Efficient Bayesian Ultra-Q Learning for Multi-Agent Games

Ward Gauderis
Vrije Universiteit Brussel
Brussels, Belgium
ward.gauderis@vub.be

Fabian Denoodt
Vrije Universiteit Brussel
Brussels, Belgium
fabian.luc.m.denoodt@vub.be

Bram Silue
Vrije Universiteit Brussel
Brussels, Belgium
bram.silue@vub.be

Pierre Vanvolsem
Vrije Universiteit Brussel
Brussels, Belgium
pierre.vanvolsem@vub.be

Andries Rosseau
Vrije Universiteit Brussel
Brussels, Belgium
andries.rosseau@vub.be

ABSTRACT

This paper presents *Bayesian Ultra-Q Learning*, a variant of Q-Learning [12] adapted for solving multi-agent games with independent learning agents. Bayesian Ultra-Q Learning is an extension of the Bayesian Hyper-Q Learning algorithm proposed by Tesauro [11] that is more efficient for solving adaptive multi-agent games. While Hyper-Q agents merely update the Q-table corresponding to a single state, Ultra-Q leverages the information that similar states most likely result in similar rewards and therefore updates the Q-values of nearby states as well.

We assess the performance of our Bayesian Ultra-Q Learning algorithm against three variants of Hyper-Q as defined by Tesauro, and against Infinitesimal Gradient Ascent (IGA) [9] and Policy Hill Climbing (PHC) [1] agents. We do so by evaluating the agents in two normal-form games, namely, the zero-sum game of rock-paper-scissors and a cooperative stochastic hill-climbing game. In rock-paper-scissors, games of Bayesian Ultra-Q agents against IGA agents end in draws where, averaged over time, all players play the Nash equilibrium, meaning no player can exploit another. Against PHC, neither Bayesian Ultra-Q nor Hyper-Q agents are able to win on average, which goes against the findings of Tesauro [11].

In the cooperation game, Bayesian Ultra-Q converges in the direction of an optimal joint strategy and vastly outperforms all other algorithms including Hyper-Q, which are unsuccessful in finding a strong equilibrium due to relative overgeneralisation.

KEYWORDS

Reinforcement Learning, Multi-Agent, Game Theory, Q-Learning

1 INTRODUCTION

In multi-agent games, a useful solution for all agents is to play according to the Nash equilibrium [7], but calculating the Nash equilibrium is often impractical or even computationally intractable, especially in imperfect information games [8]. An alternative approach is to use Reinforcement Learning (RL) [10], where the goal for an agent is to learn a policy through trial-and-error learning that maximizes a predefined reward signal. The question of building agents capable of successfully and efficiently adapting their strategic behaviour to other adaptive agents in an imperfect information setting has proven to be a significant challenge in multi-agent research.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.

State-of-the-art algorithms such as Deep Counterfactual Regret Minimization (Deep CFR) [2] and Deep Monte-Carlo (DMC) [13] have shown impressive results in highly complex multi-agent imperfect information games. It is important to note that these modern methods are often designed to tackle large-scale problems. In contrast, Q-Learning is arguably the most well-known and proven RL algorithm with a rich history of successfully being applied to single-agent smaller-scale problems [4]. However, achieving such success in multi-agent games is more challenging. The main challenges for independent Q-learning agents in multi-agent settings are partial observability (e.g., by not being able to directly observe the other agents' policies or learning algorithms) and the non-stationarity of the environment due to changes in the policies of the other agents over time. It could be argued that despite its many strengths, off-the-shelf Q-Learning is not suitable when applied to multi-agent games with mixed-strategy equilibria and non-stationary environments [11].

1.1 Hyper-Q Learning

To address these shortcomings, Tesauro [11] proposes *Hyper-Q learning*, an extension to Q-Learning. The idea behind this algorithm is to learn the value of *joint mixed strategies* (probability distributions over the joint action space) rather than just base actions. The agent, therefore, keeps a Q-table over a set of mixed strategies instead of deterministic actions, including an internal estimate of its opponents' mixed strategies. Formally, we work in the setting of *stochastic games* [6]. In a stochastic game with state space S, the goal of agent i is to find the best mixed strategy $\vec{x}_i = \vec{x}_i(s)$ for all states $s \in S$, given the *expected* mixed strategy $\vec{x}_{-i}(s)$, where $-i$ denotes the average over all agents except i. As in Tesauro's paper [11], we focus on the two-player case, where for the sake of notational simplicity the mixed strategy vector of the focal agent is denoted with x, and the *internal estimation* of the opponent's mixed strategy with y. At each time step t, the agent performs a base action sampled according to its current mixed strategy x and observes a payoff r. Next, a new state s' is observed and the agent adjusts its internal estimation of the opponent's mixed strategy from y to y'. The iterative process behind updating the Hyper-Q function $Q(s, y, x)$ can then be described as follows:

