

Joint Intrinsic Motivation for Coordinated Exploration in Multi-Agent Deep Reinforcement Learning

Extended version of the extended abstract paper published at AAMAS 2024

Maxime Toquebiau*

ECE Paris & Sorbonne Université, CNRS, ISIR
F-75005 Paris, France
maxime.toquebiau@gmail.com

Faïz Benamar

Sorbonne Université, CNRS, ISIR
F-75005 Paris, France
faiz.ben_amar@sorbonne-universite.fr

Nicolas Bredeche†

Sorbonne Université, CNRS, ISIR
F-75005 Paris, France
nicolas.bredeche@sorbonne-universite.fr

Jae-Yun Jun†

ECE Paris
Paris, France
jaeyunjk@gmail.com

ABSTRACT

Multi-agent deep reinforcement learning (MADRL) problems often encounter the challenge of sparse rewards. This challenge becomes even more pronounced when coordination among agents is necessary. As performance depends not only on one agent's behavior but rather on the joint behavior of multiple agents, finding an adequate solution becomes significantly harder. In this context, a group of agents can benefit from actively exploring different joint strategies in order to determine the most efficient one. In this paper, we propose an approach for rewarding strategies where agents collectively exhibit novel behaviors. We present JIM (Joint Intrinsic Motivation), a multi-agent intrinsic motivation method that follows the centralized learning with decentralized execution paradigm. JIM rewards joint trajectories based on a centralized measure of novelty designed to function in continuous environments. We demonstrate the strengths of this approach both in a synthetic environment designed to reveal shortcomings of state-of-the-art MADRL methods, and in simulated robotic tasks. Results show that joint exploration is crucial for solving tasks where the optimal strategy requires a high level of coordination.

Keywords. Multi-agent Systems, Deep Reinforcement Learning, Intrinsic Motivation

1 INTRODUCTION

One crucial aspect of human intelligence is its ability to act coincidentally with other human beings, to either cooperate or compete in a given task. This has led researchers to study reinforcement learning (RL) in the context of multi-agent systems (MAS), where multiple artificial agents interact with their environment and each other while concurrently learning to perform a task [11, 28]. However, having multiple agents in the environment makes the RL process significantly more difficult for several reasons [33]. In particular, the global reward depends on the actions of several independent agents, which makes the search for the optimal joint policy more complicated.

Recently, multi-agent deep reinforcement learning (MADRL) approaches have combined advancements in RL and deep learning to

tackle long-standing problems in MAS such as credit assignment or partial observability [6, 11, 22]. These techniques are able to solve very complex multi-agent tasks such as autonomous driving [25] or real-time strategy video games [17]. However, major issues still remain with these approaches, such as the problem of relative overgeneralization [30, 32] where agents struggle to find the optimal joint policy because local policies are attracted towards suboptimal areas of the search space. This makes most algorithms inefficient in tasks where the optimal strategy requires strong coordination among agents. Relative overgeneralization can be described as a problem of exploration of the joint-state space: as the success of the MAS depends on the coordination of multiple agents, exploring the joint-observation space is required to discover optimal joint behaviors. In this paper, we address the question of how to explore the joint-state space to efficiently discover superior coordinated strategies for solving the task at hand.

In single-agent RL, the problem of exploration has been studied to solve hard exploration tasks where positive reward signals are very sparse. One solution is to use intrinsic motivation [10, 18, 24] to incite agents to explore unknown parts of the environment. In addition to the environment reward, agents are given an auxiliary reward related to the novelty of encountered states. Maximizing this intrinsic reward leads agents to visit previously unexplored regions of the environment, ultimately discovering new solutions to the task. These methods have shown great success in helping RL agents solve hard exploration tasks [2, 19].

In the multi-agent setting, intrinsic objectives have also been studied to induce different kinds of behaviors in agents such as coordinated exploration [8], social influence [9, 29] or alignment with other agents' expectations [13]. However, previous works have only used local observations to generate intrinsic rewards. With partial observability, local observations often lack crucial information to fully understand the current configuration of the environment. In the context of exploration, an intrinsic reward based only on local observations will lead to each agent exploring their own observation space, without considering the current state of other agents. This can result in inefficient exploration in cooperative tasks where the success of the MAS depends on the coordination of all agents.

*Contact author: maxime.toquebiau@gmail.com.

†Authors contributed equally to the paper.

In this paper, we introduce a novel multi-agent exploration approach called Joint Intrinsic Motivation (JIM) which can be combined with any MADRL algorithm that follows the centralized training with decentralized execution paradigm (CTDE). JIM exploits centralized information to motivate agents to explore new coordinated behaviors. In order to compute joint novelty, JIM builds from two state-of-the-art approaches: NovelD [35] for exploring unknown parts of the environment, and E3B [7] for having more diverse trajectories. Adding this auxiliary reward to the agents' objective incites them to diversify their collective behavior until they have a fair knowledge of the environment and can focus on the main task at hand.

To demonstrate the advantages of our approach, we first design a simple test environment to showcase a clear example of relative overgeneralization. We show that the state-of-the-art algorithm QMIX [22] struggles in this scenario and that motivating the exploration of coordinated behavior helps solve the task. Next, we validate these results in a continuous virtual environment, showing that coordination tasks benefit from joint exploration. Finally, further analysis is conducted to confirm the strength and scalability of our approach.

2 RELATED WORKS

In recent years, deep reinforcement learning techniques have been used in the context of MAS to tackle long-standing issues in multi-agent learning. Successful single-agent RL approaches have been adapted to the CTDE framework [11, 34], using a centralized value function to guide the training of decentralized policies. Recent studies have investigated the problem of credit assignment [6] in MADRL, i.e., distributing the global reward among agents based on their participation. Value factorization methods also do this implicitly [27], combining the output of local value functions into a centralized one that predicts the current value of the system. In particular, QMIX [22] uses a separate network to predict the Q-value of the joint action, given the output of local Q-values and the global state of the environment. QMIX has established itself as a long-standing state-of-the-art approach, despite its inherent limitations that several works have tried to surpass [21, 26]. However, MADRL algorithms have been shown to suffer from the problem of relative overgeneralization [31, 32]. So far, few works have addressed this problem: Wei et al. [31] propose maximum entropy RL to explore the joint-action space, and MAVEN [14] augments QMIX using a hierarchical policy to guide the exploration of joint behaviors.

A promising approach to overcome relative overgeneralization is to intrinsically motivate agents to explore their environment, ultimately discovering the optimal reward signals. In single-agent RL, curiosity has been defined to help agents solve hard exploration tasks [10, 18, 24] by rewarding the visitation of states considered as novel. For measuring novelty, several methods have used the error of trainable prediction models. The Intrinsic Curiosity Module (ICM) [19] trains a model of environment dynamics and uses the prediction error as a measure of novelty. Random Network Distillation (RND) [3] uses a target network that produces a random encoding of the state and trains a predictor network to generate the same encoding, the prediction error being the measure of novelty. The idea behind these two approaches is that the prediction

models will yield low novelty for states similar to what they have trained on while producing high novelty for unknown parts of the environment. RIDE [20] and NovelD [35] use respectively ICM and RND to compute a reward from the difference of novelty between the next state and the current state, pushing the agents to always seek novel states. Similarly, NGU [2] and E3B [7] use clustering techniques to reward states that are distant from previous states. Finally, a similar approach is proposed by AGAC [5] which trains an adversarial policy to predict the main policy's output, the latter being rewarded with the former's prediction error.

