Hybrid Independent Learning in Cooperative Markov Games

Roi Yehoshua(✉) and Christopher Amato

Northeastern University, Boston, MA 02115, USA
r.yehoshua@northeastern.edu

Abstract. Independent agents learning by reinforcement must overcome several difficulties, including non-stationarity, miscoordination, and relative overgeneralization. An independent learner may receive different rewards for the same state and action at different time steps, depending on the actions of the other agents in that state. Existing multi-agent learning methods try to overcome these issues by using various techniques, such as hysteresis or leniency. However, they all use the latest reward signal to update the Q function. Instead, we propose to keep track of the rewards received for each state-action pair, and use a hybrid approach for updating the Q values: the agents initially adopt an optimistic disposition by using the maximum reward observed, and then transform into average reward learners. We show both analytically and empirically that this technique can improve the convergence and stability of the learning, and is able to deal robustly with overgeneralization, miscoordination, and high degree of stochasticity in the reward and transition functions. Our method outperforms state-of-the-art multi-agent learning algorithms across a spectrum of stochastic and partially observable games, while requiring little parameter tuning.

Keywords: Multi-agent reinforcement learning · Markov games · Independent learners · Distributed Q-learning

1 Introduction

Many real-world problems involve environments that contain a group of independently learning agents working towards a common goal. Examples include air traffic control [1], adaptive load-balancing of parallel applications [14], and multi-robot systems [19]. Reinforcement learning (RL) [13] has been a popular approach to solving multi-agent learning problems, since it does not require any a-priori knowledge about the dynamics of the environment, which can be stochastic and partially observable.

Two main multi-agent reinforcement learning (MARL) approaches have been proposed for handling multi-agent domains [3]. In joint-action learners (JALs), every agent is aware of the actions of the other agents. Thus, the state and action spaces of all agents can be merged together, defining a sort of super-agent in the

M. E. Taylor et al. (Eds.): DAI 2020, LNAI 12547, pp. 69–84, 2020.
https://doi.org/10.1007/978-3-030-64096-5_6

joint space. Any single-agent RL algorithm can be used to learn the optimal joint policy for such super agent. On the other hand, independent learners (ILs) do not know the actions the other agents are performing, thus each agent has to learn and act independently. Independent learning is an appealing approach, since it does not have the scalability problem with increasing the number of agents, and each agent only needs its local history of observations during the training and the inference time.

However, independent learners must overcome a number of pathologies in order to learn optimal joint policies [4,8]. These include credit assignment, the moving target problem, stochasticity, relative overgeneralization, miscoordination, and the alter-exploration problem. Due to these pathologies, reinforcement learning algorithms that converge in a single-agent setting often fail in cooperative multi-agent systems with independent learning agents.

Various methods have been proposed in the MARL literature to help ILs cope with the outlined pathologies. However, addressing one pathology often leaves agents vulnerable towards others. Distributed Q-learning [5] is an optimistic learning approach, which only increases the Q-values and ignores negative updates. It solves the relative overgeneralization and miscoordination problems, but is vulnerable to games with stochastic rewards or transitions. Other approaches, such as hysteretic Q-learning [8] and lenient learning [12] are based on varying the "degree of optimism" of agents. Hysteretic Q-learning uses a variable learning rate to minimize the influence of poor actions of other agents on the agent's own learning, while lenient Q-learning initially ignore rewards that lead to negative updates, until the state-action pair has been encountered frequently enough (see Sect. 5 for a more detailed description).

All previous methods use the latest joint reward signal received for each state-action pair to update the Q function. Since the joint reward depends both on the behavior of the other agents (which initially are mostly exploring the environment), and the noise in the environment, updating the Q values and the corresponding policy using only the latest reward signal may destabilize the learning.

Instead, we propose to keep track of the rewards received for each state-action pair, and use the reward statistics for the policy updates. Similar to the hysteretic and lenient approaches, we use a hybrid approach for the learning: initially, the agents adopt an optimistic approach by using the maximum reward observed for the updates, which helps minimize the effect of shadowed equilibria. After a predefined period of time, the agents transform into average reward learners in order to handle stochasticity. We call this approach *Hybrid Q-learning*.

