—	20000	—	—	—	5000	5000
	10000			50000				10000	10000
Hidden dimension	64	64	64	64	64	64	64	default	default
	128	128	128	128	128	128	128		
Batch size	32	default	default	32	default	default	default	default	default
	64			64					
Epsilon finish	0.01	—	—	—	—	—	—	0.01	0.01
	0.05							0.05	0.05
Eval epsilon	default	—	—	—	—	—	—	0	0
								0.05	0.05
Return number of steps	—	1	default	—	1	1	default	—	—
		5			5	5			
		10			10	10			
Reward scalarisation	—	—	—	—	mean sum	—	—	mean sum	mean sum
Target update	default	200	200	200	default	200	200	default	default
		0.01	0.01	0.01		0.01	0.01		
Entropy coefficient	—	0.01	0.01	—	0.01	0.01	0.01	—	—
		0.001	0.001		0.001	0.001	0.001		
Epochs (n -step)	—	—	2	—	—	—	2	—	—
			4				4		

TABLE A1: Hyperparameter grid search values per algorithm

C VISUAL RESULTS

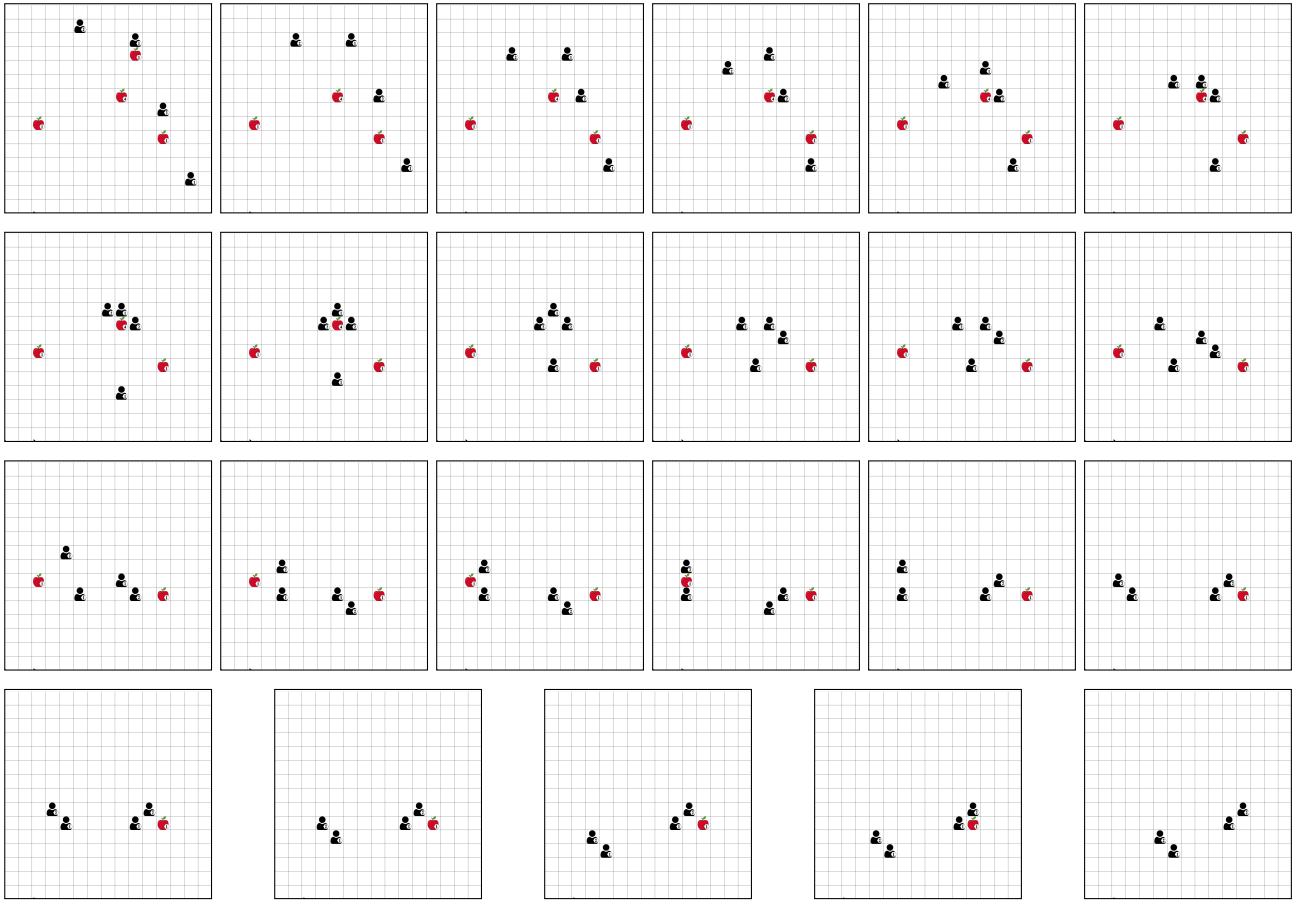


Fig. A2: Rendered episode for IA2C in a 15x15-4p-4f-v3 environment. Episode reward: 1, Timesteps: 23.