In MADRL, recent works have demonstrated the effectiveness of intrinsic rewards in promoting desirable behaviors in groups of agents. One example is social influence [9, 29] that rewards agents for performing actions that have a significant impact on other agents. Ma et al. [13] propose an intrinsic reward based on the average alignment with other agents' expectations, promoting more predictable behaviors in agents. Lupo et al. [12] propose to reward policies that perform diverse trajectories in comparison to a population of agents, which is shown to help train agents to be more versatile. Du et al. [4] use intrinsic objectives as a credit assignment technique. Finally, Iqbal and Sha [8] propose an approach for coordinated exploration using several metrics for estimating the novelty of observations that depend on all agents' past experiences. However, their model is computationally expensive and does not address the exploration of the joint-observation space, which can be problematic for hard exploration tasks where relative overgeneralization can occur.

In this paper, we address the challenge of relative overgeneralization by rewarding agents for exploring the joint-observation space. In the following sections, we will present the necessary formal background and an overview of the proposed algorithm that implements joint intrinsic motivation.

3 BACKGROUND

3.1 Dec-POMDP

To describe cooperative multi-agent tasks, we use the definition of decentralized partially-observable Markov decision process (Dec-POMDP) [16], defined as a tuple $\langle \mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{O}, \mathbf{O}, \mathbf{R}, n, \gamma \rangle$ with n being the number of agents. \mathbf{S} describes the set of global states s of the environment. \mathbf{O} is the set of joint observations, with one joint observation $\mathbf{o} = \{o_1, \dots, o_n\} \in \mathbf{O}$, and \mathbf{A} the set of joint actions, with one joint action $\mathbf{a} = \{a_1, \dots, a_n\} \in \mathbf{A}$. \mathbf{T} is the transition function defining the probability $P(s'|s, \mathbf{a})$ to transition from state s to next state s' with the joint action \mathbf{a} . \mathbf{O} is the observation function defining the probability $P(\mathbf{o}|\mathbf{a}, s')$ to observe the joint observation \mathbf{o} after taking joint action \mathbf{a} and ending up in s' . $\mathbf{R} : \mathbf{O} \times \mathbf{A} \rightarrow \mathbb{R}$ is the reward function producing at each time step the reward shared by all agents. Finally, $\gamma \in [0, 1]$ is the discount factor controlling the importance of immediate rewards against future gains.

3.2 Intrinsic rewards

In Section 2, we introduced intrinsic motivation as a way to incite agents to actively explore their environment. To this end, at each time step t , agents receive an augmented reward $r_t = r_t^{\text{ext}} + \beta r_t^{\text{int}}$, where r_t^{ext} is the extrinsic reward given by the environment, r_t^{int}

is the intrinsic reward, and β is a hyperparameter controlling the weight of the intrinsic reward in the agents' objective.

In this section, we describe three methods of intrinsic rewards from the literature that we will use later in Section 4.2.

Random Network Distillation (RND). In RND, Burda et al. [3] compute novelty using two neural networks with the same architecture: a target network ϕ and a predictor network ϕ' . The target's parameters are initialized randomly and fixed. It takes as input the state s_t and produces a random embedding $\phi(s_t)$. The predictor is trained to output the same embedding, minimizing the Euclidean distance:

$$RND_t(s_t) = \|\phi(s_t) - \phi'(s_t)\|_2. \quad (1)$$

This distance is used as a measure of the novelty of state s_t and is given as an intrinsic reward to agents.

Novelty Difference (NovelD). Zhang et al. [35] build upon RND to devise a novelty criterion termed NovelD. It is defined as follows:

$$N(s_t, s_{t+1}) = \max[RND(s_{t+1}) - \alpha RND(s_t), 0] \times \\ \times \mathbb{1}\{N_e(s_{t+1}) = 1\}, \quad (2)$$

with α a scaling factor and N_e an episodic count of visited states. The first part is the core of the novelty criterion. It uses RND to reward agents for positive gains in novelty between the current and the next states. The second part is an episodic restriction that ensures the reward is given only when state s_{t+1} is observed for the first time in this episode. This restriction limits the use of NovelD to discrete state spaces as it relies on an explicit count of visited states.

Exploration via Elliptical Episodic Bonuses (E3B). With E3B, Henaff et al. [7] propose an episodic bonus based on the position of the observed state with respect to an ellipse that fits all states previously encountered in the current episode. Formally, it is computed as follows:

$$b(s_t) = \psi(s_t)^\top C_{t-1}^{-1} \psi(s_t), \quad (3)$$

with

$$C_{t-1} = \sum_{i=1}^{t-1} \psi(s_i) \psi(s_i)^\top + \lambda I, \quad (4)$$

where I is the identity matrix and λ a scalar coefficient. ψ is an embedding network trained using an inverse dynamics model [19]: embeddings of following states $\psi(s_t)$ and $\psi(s_{t+1})$ are used by a separate neural network trained to predict the action a_t taken between these states. As a result of this training process, ψ encodes parts of the observation that are controllable by the agents (details in [7]). Intuitively, b can be understood as a generalization of a count-based episodic bonus for a continuous state space. States that are close to previously encountered states in the current episode will yield low bonuses, whereas states that are very different will produce high bonuses.

4 ALGORITHM

In this section, we introduce the Joint Intrinsic Motivation (JIM) exploration criterion for coordinated multi-agent exploration. Firstly, we describe the motivation behind our approach by providing a detailed description of the problem of relative overgeneralization.

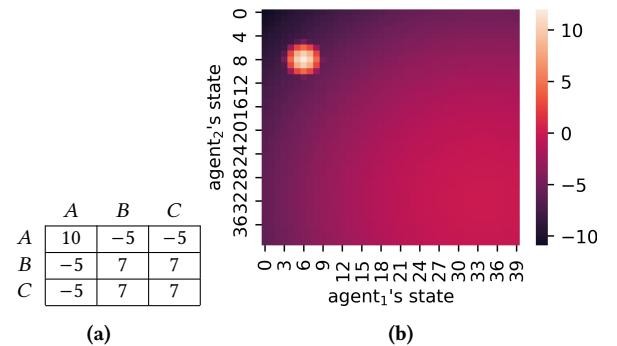


Figure 1: Two examples of relative overgeneralization: (a) payoff matrix of a social dilemma game, (b) heat-map of the reward function in the `rel_overgen` environment for two agents, with $D = 40$ and $\delta = 30$. Details in Section 6.1.

Then, we define our intrinsic reward and explain how it is used in a multi-agent setting with JIM.

4.1 The challenge of coordinated actions

Addressing hard exploration environments is challenging because of the sparse positive reward signals that exist to guide the agent's learning process. This becomes even worse with MAS as the completion of a task depends on the actions of multiple independent agents. When strong coordination is needed, agents will struggle to find the optimal strategy and settle for an easier suboptimal joint strategy, which is a problem known as relative overgeneralization [30, 32]. Figure 1a provides an example of a social dilemma game where relative overgeneralization occurs. The optimal strategy requires both agents to choose action A. But if only one agent chooses action A, the payoff is very bad. Therefore, agents will independently prefer to take actions B or C, as action A most often leads to sub-optimal outcomes.

In MAS, this can be seen as a problem of ill-coordinated exploration. As success depends on coordinated behaviors, exploration of joint policies is required in order to discover which ones lead to optimal returns. In the example of Figure 1a, exploring independent strategies will lead to ultimately choosing suboptimal actions as they individually may yield better expected returns. On the other hand, we argue that uniformly exploring joint actions would enable agents to choose optimal joint strategies more often and consequently learn more efficient individual behaviors. The approach described in the following two sections implements an algorithm that efficiently rewards agents for exploring the joint-observation space, in order to consistently find optimal strategies.

4.2 Double-timescale Intrinsic Reward

Similarly to previous works on single-agent intrinsic motivation [2], we define a novelty metric that combines two exploration criteria working at different timescales:

- A **life-long exploration criterion (LLEC)** that captures how novel is the current observation with respect to all observations since the beginning of training.

- An **episodic exploration criterion (EEC)** that captures the difference between the current observation and all previous observations in the current episode.

Intuitively, the *life-long reward* motivates agents to search for never-experienced parts of the environment. Meanwhile, the *episodic bonus* induces more diverse trajectories. These two elements will feed each other and reinforce agents to efficiently explore their environment.