We first analyze the Hybrid Q-learning algorithm, and show that it is able to handle each of the outlined MARL pathologies. We then compare it against baseline algorithms from the literature, including traditional Q-learning, Distributed Q-learning [5], Hysteretic Q-learning [8], and a recent version of Lenient Q-Learning called LMRL2 [18]. For the comparison we use a diverse array of situations, including: stochastic and repeated games, deterministic and stochastic rewards, stochastic state transition functions, partial observability, miscoordi-

nation, and relative overgeneralization. Our method outperforms the baseline algorithms on all tested problems.

2 Theoretical Framework

2.1 Markov Games

Markov games (also called Stochastic games) [7] are the foundation for much of the research in multi-agent reinforcement learning. Markov games are a superset of Markov decision processes (MDPs) and matrix games, including both multiple agents and multiple states.

Formally, a Markov game consists of a tuple $\langle N, S, A, T, R \rangle$ where:

- N is a finite set of agents, $n = |N|$
- S is a set of states
- $A = A_1 \times ... \times A_n$ is the set of joint actions.
- $T : S \times A \times S \rightarrow [0, 1]$ is the transition probability function, with $T(s, a, s') = P(s'|s, \boldsymbol{a})$ being the probability of transitioning to state s' after taking joint action $\boldsymbol{a}$ in state s.
- $R : S \times A \rightarrow \mathbb{R}^n$ is the joint reward function, with $R_i(s, \boldsymbol{a})$ being the reward of agent i in state s after taking the joint action $\boldsymbol{a}$.

In a Markov game, each agent is independently choosing actions and receiving rewards, while the state transitions with respect to the full joint-action of all agents. If all agents receive the same rewards ($R_1 = ... = R_n = R$) the Markov game is *fully-cooperative*.

2.2 Policies and Nash Equilibria

For agent i, the policy π_i represents a mapping from a state to a probability distribution over actions: $\pi_i : S \times A \rightarrow [0, 1]$. A solution to a stochastic game is a *joint policy* π—a set of policies, one for each agent. Joint policies excluding agent i are defined as π_{-i}. The notation $\langle \pi_i, \pi_{-i} \rangle$ refers to a joint policy with agent i following π_i while the remaining agents follow π_{-i}.

Given a joint policy π, the value (or gain) of state s for agent i is defined as the expected sum of discounted future rewards, when all agents follow π from state s:

$$V^{\pi_i}(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{i,t+k} \middle| s_t = s \right] \tag{1}$$

where $r_{i,t}$ is the reward received by agent i at time step t, and $\gamma \in (0, 1]$ is a discount factor.

From a game theoretic point of view, two common concepts of equilibrium are used to define solutions in Markov games. A joint policy π^* is a *Nash equilibrium*

[10], if and only if no agent can improve its gain by unilaterally deviating from π^*, i.e., for agent i

$$\forall \pi_i, \forall s \in S, V^{\langle \pi_i^*, \pi_{-i}^* \rangle}(s) \geq V^{\langle \pi_i, \pi_{-i}^* \rangle}(s) \tag{2}$$

In contrast, a Pareto-optimal equilibrium is a joint policy $\hat{\pi}$ from which no agent can improve its gain without decreasing the expected gain of any other agent. In cooperative games, every Parto-optimal equilibrium is also a Nash equilibrium. Thus, the objective in cooperative games is to find the Paerto-optimal Nash equilibria, also called *optimal equilibria*. Since there can be several optimal equilibria, some coordination mechanisms are necessary.

2.3 Q-Learning

Q-learning [17] is one of the most popular single-agent reinforcement learning algorithms, due to its simplicity and robustness. That is why it was one of the first algorithms to be applied to multi-agent settings.

Let $Q_i(s, a_i)$ be the value of the state-action pair (s, a_i) for agent i, where a_i is the agent's chosen action.[1] The update equation for agent i is:

$$Q_i(s, a_i) \leftarrow (1 - \alpha) Q_i(s, a_i) + \alpha \Big[r + \gamma \max_{a \in A_i} Q_i(s', a) \Big] \tag{3}$$

where s' is the new state, $r = R(s, \langle a_i, a_{-i} \rangle)$ is the global reward received, $\alpha \in [0, 1]$ is the learning rate, and $\gamma \in [0, 1]$ is the discount factor.