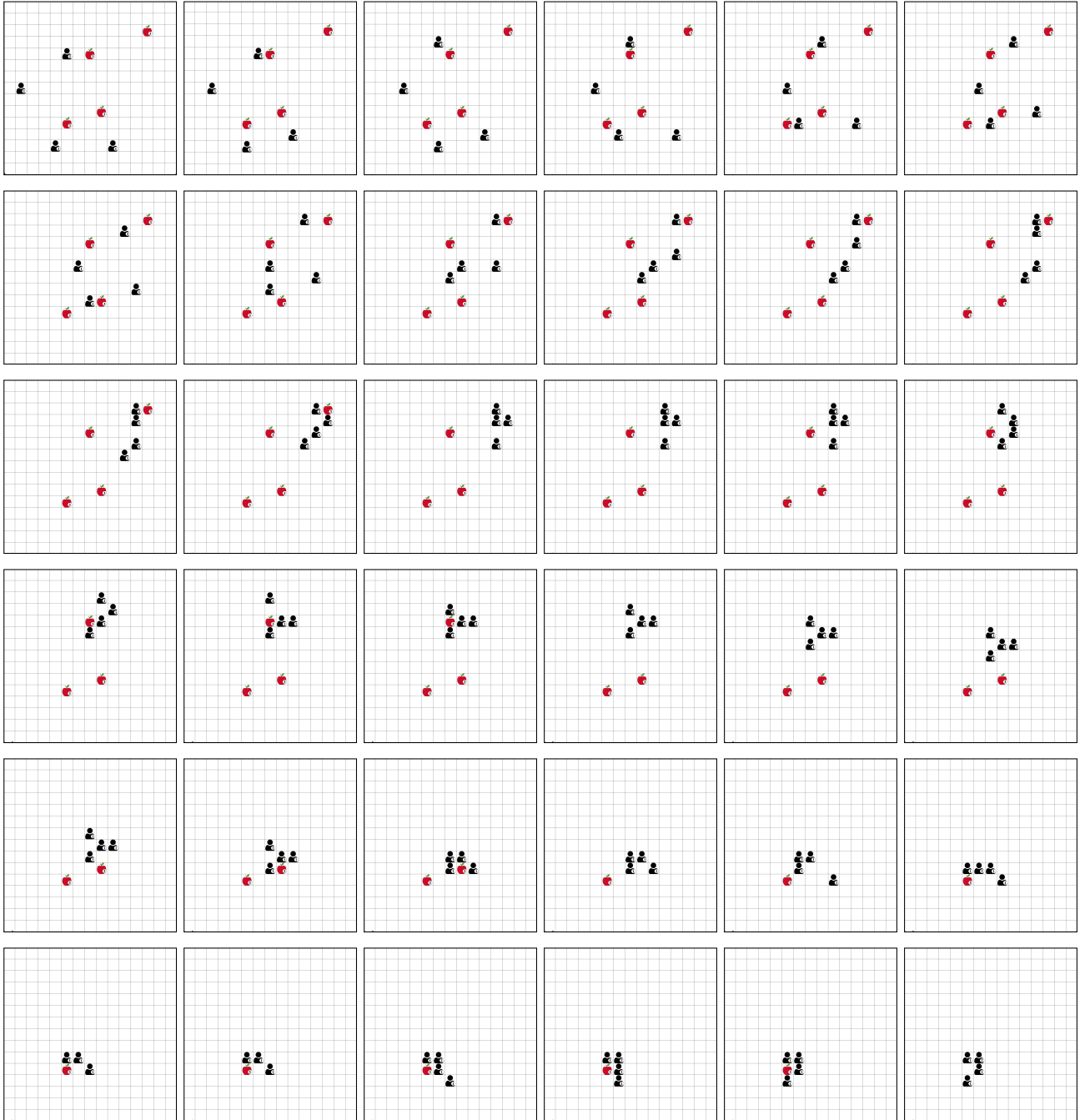


Fig. A3: Rendered episode for IA2C in a 15x15-4p-4f-coop-v3 environment. Episode reward: 1, Timesteps: 36.



Fig. A4: Rendered episode for IA2C in a 15x15-2s-4p-4f-v3 environment. Episode reward: 1, Timesteps: 28.