Concretely, for each transition from state s_t to the next state s_{t+1} , we define the double-timescale intrinsic reward as follows:

$$r_t(s_t, s_{t+1}) = N_{LLEC}(s_t, s_{t+1}) \times N_{EBC}(s_{t+1}), \quad (5)$$

with the life-long novelty N_{LLEC} inspired from NovelD [35] (see Eq. (2)):

$$N_{LLEC}(s_t, s_{t+1}) = \max[RND(s_{t+1}) - \alpha RND(s_t), 0], \quad (6)$$

with α a scaling factor and RND the novelty measure (see Eq. (1)). Further, the episodic novelty N_{EBC} uses the bonus from E3B [7] (see Eq. (3)):

$$N_{EBC}(s_{t+1}) = \sqrt{2b(s_{t+1})}. \quad (7)$$

We remove the episodic restriction of NovelD as it relies on an episodic count of visited states. This makes it impractical in a continuous state space, as one state is very unlikely to be visited twice. Instead, we scale the life-long novelty using the elliptical episodic bonus b from E3B [7]. This bonus acts as an episodic restriction by scaling N_{LLEC} up or down, depending on the novelty of the current state compared to what has been observed during the current episode. As b provides very large bonuses and decreases very fast, we use $\sqrt{2b(s_{t+1})}$ to both smooth out large values and increase small ones.

Combining these two rewards makes it possible to take the benefits of both. N_{LLEC} pushes agents to explore regions of the state space that are not well-known to agents. Meanwhile, N_{EBC} favors diverse trajectories, inciting agents to always seek new observations during a single episode. As the agents explore their environment, the prediction error of RND (see Eq. (1)) slowly decreases. Thus, N_{LLEC} decreases as well, tending toward zero, allowing agents to progressively focus on the extrinsic reward. Finally, as the episodic restriction does not rely on any explicit count of visited states, it can be used in continuous state spaces.

4.3 The Joint Intrinsic Motivation algorithm

Building from the intrinsic reward introduced previously, we propose the Joint Intrinsic Motivation (JIM) algorithm to incite MADRL agents to explore the joint-observation space. At each time step, all agents receive the same global reward $r_t = r_t^{\text{ext}} + \beta r_t^{\text{JIM}}$, where r_t^{ext} is the extrinsic reward given by the environment, r_t^{JIM} is our joint exploration criterion, and β is a hyper-parameter controlling the weight of the intrinsic reward. The exploration criterion in JIM uses the double-timescale intrinsic reward defined earlier to compute the novelty of the joint observation:

$$r_t^{\text{JIM}}(\mathbf{o}_t, \mathbf{o}_{t+1}) = N_{LLEC}(\mathbf{o}_t, \mathbf{o}_{t+1}) \times N_{EBC}(\mathbf{o}_{t+1}), \quad (8)$$

where $\mathbf{o}_t = \{\mathbf{o}_t^i\}_{0 \leq i \leq N}$, i.e., the concatenation of all local observations. Figure 2 shows the architecture for JIM. Compared to a local method that would use one intrinsic motivation module per

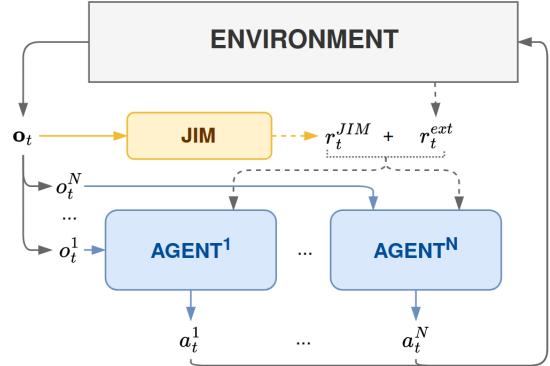


Figure 2: Architecture for the Joint Intrinsic Motivation (JIM) algorithm. JIM has only one intrinsic motivation module for the whole multi-agent system, computing novelty of the joint observation \mathbf{o}_t . However, agents only use their local observation to choose their action.

agent, JIM computes only one intrinsic reward. This requires fewer parameters and makes it possible to capture novelty at the team level, rather than at the individual level. As agents are rewarded by the novelty of the joint observation, they will learn to search for new combinations of observations with other agents of the system, rather than only exploring their local-observation space.

As JIM uses joint observations for computing the intrinsic reward, it can be associated with any MADRL algorithm that fits in the CTDE paradigm. These algorithms usually employ a centralized value function [11, 22, 34] that looks at the joint observation to predict the value of the agents' actions. Such centralized value functions will be able to associate rewards provided by JIM to new configurations in the joint observation space, thus inducing agents to actively search for these configurations.

One could note that the joint observation has two notable drawbacks: the number of dimensions grows linearly with the number of agents and there is a risk of capturing redundant information. These issues are both alleviated by using embedding networks to capture the current state of the joint observation into a more condensed latent representation. Both N_{LLEC} and N_{EBC} use embedding networks, respectively ϕ and ψ (as described in Section 3.2), to encode the joint observation. This allows for a more controllable number of parameters in JIM, as only the dimension of the input layers of ϕ and ψ depend on the size of the joint observation. Furthermore, embedding networks learn to cast away useless or redundant information in order to produce a faithful, more compact representation of the joint observation. It is important to note that both ϕ and ψ were originally used (respectively in RND [3] and E3B [7]) with raw pixel images as input, showing the significant dimensionality reduction capabilities of these techniques.

5 IMPLEMENTATION DETAILS

In the next section, we use JIM with QMIX [22]. We use the default QMIX architecture and hyperparameters, along with prioritized experience replay [23]. In all experiments, we compare three algorithms:

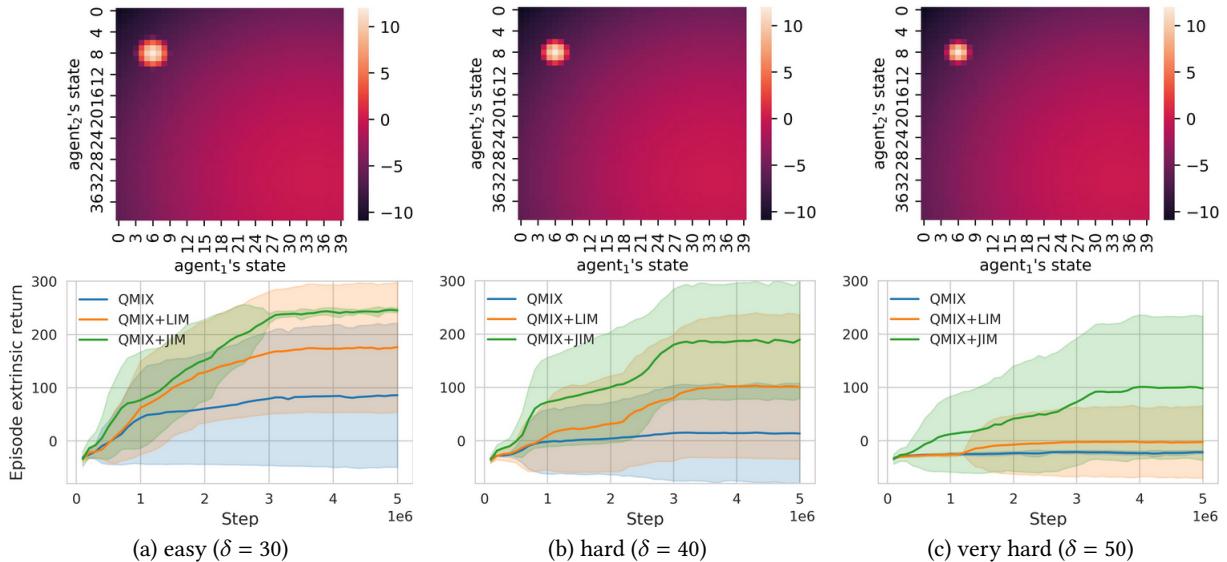


Figure 3: Performance of variants of QMIX in the `rel_overgen` environment, with three levels of difficulty. On top, we show the heat maps representing the reward function in each instance, where the difficulty is dictated by the width coefficient of the optimal reward spike δ (as defined in Eq. (9)). Increasing δ leads to a smaller optimal reward spike. Below is shown the performance during training of QMIX with no intrinsic reward (QMIX), local intrinsic motivation (QMIX+LIM), and joint intrinsic motivation (QMIX+JIM) (mean and standard deviation shown for 15 runs each). We see that a slight decrease in the size of the optimal reward spike results in a considerable increase in the difficulty of the task.