3 Hybrid Q-Learning

In this section we present our Hybrid Q-learning approach for solving cooperative stochastic games (shown in Algorithm 1). The algorithm combines between two types of independent learners, that attempt to estimate the quality of an action a, when paired with the actions A' of the other agents [18]:

- *Maximum-based learners* estimate the quality of a based on the maximum reward observed:
$$\text{quality}(a) = \max_{a' \in A'} R_i(a, a')$$
- *Average-based learners* estimate the quality of a based on the average reward:
$$\text{quality}(a) = \frac{\sum_{a' \in A'} R_i(a, a')}{|A'|}$$

[1] We sometimes omit the subscript i, when it is clear that we are referring to a specific agent.

Our agents start as maximum-based learners, by using the maximum reward obtained in each state-action pair for the Q updates. This allows the agents to be initially forgiving towards suboptimal actions by teammates, which are mostly doing exploration in that phase. It also prevents premature convergence of the agents upon a sub-optimal joint policy (see Sect. 4 for a formal analysis).

After the agents have gained enough information about the potential rewards that may be obtained in each state-action pair, they transition into using the average rewards for the Q updates. This allows the agents to learn whether the variation in the observed rewards is caused due to the noise in the environment or to the behaviors of the other agents, thus making them more robust against stochastic rewards and transitions.

Note that the average rewards are being computed from the beginning of the game, thus all the knowledge the agents have accumulated during the exploration phase is being utilized. This is in contrast to other types of learners which initially ignore rewards that lead to negative updates.

A parameter denoted by ρ ($0 \leq \rho \leq 1$) controls the number of "optimistic" steps, during which the maximum reward is used for the Q updates. After ρT number of steps is reached (T is the time horizon of the game), the agents move to using the average rewards for the updates. In our experiments ρ was set to be within the range of [0.1–0.2].

In order to keep track of the reward estimates, each agent maintains four tables:

1. $Q(s, a)$, a table of Q-values per state and action
2. $R_{\max}(s, a)$, a table of maximum rewards per state and action
3. $\overline{R}(s, a)$, a table of average rewards per state and action
4. $N(s, a)$, the number of visits to each state and action

To compute the average rewards in a computationally efficient manner, we make incremental updates to the average rewards received at a state-action pair (s, a) using the following equation:

$$\overline{R}_{n+1}(s, a) = \overline{R}_n(s, a) + \frac{1}{n+1}\left(r_{n+1} - \overline{R}_n(s, a)\right) \tag{4}$$

where r_{n+1} is the reward received after the $(n+1)$th visit to (s, a).

$Q(s, a)$ is updated in Q-learning style to incorporate both the immediate reward and the expected future utility. However, instead of using the current reward signal, which may be unreliable due to noise from the environment and the other agents' actions, we use either the maximum reward $R_{\max}(s, a)$ or the average reward $\overline{R}(s, a)$, depending on the current stage of the learner.

The extension of Algorithm 1 to partially observable environments is pretty straightforward. In the Q and R tables, instead of using the state variable we use the agent's observation history, e.g., $Q(s, a)$ changes into $Q(h, a)$.

Algorithm 1. Hybrid Q-Learning for agent i

Input: Max steps T, ratio of optimistic updates ρ, learning rate α, discount factor γ, exploration ratio ϵ
for $s \in S$ and $a \in A_i$ **do**
  $Q(s,a) \leftarrow 0$
  $R_{\max}(s,a) \leftarrow -\infty$
  $\overline{R}(s,a) \leftarrow 0$
  $N(s,a) \leftarrow 0$
$s \leftarrow$ initial state
for $t \leftarrow 1$ to T **do**
  From s select a using an ϵ-greedy selection rule
  Apply a and observe reward r and next state s'
  $R_{\max}(s,a) \leftarrow \max(R_{\max}(s,a), r)$
  $N(s,a) \leftarrow N(s,a) + 1$
  $\overline{R}(s,a) \leftarrow \overline{R}(s,a) + \frac{1}{N(s,a)}(r - \overline{R}(s,a))$
  if $t < \rho$ **then**
    $R \leftarrow R_{\max}(s,a)$
  else
    $R \leftarrow \overline{R}(s,a)$
  $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[R + \gamma \max_{a' \in A_i} Q(s',a')]$
  $s \leftarrow s'$

4 Pathologies in Multi-Agent RL

In this section we discuss typical pathologies found in multi-agent learning problems, and describe how our approach tackles each one of them.