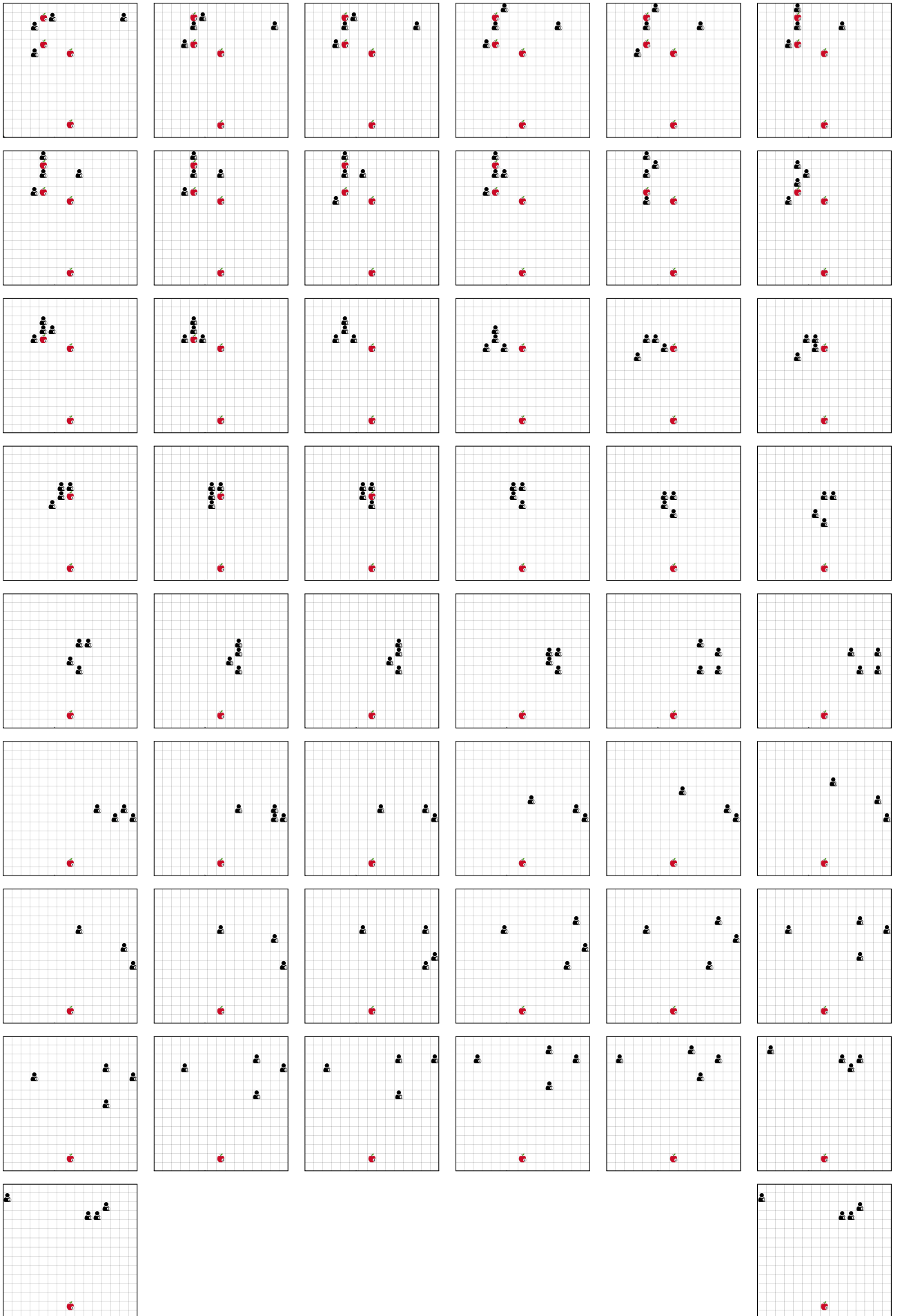


Fig. A5: Rendered episode for IA2C in a 15x15-2s-4p-4f-coop-v3 environment. Episode reward: 0.75, Timesteps: 50.

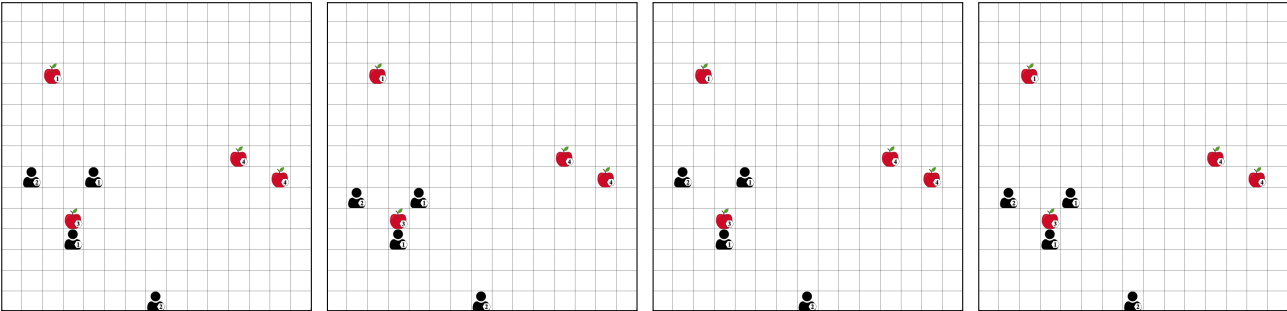


Fig. A6: Repetitive and oscillatory movement in QMIX in a 15x15-2s-4p-4f-v3 environment.