- **QMIX+JIM**, augmenting QMIX with joint exploration, as shown in Figure 2 and described in Section 4.2.
- **QMIX+LIM**, a degraded version of QMIX+JIM where local (rather than global) intrinsic motivation is used. Each agent generates its own intrinsic reward based solely on its local observation, using the same reward definition as JIM (see Section 4.2). The architecture for LIM (Local Intrinsic Motivation) is described in Appendix A.
- The original state-of-the-art **QMIX** algorithm [22] with no intrinsic motivation, used as a baseline.

Note that the only difference between these three algorithms is the definition of the reward function given to each agent during training. The actual training and execution algorithms are identical.

To ensure a fair comparison between JIM and LIM, we use different values for some specific hyperparameters (e.g., dimension of the hidden layers) in the two versions in order for them to have a similar number of trainable parameters. All hyperparameters used in our experiments are listed in Appendix B. The code used to run all experiments is freely available online¹.

6 EXPERIMENTS

In this section, we present a set of experiments to evaluate the exploration criterion of JIM when used along the state-of-the-art QMIX algorithm [22]. First, we show the results in a synthetic discrete environment where the problem of relative overgeneralization can be artificially tuned and observe that JIM helps alleviate this issue. Then, we test our approach on pseudo-realistic robotic tasks

in a continuous environment and show that exploring the joint-observation space helps solve cooperative tasks. Next, we present an ablation study by comparing JIM with two simpler versions that each lack one of the two exploration criteria described in Section 4.2, showing the advantage of combining the two. Finally, we show in Section 6.4 that JIM remains relevant when problems are scaled up.

6.1 Addressing relative overgeneralization

6.1.1 Environment definition. To demonstrate how joint exploration helps solve the problem of relative overgeneralization, we design a simple test environment that expands the example shown in Figure 1a. In this environment called `rel_overgen`, two agents can move on a discrete one-dimensional axis with D possible positions. The two agents are denoted by their position, namely x (for the first agent) and y (for the second agent). At each time step, agents observe their position as a one-hot vector (e.g., for agent x , $o_t^x = \{o_t^{x,i} = 1 \text{ if } x = i, 0 \text{ otherwise}\}_{0 \leq i < D}$) and can choose between three actions: move in one direction or the other, or stay in position. They receive a reward corresponding to their combined position:

$$r_t^{\text{ext}}(x, y; \delta) = \max\left(R^+ - \frac{\delta}{D} [(x - r_x^+)^2 + (y - r_y^+)^2], R^- - \frac{1}{8D} [(x - r_x^-)^2 + (y - r_y^-)^2]\right). \quad (9)$$

The result of this formula is displayed in Figure 1b. The reward combines two hyperbolas in opposite corners: one narrow that culminates at R^+ at position (r_x^+, r_y^+) , and another much wider that plateaus at R^- at position (r_x^-, r_y^-) . We set the optimal reward R^+ to 12 and the suboptimal R^- to 0. The width of the optimal reward

¹<https://github.com/MToquebiau/Joint-Intrinsic-Motivation>

spike is controlled by the parameter δ : a higher δ value yields a narrower spike.

The goal of the agents is to find where to go to maximize the global reward. The wide suboptimal hyperbole is deceptive as it is an obvious path for agents to minimize their loss. The optimal reward spike is difficult to find because it covers a small portion of the state space, but it guarantees much greater returns. We can vary the difficulty of the task by changing the width of this optimal reward spike: the narrower the spike, the harder it is to find.

In this environment, we expect MADRL methods to struggle to find the optimal reward spike. Exploring local states could help but would not be sufficient to consistently solve the task. As the dimension D of the local-state space is fairly small, novelty rewards will quickly vanish and will not help agents find the optimal reward spike. Exploring the joint-observation space adequately is required in order to consistently find optimal rewards. As JIM will reward exploration until all combined positions (x, y) are visited several times, agents will visit the optimal reward spike more often, thus helping them to learn the optimal coordinated strategy.

6.1.2 Results. The results shown in Figure 3 confirm the hypotheses formulated in the previous section. We show the performance of QMIX, QMIX+LIM, and QMIX+JIM across 15 independent runs each. Further, we present results in three difficulty levels dictated by the width of the optimal reward spike. The results clearly demonstrate the importance of exploring the joint-state space. QMIX alone manages to get a positive reward on the easy scenario, but its performance is both lower and with a larger standard deviation compared to the two other algorithms. In the harder scenarios, QMIX's performance degrades strongly, never finding any positive reward in the hardest case. JIM clearly improves the performance. In the easy scenario, QMIX+JIM consistently goes for the optimal reward spike. In the harder settings, it still performs well on average, even in the "very hard" scenario where the optimal reward spike covers only 0.013% of all combined positions. The results of QMIX+LIM show that exploring the local-observation space helps agents find the optimal reward spike more often. However, it performs worse than JIM as it does not ensure that all combined positions are sufficiently explored. This shows that exploring the joint-observation space is crucial to allow agents to discover optimal coordinated behaviors.

6.2 Coordination tasks in a continuous environment

6.2.1 Environment definition and setups. Next, we study how JIM scales to more realistic continuous environments and more complex tasks. We use the multi-agent particle environment² (MPE) [11, 15] to simulate cooperative robotic tasks that require a high degree of coordination. The state space of MPE is continuous: in our setups, agents receive as observation a vector with their position in the two-dimensional space and, for all the other entities in the environment, their relative position and velocity. Agents navigate in a closed two-by-two-meter area by choosing between five discrete actions: move in any four cardinal directions or stay in place.

The first task is a cooperative box-pushing task that requires agents to push an object and place it on top of a landmark. Figure 4a

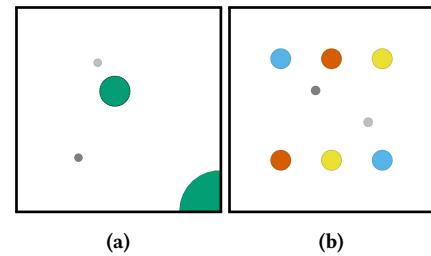


Figure 4: Screenshots of our custom tasks in the MPE, agents are the small grey circles. (a) cooperative box pushing scenario: agents must deliver the object (green circle in the middle) to the landmark in the bottom right corner. (b) coordinated placement scenario: agents must navigate to position themselves on the colored circles representing landmarks.

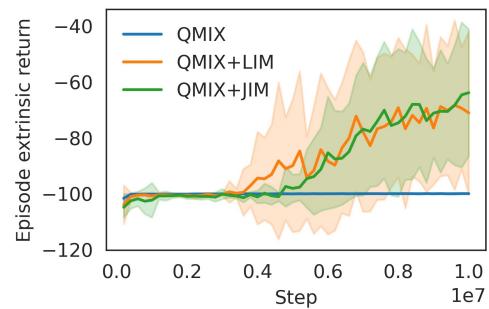


Figure 5: Training curves of the three variants of QMIX in the cooperative box pushing task, with the mean and standard deviation across 11 runs each.

shows a screenshot of this scenario. At the start of each episode, the landmark is randomly placed in any one of the four corners. The initial positions of the agents and the object are randomly set. If the agents manage to push the object and place it on the landmark, the episode ends and they receive a reward of +100. Agents also get a small penalty of -1 at each time step to reward faster strategies. This reward is purposefully defined to be very sparse, in order to study how exploration helps in this kind of situation.