4.1 Relative Overgeneralization

Relative overgeneralization is a type of equilibrium shadowing, occurring when a sub-optimal Nash equilibrium yields a higher payoff on average, when an agent's selected action is paired with arbitrary actions chosen by the other agents. As a result of the shadowed equilibrium, average-based learners converge upon the sub-optimal Nash equilibrium, that is pareto-dominated by at least one other Nash equilibrium.

For example, in the Climb game (Table 1), the joint action $\langle a, a \rangle$ defines a Pareto-optimal Nash equilibrium. However, if the agents sampled their actions randomly and based their decisions on the average reward, action a would have the worst score (-6.333), action c would have the best score (1.667), and b would be in the middle (-5.667). Thus, average-based learners would tend to converge to the joint action $\langle c, c \rangle$, i.e., the Nash equilibrium is shadowed by the joint action $\langle c, c \rangle$.

On the other hand, maximum-based learners would have evaluated action a as the best action, c as the second best, and b as the worst. Hence, these agents would tend to converge to the proper equilibrium $\langle a, a \rangle$.

In our hybrid approach, the agents start as maximum-based learners. Thus, given enough time for initial exploration, once both agents choose action a, it will become their preferred action. Transitioning into average-based learners will not change their policy, since after both agents' policies have converged to the joint action $\langle a, a \rangle$, the average reward for taking action a will be significantly higher than taking any other action.

Table 1. The Climb game

		Agent 2		
		a	b	c
Agent 1	a	11	−30	0
	b	−30	7	6
	c	0	0	5

4.2 The Stochasticity Problem

In stochastic games, the difficulty for independent learners is to distinguish whether the variation in the observed rewards is due to the noise in the environmnet or to the other agent's behaviors.

For example, in the partially stochastic Climb game (Table 2a) the joint action $\langle b, b \rangle$ yields stochastic rewards of 14 and 0 with 50% probability. An IL must distinguish if distinct rewards received for action b are due to various behaviors of the other agent or to stochastic rewards. In this case, maximum-based learners are drawn towards $\langle b, b \rangle$, despite each agent only receiving a reward of 7 on average. On the other hand, average-based learners learn that the average reward of the joint action $\langle b, b \rangle$ is 7, and thus that the action b is sub-optimal.

Table 2. Stochastic variations of the climb game. The probability of each reward is 50%.

		Agent 2		
		a	b	c
Agent 1	a	11	−30	0
	b	−30	14|0	6
	c	0	0	5

(a) Climb-PS

		Agent 2		
		a	b	c
Agent 1	a	10|12	5|(−65)	8|(−8)
	b	5|(−65)	14|0	12|0
	c	8|(−8)	12|0	10|0

(b) Climb-FS

Using our hybrid learning approach can solve this problem. In the optimistic training phase, each agent will learn the maximum attainable reward for each

action, i.e., 11 for a, 14 for b, and 5 for c. Then, after the transition into using the average rewards, the reward for action b will start to drop off towards 7, thus the agents will switch to the second best action a and stick to it (since its average reward will be the same as the maximum).

Stochastic transitions present an additional hurdle. As was the case for stochastic rewards, stochastic transitions are largely overcome by the agents' gradual shift from being maximum-based to being average-based.

4.3 Miscoordination

Miscoordination occurs when there are multiple Pareto-optimal equilibria, and they are offset in such a way that agents, each selecting what appears to be an optimal action, end up with poor joint actions. For example, in the Panelty game (Table 3), both actions a and c are reasonable for each agent, but if one agent chooses a and the other one chooses c, they receive a low reward.

Table 3. The penalty game

		Agent 2		
		a	b	c
Agent 1	a	10	0	−10
	b	0	2	0
	c	−10	0	10

With our Hybrid-Q approach, if Agent 1 keeps choosing action a and Agent 2 keeps choosing action c, when they turn into average-reward learners, they will both receive an average reward of 5. At that point, when one of the agents tries the other action (as exploration continues with a small probability), both agents will start receiving an average reward that is greater than 5, which will keep them selecting the same action.

4.4 The Alter-Exploration Problem

In multi-agent settings, the exploration of one agent induces noise in received rewards of other agents and can destabilize their learned policies. This alter-exploration problem amplifies the issue of shadowed equilibria, as a high global exploration can result in agents converging upon a sub-optimal joint policy [9].