The second scenario is a cooperative placement task where agents must position themselves over landmarks in order to maximize their reward. As shown in Figure 4b, there are two sets of three colored landmarks. The reward given at each time step depends on the placement of the agents on the landmarks. The optimal state is having both agents placed on the red landmarks, yielding a reward of +10 at each time step. The blue and yellow landmarks act as deceiving rewards, yielding much smaller rewards (+2 for blue, +1 for yellow). To increase the deceiving aspect of the blue and yellow landmarks, we also reward agents collectively by +0.5 if only one of them stands on one of these two colors. This leads to a need for coordination between agents, as they will locally find that going on blue or yellow landmarks systematically leads to a small reward. Only if agents explore their environment well, will they discover that they need to be both on red to get the optimal reward signal. Importantly, this scenario features partial observability, with

²<https://github.com/openai/multiagent-particle-envs>

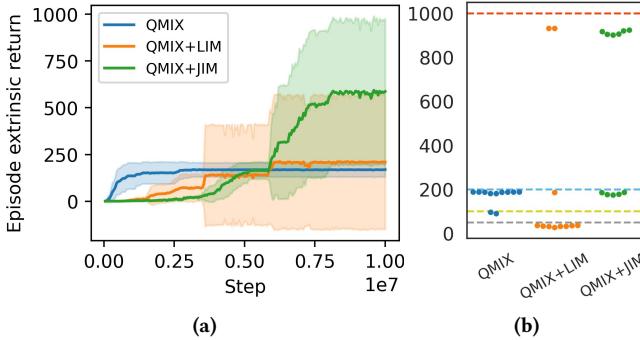


Figure 6: Performance of the three variants of QMIX in the coordinated placement task. (a) shows the training curves with the mean and standard deviation across 11 independent runs each, while (b) displays the performance of each independent run at the last iteration of training. Dashed lines on (b) indicate the optimal (unattainable) level of return obtained with different strategies: red, blue, and yellow lines represent the return obtained if both agents are on landmarks of the related color during 100 steps (the duration of an episode), and the grey line similarly describes only one agent being either on blue or yellow.

agents only having information about entities close to them (less than sixty centimeters from them), the others being masked with neutral values. This means that agents do not necessarily see which landmark the other agent goes to³.

6.2.2 Results. Results of training in the cooperative box pushing scenario are shown in Figure 5 (median and confidence interval shown for 11 runs each). First, we observe that QMIX alone performs very poorly as it is unable to find the solution to the task. The high sparsity of the reward function makes it impossible for agents to discover the objective with random exploration of the environment. Second, we see that JIM and LIM achieve similar levels of performance. While coordination can help agents perform well, it is actually not a requirement for this task. In fact, one agent alone is able to push the object and place it on the landmark. Thus, exploring the space of joint configurations is not helpful in this scenario. This shows however that actively exploring the environment is crucial in tasks where the reward function is very sparse.

Conversely, experiments in the coordinated placement task demonstrate well the importance of exploring jointly. Figure 6a shows the training curves of QMIX, QMIX+LIM, and QMIX+JIM, with 11 independent runs each. Figure 6b shows the performance of each run at the last iteration of training. The colored dashed lines give an insight into the level of strategy learned by each run. These levels of strategy can be visualized with example trajectories displayed in Figure 8. QMIX alone almost always goes for the blue landmarks, while sometimes settling for the yellow ones. This indicates that without actively exploring the environment, QMIX gets stuck because of deceptive rewards and is unable to find the optimal strategy. While QMIX+LIM seems slightly better than QMIX on the training curves, the individual run performance shown in Figure

³See Appendix C for a detailed description of these two tasks.

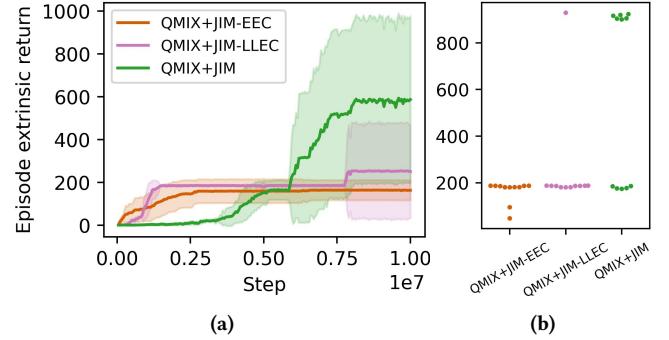


Figure 7: Ablation study of JIM in the coordinated placement task. (a) shows the training curves with the mean and standard deviation across 11 independent runs each, while (b) displays the performance of each independent run at the last iteration of training. The two ablated versions only feature one of the two exploration criteria defined in Section 4.2: JIM-EEC for N_{EEC} and JIM-LLEC for N_{LLEC} . The results show the importance of combining the two criteria.

6b shows that LIM arguably performs worse. Two runs manage to find the optimal strategy, but LIM often performs poorly with only one agent on a blue or yellow landmark. This demonstrates that exploring the space of local observations can be helpful, as it pushes agents to explore the environment. However, exploring local observations can also be misleading as they do not contain all the information about the current state of the environment. With JIM, exploring the joint-observation space clearly improves the quality of the chosen strategies. More than half of the time, QMIX+JIM finds the optimal reward signal and learns an effective strategy to go on red landmarks, showing that JIM allows for more efficient exploration of coordinated behaviors. When agents do not find the optimal strategy, they stick with the best sub-optimal strategy to go both on blue. This shows that agents benefit from exploring the space of joint observations as they are directly linked to the obtained reward, whereas local observations lack crucial information to understand the global reward.

Another advantage of JIM is the simplicity of its architecture. While LIM and similar approaches in previous works [4, 8, 29] require computing one intrinsic reward for each agent, JIM only computes one intrinsic reward for the whole group of agents. This makes JIM significantly more efficient to run, with LIM being approximately 24% slower than JIM to train (see Appendix D for details).

6.3 Ablation study

In this ablation study, JIM is compared with two ablated versions of the reward: one with only the *episodic exploration criterion* N_{EEC} (JIM-EEC) and one with only the *life-long exploration criterion* N_{LLEC} (JIM-LLEC). Note that JIM-LLEC is actually equivalent to NovelID [35] in this environment as the episodic restriction of NovelID (see Section 3.2) would be ineffective in a continuous environment such as MPE. To compare these three versions properly, we

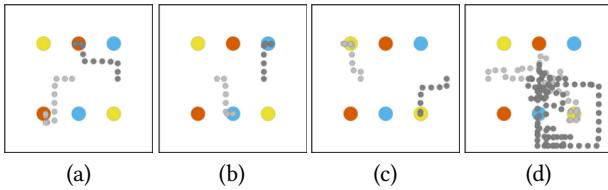


Figure 8: Examples of trajectories in the coordinated placement task. They present different levels of strategy, with (a) > (b) > (c) > (d): (a) optimal strategy with both agents on red, (b) both on blue, (c) both on yellow, and (d) one on blue/yellow.

scale the intrinsic rewards of the two ablated models to be at a similar magnitude as the intrinsic reward generated by JIM. Figure 7 shows the results of training these versions in the coordinated placement task, with 11 independent runs each. Both ablated algorithms perform significantly worse than JIM. First, the episodic bonus of JIM-EEC alone lacks the motivation for discovering unseen configurations. Thus, it explores less and is not able to find the optimal solution to the task. Meanwhile, without the episodic restriction, JIM-LLEC develops less efficient exploration strategies. This confirms that, as shown in a recent study [1], the episodic restriction implemented in NovelD and other intrinsic rewards [2, 20] is crucial for developing efficient exploration strategies. Overall, this proves the importance of combining the two stages of exploration defined in N_{LLEC} and N_{EEC} .