Our approach handles the alter-exploration issue by adopting a maximum-based learning approach during the exploration phase, i.e., it allows the learning of individual policies while other agents are exploring. By the time the agents have gathered enough data about the rewards, they can change into average-based learners without having the risk of being trapped in a sub-optimal policy due to the other agents' exploring actions.

5 Independent Learner Baselines

We compare our Hybrid Q-learning against four other independent learner algorithms in several cooperative test problems. We briefly discuss these algorithms below.

5.1 Independent Q-Learning

In a MARL context, standard Q-learning is also known as Independent Q-learning (IQL) or Decentralized Q-learning. In IQL, each agent runs an independent Q-learning algorithm with its own Q value estimates, and there is no mechanism for addressing coordination problems. This algorithm has a low robustness face to exploration, since explorative actions by other agents can cause the agent's learned policy to oscillate. Nevertheless, IQL has been successfully applied in some domains [1,6,16].

5.2 Distributed Q-Learning

In Distributed Q-learning [5], the agents optimistically assume that all the other agents will act to maximize their reward. Thus, they update the Q-value of an action only if it is greater than the previous one. This optimistic update avoids sub-optimal policies caused by shadowed equilibria.

To deal with miscoordination, an additional procedure is used to solve the Pareto-optimal equilibria selection problem: the current policy π_i is only updated when the Q-value of the policy's chosen action is no longer the highest such value. The idea is to cause agents to lock onto the earliest-discovered good action (and Nash equilibrium) even when other equivalent equilibria exist.

Distributed Q-learning was proved to find optimal policies in deterministic cooperative Markov games. However, this approach generally does not converge in stochastic environments. In such environments, optimistic learners ignore penalties in the update which are caused by noise in the environment, thus they overestimate the real Q values.

5.3 Hysteretic Q-Learning

Hysteretic Q-learning is an optimistic MARL algorithm introduced to address maximum based learner's vulnerability towards stochasticity [8]. The idea is that agents on one hand should not be blind to penalties, and on the other hand they must be optimistic to minimize the effect of shadowed equilibria.

Hysteretic Q-learning uses two learning rates α and β for positive and negative updates of the Q values, where $\beta < \alpha$. The smaller β learning rate reduces the impact of negative Q-value updates.

However, hysteretic Q-learners still have a tendency to converge to sub-optimal policies when receiving misleading stochastic rewards. [18]

5.4 LMRL2

LMRL2 (Lenient Multiagent Reinforcement Learning 2) [18] is an extension of the lenient learning algorithm, LMRL [12], for stochastic games. LMRL2 is a modified version of Q-learning which maintains per-action temperatures, that are slowly decreased throughout the learning process. An action's temperature affects two things. First, it affects the lenience of the algorithm: high temperatures cause the algorithm to be lenient towards negative policy updates, and so only mix rewards into Q-values if they are better than the current Q-value. With a low temperature, LMRL2 mixes all rewards into the Q value.

Second, the temperature affects the degree of exploration: with high temperatures, action selection is largely random, and with low temperatures, action selection is greedily based on the actions with the highest Q-values. However, when the average temperature drops below a certain minimum temperature, then LMRL2's action selection will suddenly become purely greedy.

5.5 Parameters

All the mentioned techniques have a variety of tunable parameters. For the benchmarks used in our experiments, we have applied the same parameter settings as in [18] (also shown in Table 4). Note that some of these parameters are unique to specific algorithms, e.g., the temperature parameters are relevant only for lenient learning. Furthermore, some parameter values had to be tuned for specific problems in order to obtain the best policy.

Table 4. Default parameter settings

Discount factor	γ	0.9
Learning rate	α	0.1
Exploration ratio	ϵ	0.1
Hysteretic learning rate	β	0.02
Temprature diffusion coefficient	τ	0.1
Temprature decay coefficient	δ	0.995
Minimum temprature	MinTemp	50.0
Maximum temprature	MaxTemp	2.0
Action selection moderation factor	ω	0.1
Leniency moderation factor	θ	1

6 Experiments

We compared our method against all the aforementioned algorithms using four types of test problems with various degrees of stochasticity and partial observability. Each of these problems exhibits some degree of the multi-agent pathologies discussed in Sect. 4.

6.1 Climb Games

First, we apply our algorithm to the three versions of the Climb game presented in Tables 1 and 2. These games have received considerable attention from the community [3,5,9,11,18], as they remain challenging for state-of-the-art algorithms despite their apparent simplicity.