6.4 Scaling up to more agents

Finally, we evaluate JIM in a scenario with four agents using a modified version of the `rel_overgen` environment introduced in Section 6.1 (see Appendix E for more details). As in the two-agent version, the reward function has a high reward spike in one corner of the space and a low reward plateau in the other corner, making relative overgeneralization prone to arise. With more agents, the number of dimensions of the joint observation increases linearly (in this case: from 80 dimensions with two agents to 160 with four agents) while the number of possible states increases exponentially, making the search for the optimal strategy significantly more challenging.

Figure 9 shows the results in this scenario for 11 independent runs each. As with previous experiments, JIM improves the performance of QMIX by upgrading its exploration capabilities. Taking a closer look at the results reveals that QMIX and QMIX+LIM find the optimal solution in respectively 2 and 0 runs out of the 11, while QMIX+JIM finds the optimal solution in 8 out of 11 runs in the allocated time (cf. Figure 9-bottom). The two positive results for QMIX can be attributed to beneficial initial conditions as QMIX never reaches the optimal performance otherwise. This is not the case with QMIX+JIM, which shows robustness to initial conditions thanks to active exploration of the environment. In fact, the impact of JIM becomes evident when looking at curves from individual runs, where we consistently observe significant performance enhancements subsequent to a minor initial drop in efficiency. This phenomenon reflects a deliberate shift towards exploring novel approaches when the system would otherwise remain stagnant. This is unique to JIM, as QMIX+LIM never succeeds in finding

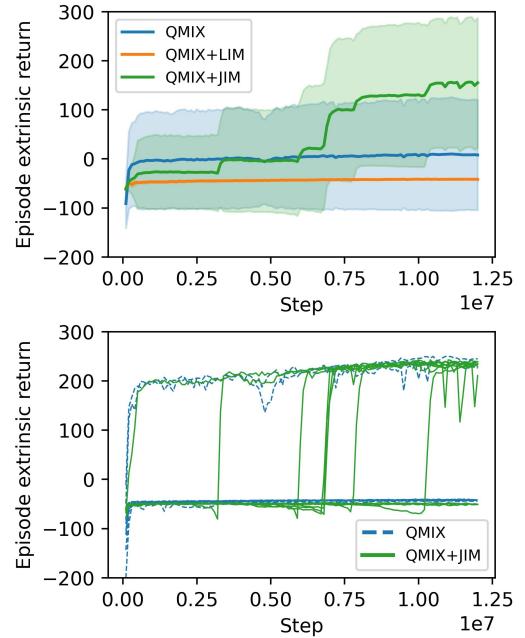


Figure 9: Training performance of QMIX, QMIX+LIM and QMIX+JIM in the four-agent version of `rel_overgen`. Top graph shows the mean and standard deviation across 11 runs each, bottom graph displays all single runs (QMIX in dotted lines for clarity, QMIX+LIM is omitted as it never goes beyond the suboptimal strategy).

the optimal solution, advocating for the benefits of using a global, rather than local, intrinsic reward for exploration.

7 CONCLUSION

In this paper, we present an algorithm for joint intrinsic motivation (JIM), which is the first method to reward active exploration of the joint-observation space. It can be integrated to enhance any Multi-Agent Deep Reinforcement Learning algorithm that uses centralized training with decentralized execution. By combining JIM with the state-of-the-art QMIX algorithm, we demonstrate that it outperforms the original QMIX implementation as well as a modified QMIX algorithm using single-agent intrinsic rewards. We show that active exploration is a key component for multi-agent learning in environments with sparse rewards. Moreover, joint exploration enables the discovery of optimal coordinated behaviors that would be hard to find otherwise as they necessitate a high level of coordination between agents.

This shows the importance of using joint observations in the process of computing intrinsic rewards for a multi-agent system. In fact, the joint observation is the best estimate of the global state of the environment available for the agents. Using it allows more efficient learning of multi-agent joint behaviors and is computationally less expensive than having to compute local intrinsic rewards for each agent. These results should encourage research on how joint observations can be used in other kinds of intrinsic rewards to shape the agents' behavior further.

REFERENCES

- [1] Alain Andres, Esther Villar-Rodriguez, and Javier Del Ser. 2022. An Evaluation Study of Intrinsic Motivation Techniques applied to Reinforcement Learning over Hard Exploration Environments. In *arXiv:2205.11184*.
- [2] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martin Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. 2020. Never Give Up: Learning Directed Exploration Strategies. In *8th International Conference on Learning Representations*. <https://openreview.net/forum?id=Sye57xStvB>
- [3] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2019. Exploration by Random Network Distillation. In *7th International Conference on Learning Representations*.
- [4] Yali Du, Lei Han, Meng Fang, Tianhong Dai, Ji Liu, and Dacheng Tao. 2019. LIIR: Learning Individual Intrinsic Reward in Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 32. <https://proceedings.neurips.cc/paper/2019/hash/07a9d3fed4c5ea6b17e80258dee231fa-Abstract.html>
- [5] Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. 2021. Adversarially Guided Actor-Critic. In *9th International Conference on Learning Representations*. <https://hal.inria.fr/hal-03167169>
- [6] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. <https://ojs.aaai.org/index.php/AAAI/article/view/11794>
- [7] Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. 2022. Exploration via Elliptical Episodic Bonuses. In *Advances in Neural Information Processing Systems*, Vol. 35. 37631–37646. <https://openreview.net/forum?id=XgyZos9jQ>
- [8] Shariq Iqbal and Fei Sha. 2019. Coordinated Exploration via Intrinsic Rewards for Multi-Agent Reinforcement Learning. In *arXiv:1905.12127*. <https://openreview.net/forum?id=rkltE0VKh>
- [9] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, and Nando de Freitas. 2019. Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97. 3040–3049.
- [10] Joel Lehman and Kenneth O. Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. In *Evolutionary Computation*, Vol. 19. 189–223.
- [11] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, Vol. 30.
- [12] Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. 2021. Trajectory Diversity for Zero-Shot Coordination. In *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139. 7204–7213. <https://proceedings.mlr.press/v139/lupu21a.html>
- [13] Zixian Ma, Rose E Wang, Li Fei-Fei, Michael S. Bernstein, and Ranjay Krishna. 2022. ELIGN: Expectation Alignment as a Multi-Agent Intrinsic Reward. In *Advances in Neural Information Processing Systems*, Vol. 35. 8304–8317. <https://openreview.net/forum?id=uPyNR2yPoe>
- [14] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-Agent Variational Exploration. In *Advances in Neural Information Processing Systems*, Vol. 32. <https://proceedings.neurips.cc/paper/2019/file/f816dc0acf7498e10496222c9db10-Paper.pdf>
- [15] Igor Mordatch and Pieter Abbeel. 2018. Emergence of Grounded Compositional Language in Multi-Agent Populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*. <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17007>
- [16] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer. <https://www.ccs.northeastern.edu/home/camato/publications/OliehoekAmato16book.pdf>
- [17] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafał Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki,
- [18] Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. In *arXiv:1912.06680*.
- [19] Pierre-Yves Oudeyer and Frederic Kaplan. 2007. What is Intrinsic Motivation? A Typology of Computational Approaches. In *Frontiers in neurorobotics*, Vol. 1. 6. <https://doi.org/10.3389/neuro.12.006.2007>
- [20] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. PMLR 70. 2778–2787.
- [21] Roberta Raileanu and Tim Rocktäschel. 2020. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. In *9th International Conference on Learning Representations*. <https://openreview.net/forum?id=rkg-TJBFB>
- [22] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems* 33. 10199–10210. <https://proceedings.neurips.cc/paper/2020/hash/73a427badbe0e32caa2e1fc7530b7f3-Abstract.html>
- [23] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *4th International Conference on Learning Representations*.
- [24] Jürgen Schmidhuber. 1991. A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers. In *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*. 222–227.
- [25] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. In *arXiv:1610.03295*.
- [26] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, Vol. PMLR 97. 5887–5896.
- [27] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vincenzo Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2085–2087.
- [28] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In *Proceedings of the Tenth International Conference on Machine Learning*. 330–337.
- [29] Tonghan Wang, Jianhua Wang, Yi Wu, and Chongjie Zhang. 2020. Influence-Based Multi-Agent Exploration. In *8th International Conference on Learning Representations*. <https://openreview.net/forum?id=Bjgy96Eyvr>
- [30] Ermo Wei and Sean Luke. 2016. Lenient Learning in Independent-Learner Stochastic Cooperative Games. In *The Journal of Machine Learning Research*, Vol. 17. 2914–2955.
- [31] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. 2018. Multiagent Soft Q-Learning. In *arXiv:1804.09817*.
- [32] Rudolph Paul Wiegand. 2003. *An Analysis of Cooperative Coevolutionary Algorithms*. Ph.D. Dissertation. George Mason University.
- [33] Michael Wooldridge. 2009. *An introduction to multiagent systems*. John Wiley & Sons.
- [34] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. In *arXiv:2103.01955*.
- [35] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. 2021. NovelD: A Simple yet Effective Exploration Criterion. In *Advances in Neural Information Processing Systems*, Vol. 34. 25217–25230. <https://proceedings.neurips.cc/paper/2021/file/d428d070622e0f4363fceae11f4a3576-Paper.pdf>