The partially stochastic and fully stochastic versions, here designated Climb-PS and Climb-FS respectively, define the reward for a joint action as one of two different values with certain probabilities. However, the expected reward is always the same as in the original Climb game. Stochasticity greatly complicates the problem: past learners have failed to coordinate in the fully-stochastic Climb game [9,18].

For the baseline algorithms, some of the parameters were tuned according to the values suggested in [18]. For the Hybrid-Q algorithm, we have used the same parameter values for all three versions of the game: $\rho = 0.1$, $\alpha = 0.1$, $\gamma = 0.9$, and $\epsilon = 0.01$. We have found out that higher exploration rates destabilize the learned policy when the agents transition into average-based learners (interestingly, small exploration rates have also been used in [18] for the baseline algorithms).

We have tested each algorithm on 10,000 trials of the game with different random seeds. A trial consists of 10,000 repetitions of the game ($T = 10000$). At the end of each trial, we determine if the greedy joint action is the optimal one. Table 5 shows the percentage of trials converging to the optimal joint action according to the algorithm and the game.

Table 5. Percentage of trials that converged to the optimal joint action in the Climb games (averaged over 10,000 trials). Boldface values indicate the highest-performing methods for a given problem.

	Climb	Climb-PS	Climb-FS
Hybrid Q	**100%**	**99.65%**	**99.45%**
IQL	58.80%	58.93%	50.34%
Distributed Q	**100%**	28.83%	38.11%
Hysteretic Q	**100%**	93.92%	94.98%
LMRL2	**100%**	71.30%	35.11%

We can notice that our Hybrid-Q approach outperforms all baseline algorithms in the stochastic versions of the game, and on par with the best algorithms in the deterministic version.

6.2 Heaven and Hell Game

The Heaven and Hell game [18] is a deterministic game with four states and two actions for each player (Table 6). This game has a high-reward state ("heaven"), a low-reward state ("hell"), and two medium ("purgatory") states of different

levels of reward. Choice of a high-reward action will deceptively transition to a lower-reward future state, and the converse is also true.

Table 6. Heaven and Hell game. The optimal joint action in each state is indicated with boldface.

S	A_1	A_2	Reward	S'
s_1	a	a	20	s_2
s_1	**any**	**other**	10	s_1
s_1	b	b	15	s_3
s_2	a	a	0	s_4
s_2	**any**	**other**	−10	s_1
s_2	b	b	−5	s_2
s_3	a	a	10	s_4
s_3	**any**	**other**	0	s_1
s_3	b	b	5	s_2
s_4	a	a	−10	s_4
s_4	**any**	**other**	−20	s_3
s_4	b	b	−15	s_2

In this game, it is easy for all algorithms to find that the best state for the agents to be in is s_1, thus any joint action that keeps the agents in that state (either $\langle a, b\rangle$ or $\langle b, a\rangle$) is optimal. Therefore, we have also checked which of the algorithms has found the more general *complete policy*, meaning a policy that determines the correct joint action for every state, even ones which would never be visited if it followed a correct policy.

As in the Climb games, each trial consisted of 10,000 repetitions of the game. This time we have conducted 1,000 trials for each algorithm. Table 7 shows the percentage of trials converging to a complete and optimal joint policy according to the algorithm.

Table 7. Percentage of trials that converged to the complete and optimal joint policy in the heaven and hell problem (averaged over 1,000 trials)

Hybrid Q	IQL	Distributed Q	Hysteretic Q	LMRL2
71.0%	17.2%	23.1%	56.8%	63.7%

We can see that the success ratio in finding complete policies is lower than what was achieved in the Climb games, since in this game a complete policy is not needed for maximizing the total accumulated reward.

6.3 Common Interest Game

In contrast to the previous games, the Common Interest game [15] is a stochastic game with two states and stochastic transitions between these states. Table 8 shows the rewards and the transition probabilities for each state and joint action. This game poses a miscoordination challenge, as there are two Pareto-optimal equilibria in the game. In the first, the agents play joint action (a, b) in state 1 and again (a, b) in state 2, while in the second they play (b, a) in state 1 and (a, b) in state 2.

Table 8. Common interest game

S	A_1	A_2	Reward	S
s_1	a	a	0.5	s_1(10%), s_2(90%)
s_1	***a***	***b***	0.6	s_1(10%), s_2(90%)
s_1	***b***	***a***	0.6	s_1(10%), s_2(90%)
s_1	b	b	0.7	s_1 (90%), s_2 (10%)
s_2	a	a	0	s_1(10%), s_2(90%)
s_2	***a***	***b***	1.0	s_1(10%), s_2(90%)
s_2	b	a	0.5	s_1(10%), s_2(90%)
s_2	b	b	0	s_1(90%), s_2(10%)

Similarly to the Climb games, we have executed the Common Interest game for 10,000 time steps, and tested each algorithm on 10,000 instances of the game with different random seeds. At the end of each game, we checked if the final joint policy is one of the two Pareto-optimal equilibria. Table 9 shows the percentage of trials converging to an optimal joint policy according to the algorithm. As can be seen, the Hybrid-Q algorithm has achieved the highest percentage of optimal solutions.

Table 9. Percentage of trials that converged to the optimal joint action in the Common Interest problem (averaged over 10,000 trials)

Hybrid Q	IQL	Distributed Q	Hysteretic Q	LMRL2
99.91%	91.70%	30.58%	59.90%	98.79%

6.4 Meeting in a Grid

In our last experiment, we have tested our algorithm on a partially observable and stochastic environment. We used a variation of the meeting-in-a-grid Dec-POMDP benchmark [2]. The environment consists of a static target and two

agents in a 8×8 toroidal grid world (Fig. 1). The agents get a reward of 1 for simultaneously landing on the target location, otherwise a reward of -0.01 (which encourages the agents to reach the target as quickly as possible). Episodes terminate after 40 transitions or upon successful capturing of the target. Each agent observes its own location and the location of the target, but does not know where the other agent is located.

Each agent can execute one of the five possible actions: move North, South, East, West, or stand still. Actions result in stochastic transitions: each action succeeds with probability $1 - p$, but with probability p, the agent moves at right angles to the intended direction. For example, if $p = 0.2$, and the agent is located at cell (2, 2) and chooses to move North, then with probability 0.8 it will move to cell (2, 1), but with probability 0.1 it will move left to cell (1, 2), and with probability 0.1 it will move right to cell (3, 2).

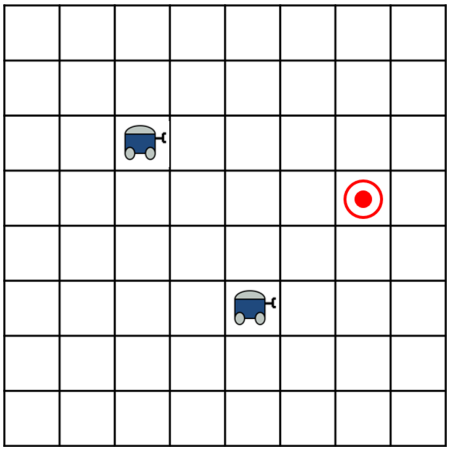

Fig. 1. The meeting-in-a-grid environment.

We ran 50 experiments with different random locations of the agents and the target, each consisting of 1,000 episodes. For each experiment we measure the total return over all the episodes. Therefore, the maximum score that can be obtained in each experiment is 1,000, which occurs when the agents learn to capture the target on the first episode, and keep capturing it in all subsequent episodes.

We have varied the action noise probability p from 0.1 to 0.5 by intervals of 0.05. Figure 2 shows the average return achieved by the various algorithms for the different values of p. As the action noise level increases, the gap between the Hybrid-Q method and the other algorithms widens, which indicates that our approach can better handle higher levels of stochasticity in the environment.