APPENDICES

A LOCAL INTRINSIC MOTIVATION

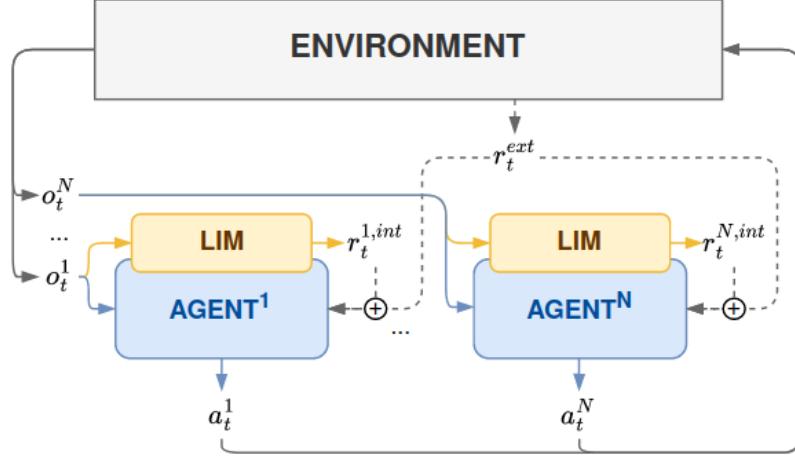


Figure 10: Architecture for local intrinsic motivation (LIM). Each agent has its own module for computing an intrinsic reward based on its local observation.

B HYPERPARAMETERS

Table 1: Hyperparameters used in the `rel_overgen` environment.

Hyperparameter	Algorithm		
	JIM	JIM 4 agents	LIM
Intrinsic reward weight β	1		
Encoding dim $D_{\phi/\psi}$	64	64	32
Hidden dim D_{hidden}	128	256	64
Scaling factor α	0.5, 0.6		
Intrinsic reward learning rate α_{int}	0.0001	0.0001, 0.0002	0.0001

Table 2: Hyperparameters used in the cooperative box pushing scenario.

Hyperparameter	Algorithm	
	JIM	LIM
Intrinsic reward weight β	1	1
Encoding dim $D_{\phi/\psi}$	64	32
Hidden dim D_{hidden}	128	64
Scaling factor α	0.5	
Intrinsic reward learning rate α_{int}	0.0001	

Table 3: Hyperparameters used in the coordinated placement scenario.

Hyperparameter	Algorithm			
	JIM	LIM	JIM-LLEC	JIM-EEC
Intrinsic reward weight β	1, 2, 4	1, 4, 8	1, 3	0.1, 1
Encoding dim $D_{\phi/\psi}$	64	36	64	64
Hidden dim D_{hidden}	512	256	512	512
Scaling factor α	0.5			
Intrinsic reward learning rate α_{int}	0.0001			

Tables 1 to 3 present the values chosen for hyperparameters. We only list hyperparameters specific to the intrinsic reward module, as for QMIX we use the default hyperparameters described in the original paper [22]. When we performed some search over a specific hyperparameter, we put the list of all the values we tried and put the best one in bold. Here, we detail the different parameters presented:

- Intrinsic reward weight β : weight of the intrinsic reward r_i against the extrinsic reward r_e in the reward given to agents: $r_t = r_t^e + \beta r_t^{int}$.
- Encoding dimension $D_{\phi/\psi}$: dimension of the output of the embedding networks ϕ and ψ used in N_{LLEC} and N_{EEC} respectively (see Section 3).

- Hidden dimension D_{hidden} : dimension of the hidden layers in the embedding network ϕ and ψ used in N_{LLEC} and N_{EEC} respectively.
- Scaling factor α : parameter used in the definition of N_{LLEC} (see Eq. (6)). It controls how much novelty gain we want agents to find between each step.
- Intrinsic reward learning rate α_{int} : learning rate used for training the intrinsic reward module.

C DETAILS ON CUSTOM TASKS IN THE MULTI-AGENT PARTICLE ENVIRONMENT

C.1 Cooperative box pushing

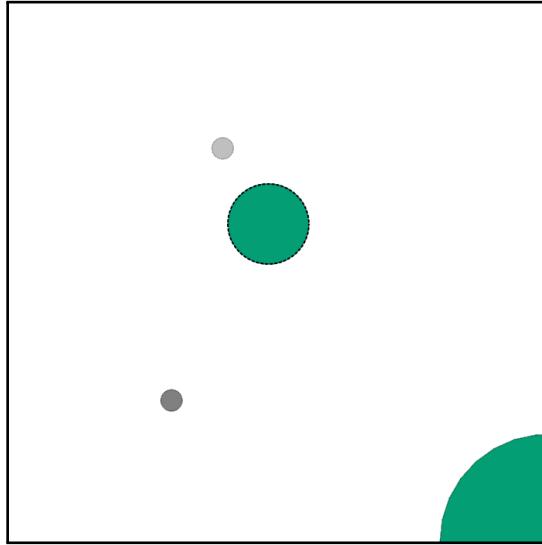


Figure 11: Cooperative box pushing task, agents are the small grey circles, the green circle in the middle is an object to deliver to the landmark in the bottom right corner.

The cooperative box pushing task requires pushing a round object and placing it on top of a landmark. The environment is a 2x2 meter area with walls on the sides that block entities from moving away. At each time step, an agent gets as observation:

- its personal data: its position and velocity $pos_{self,x}, pos_{self,y}, vel_{self,x}, vel_{self,y}$,
- data about the other agent: a boolean indicating if the other agent is visible or not, the relative position, and the velocity of this agent $is_visible_{agent}, dist_{agent,x}, dist_{agent,y}, vel_{agent,x}, vel_{agent,y}$,
- data about the object: a boolean indicating if the object is visible or not, the relative position, and the velocity of this object: $is_visible_{object}, dist_{object,x}, dist_{object,y}, vel_{object,x}, vel_{object,y}$,
- and data about the landmark: a boolean indicating if the landmark is visible or not and the number of the corner it is located into (from 1 to 4): $is_visible_{landmark}, corner_{landmark}$.