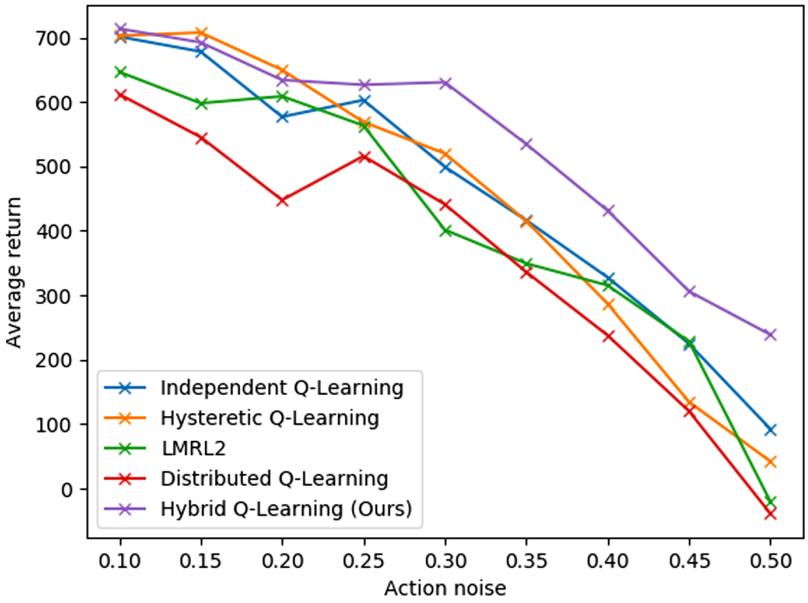

Fig. 2. Average return in meeting-in-a-grid task for various action noise levels.

7 Conclusions

We presented a new hybrid approach for multi-agent reinforcement learning, which is based on agents transitioning from being maximum-based learners into average-based learners. We have shown that our approach consistently converges towards the optimal joint policy in several benchmarks from the multi-agent learning literature.

Despite the simplicity of our approach, it can serve as a basis for a general framework that could be easily extended to other settings as well, e.g., integrating it with deep reinforcement learning, or using smooth transitions from maximum-based learning to average-based learning.

Acknowledgments. This work is funded by the U.S. Air Force Research Laboratory (AFRL), BAA Number: FA8750-18-S-7007, and NSF grant no. 1816382.

References

1. Agogino, A.K., Tumer, K.: A multiagent approach to managing air traffic flow. Auton. Agent. Multi-Agent Syst. **24**(1), 1–25 (2012). https://doi.org/10.1007/s10458-010-9142-5
2. Amato, C., Dibangoye, J.S., Zilberstein, S.: Incremental policy generation for finite-horizon DEC-POMDPs. In: International Conference on Automated Planning and Scheduling (ICAPS) (2009)

3. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. AAAI/IAAI **1998**, 746–752 (1998)
4. Fulda, N., Ventura, D.: Predicting and preventing coordination problems in cooperative q-learning systems. Int. Joint Conf. Artif. Intell. (IJCAI). **2007**, 780–785 (2007)
5. Lauer, M., Riedmiller, M.: An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: International Conference on Machine Learning (ICML) (2000)
6. Leibo, J.Z., Zambaldi, V., Lanctot, M., Marecki, J., Graepel, T.: Multi-agent reinforcement learning in sequential social dilemmas. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 464–473 (2017)
7. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: International Conference on Machine Learning (ICML), pp. 157–163 (1994)
8. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 64–69 (2007)
9. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. Knowl. Eng. Rev. **27**(1), 1–31 (2012)
10. Nash, J.F.: Equilibrium points in n-person games. Proc. Natl. Acad. Sci. U.S.A. **36**(1), 48–49 (1950)
11. Palmer, G., Savani, R., Tuyls, K.: Negative update intervals in deep multi-agent reinforcement learning. In: International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), pp. 43–51 (2019)
12. Panait, L., Sullivan, K., Luke, S.: Lenient learners in cooperative multiagent systems. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 801–803 (2006)
13. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
14. Verbeeck, K., Nowé, A., Parent, J., Tuyls, K.: Exploring selfish reinforcement learning in repeated games with stochastic rewards. Auton. Agent. Multi-Agent Syst. **14**(3), 239–269 (2007)
15. Vrancx, P., Tuyls, K., Westra, R.: Switching dynamics of multi-agent learning. Int. Conf. Auton. Agent. Multiagent Syst. (AAMAS) **1**, 307–313 (2008)
16. Wang, Y., De Silva, C.W.: Multi-robot box-pushing: single-agent q-learning vs. team q-learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3694–3699 (2006)
17. Watkins, C.J., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)
18. Wei, E., Luke, S.: Lenient learning in independent-learner stochastic cooperative games. J. Mach. Learn. Res. **17**(1), 2914–2955 (2016)
19. Yang, E., Gu, D.: Multiagent reinforcement learning for multi-robot systems: A survey. Technical report, Department of Computer Science, University of Essex, Technical report (2004)