Thus, the observation is a vector of dimension 16 containing this information. Relative positions of other entities (agent or object) are actually the distance to the agent, normalized by their range of observation, i.e.,

$$dist_{agent,x} = \frac{pos_{agent,x} - pos_{self,x}}{obs_range}.$$

The environment can be either fully observable or partially observable. In the latter case, agents have a range of observation of 60 centimeters around them. When agents or objects are outside this range, their relative position is masked with ones and their velocity with zeros. When the landmark is outside the range of observation, the corner number is masked with zero. In the fully observable case, the observation range is set to 2.83 meters, i.e., the largest distance possible to have in this 2x2 meter area.

The reward function is very sparse. At each time step, agents receive a penalty of 0.1, plus a penalty of 2 if there is a collision between agents. If the task is completed, i.e., the center of the object is placed in the area of the landmark, the agents get a reward of 100 and the episode ends.

At the start of each episode, the positions of all entities in the environment are randomly set: the agents and the object are randomly placed inside the environment, and the landmark is placed in one of the four corners of the map.

C.2 Coordinated placement

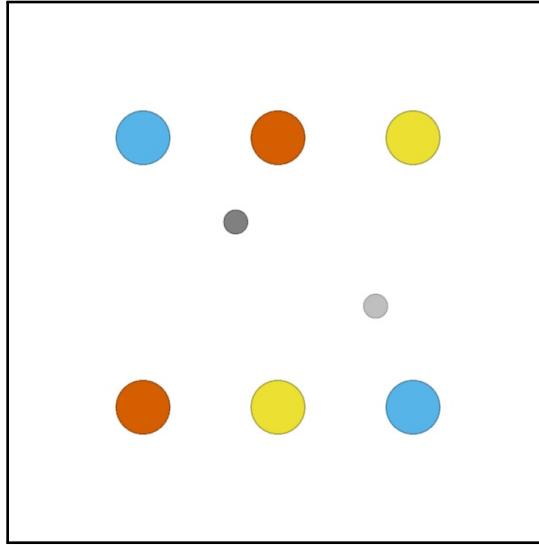


Figure 12: Coordinated placement task, agents are the small grey circles, the colored circles represent landmarks the agents have to navigate on to gain rewards.

The coordinated placement task requires navigating on top of landmarks and choosing the right landmark colors in order to maximize the obtained reward. The environment is a 2x2 meter area with walls on the sides that block entities from moving away. At each time step, an agent gets as observation:

- its personal data: its position and velocity $\text{pos}_{\text{self},x}$, $\text{pos}_{\text{self},y}$, $\text{vel}_{\text{self},x}$, $\text{vel}_{\text{self},y}$,
- data about the other agent: a boolean indicating if the other agent is visible or not, the relative position, and the velocity of this agent $\text{is_visible}_{\text{agent}}$, $\text{dist}_{\text{agent},x}$, $\text{dist}_{\text{agent},y}$, $\text{vel}_{\text{agent},x}$, $\text{vel}_{\text{agent},y}$,
- for each landmark in the environment: a boolean indicating if the landmark is visible or not, the relative position of this landmark, and its color as a one-hot encoding: $\text{is_visible}_{\text{landmark}}$, $\text{dist}_{\text{landmark},x}$, $\text{dist}_{\text{landmark},y}$, is_red , is_blue , is_yellow .

Thus, the observation is a vector of dimension 43 containing this information. Relative positions of other entities (agent or landmarks) are actually the distance to the agent, normalized by their range of observation, i.e.,

$$\text{dist}_{\text{agent},x} = \frac{\text{pos}_{\text{agent},x} - \text{pos}_{\text{self},x}}{\text{obs_range}}.$$

This scenario is partially observable. Agents have a range of observation of 60 centimeters around them. When agents or objects are outside this range, their relative position is masked with ones and their velocity with zeros. For landmarks outside of the observation range, the color is masked with zeros.

The reward given at each time step depends only on which landmark has an agent placed on top:

- if two agents are placed on top of red landmarks, $r_t^e = 10$,
- if two agents are placed on top of blue landmarks, $r_t^e = 2$,
- if two agents are placed on top of yellow landmarks, $r_t^e = 1$,
- if only one agent is placed on top of either a blue or yellow landmark, $r_t^e = 0.5$,
- else, $r_t^e = 0$.

At the start of each episode, agents are placed randomly on the horizontal line in the middle of the map, i.e., $\text{pos} = (x = \text{uniform}(-1, 1), y = 0)$. The landmarks are always in the same positions, with the colors in the same order, as displayed in Figure 12.

D EXECUTION TIMES

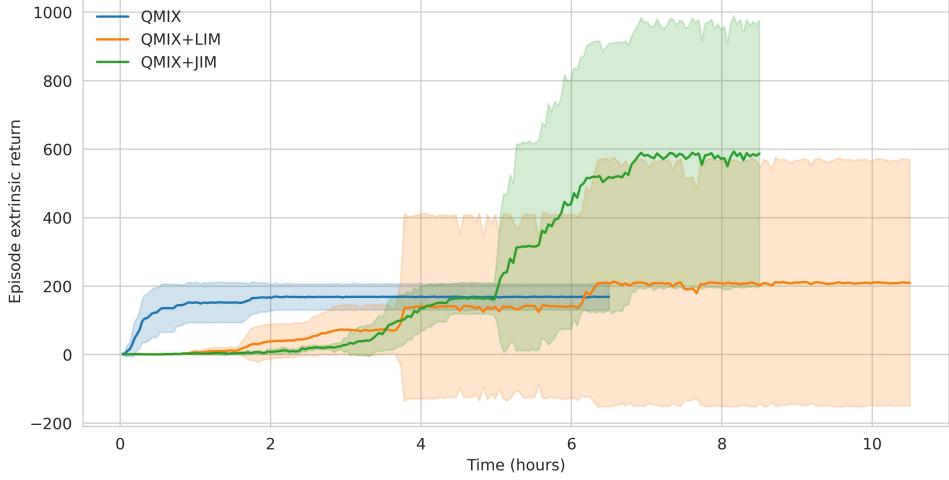


Figure 13: Training curves of QMIX, QMIX+LIM, and QMIX+JIM in the coordinated placement task with execution time on the x-axis. QMIX takes on average 6.5 hours to train during 10 million steps, while QMIX+JIM takes 8.5 hours and QMIX+LIM 10.5 hours.

E N-AGENT rel_overgen ENVIRONMENT

To study the problem of relative overgeneralization with more than two agents, we extend the `rel_overgen` environment to accept N agents. To do so, we modify the reward definition given in the paper (see Eq. (9)):

$$r_t^{\text{ext}}(\mathbf{p}; \delta) = \max \left(R^+ - \frac{\delta}{D} \sum_{i=0}^N (p_i - r_i^+)^2, R^- - \frac{1}{8D} \sum_{i=0}^N (p_i - r_i^-)^2 \right),$$

with $\mathbf{p} = \{p_i\}_{0 < i \leq N}$ the positions of the agents, δ the coefficient controlling the size of the optimal reward spike, D the dimension of each agent's state, R^+ the maximum value of the optimal reward spike placed at position $\mathbf{r}^+ = \{r_i^+\}_{0 < i \leq N}$ and R^- the maximum value of the suboptimal plateau placed at position $\mathbf{r}^- = \{r_i^-\}_{0 < i \leq N}$. This formula yields the same results as the two-dimensional examples shown in the paper but in a N -dimensional space.

Adding agents increases the complexity of the task exponentially. To compensate for this, we have to make the optimal reward spike larger for the task to be solvable by QMIX. In the 4-agent experiments, we use $\delta = 0.9$.

Finally, with four agents we had to lower the initial value of the ϵ parameter of QMIX for its ϵ -greedy strategy. We found that increasing the number of agents led to bad results with the default 0.3 initial value of ϵ . With this hyperparameter set to 0.1, the results were significantly better. This is likely due to the fact that agents choose separately if they explore or exploit (at least that is the case in our implementation), meaning that increasing the number of agents leads to having more randomness in the selection of each joint action.