$$\begin{aligned} Q(s,y,x) &\leftarrow Q(s,y,x) + \Delta Q(s,y,x), \\ \Delta Q(s,y,x) &= \alpha(t)\Big[\, r + \gamma \max_{x'} Q(s',y',x') - Q(s,y,x) \,\Big] \end{aligned} \tag{1}$$

where $\alpha(t)$ is the learning rate and γ is the discount factor. The greedy policy $\hat{x}$, is defined as

$$\hat{x}(s, y) = \arg\max_x Q(s, y, x) \tag{2}$$

Regarding the convergence of Hyper-Q Learning, Tesauro argues that for certain types of opponent dynamics, a Finite-Difference Reinforcement Learning implementation will likely be convergent. However, this has not been proven formally. Convergence of Hyper-Q should, similar to Q-Learning, require having visited every state-action pair, which can be achieved through exploring starts or an adequate ϵ-greedy exploration scheme.

Hyper-Q Learning involves estimating the opponent's strategy. Tesauro [11] describes two different *model-free* based techniques to achieve this, each producing their own variant of the Hyper-Q Learning approach. The first way is to use the *Exponential Moving Average* (EMA). More concretely, a moving average of the opponent's strategy is maintained and updated after every action observation according to the following formula:

$$\bar{y}(t+1) = (1-\mu)\bar{y}(t) + \mu\vec{u}_a(t) \tag{3}$$

where $\vec{u}_a(t)$ is the opponent's discrete action a represented in unit vector form, and μ is a hyper-parameter that influences the speed at which the strategy of the agent evolves. The second way to estimate the opponent's strategy is to use *Bayesian strategy estimation*. Using Bayes' rule, we can compute $P(y|H)$, i.e. the probability for opponent strategy y given the history of observed actions H. The formula is stated as follows:

$$P(y|H) = \frac{P(H|yP(y)}{\sum_{y'} P(H|y')P(y')} \tag{4}$$

where $P(y)$ is the prior probability of y, and $P(H|y)$ can be computed as

$$P(H|y) = \prod_{k=0}^{t} y_{a(k)}^{w_k} \tag{5}$$

with w_k the exponent weights which are defined as

$$w_k = 1 - \mu(t-k) \tag{6}$$

The learning algorithms discussed here are in theory applicable to any multi-agent Reinforcement Learning task. In this work, we focus on stochastic normal-form games with two agents which allow for efficient two-dimensional tabular implementations.

1.2 Bayesian Ultra-Q Learning

One particular strength of the Bayesian variant of Tesauro's Hyper-Q Learning agent is the ability to update the Q-values of an entire *row* of the Hyper-Q table, i.e. of all opponent strategies y, in one single iteration. This allows the agent to explore and eventually converge much more efficiently by using the knowledge that mixed opponent strategies that are similar are likely to have similar Q-values.

This line of thought was the inspiration for our *Bayesian Ultra-Q Learning* algorithm, which applies a Bayesian update to the *columns* of the Q-table as well, which correspond to the agent's own strategies x. The agent, therefore, uses the knowledge that its own similar mixed strategies are likely to have similar Hyper-Q values as well. This means that not just one row is updated per iteration, but the entire Q-table. The learning equation for our Bayesian Ultra-Q Learning agent then becomes:

$$\begin{aligned}\Delta Q(y, x) = \; & \alpha \langle x, x_{real} \rangle \; P(y|H) \Big[r \\ & + \gamma \max_{x'} \sum_{y'} P(y'|H') Q(y', x') - Q(y, x) \Big]\end{aligned} \tag{7}$$

Additionally, Bayesian Ultra-Q Learning introduces a weighting factor $\langle x, x_{real} \rangle$ to the learning equation. This is the cosine similarity between the strategy x and the strategy that is actually used, x_{real}. It enables updating similar strategies in the Q-table more strongly than dissimilar strategies.

Alternatively, this weighting factor could be replaced with the probability $P(a|x)$, i.e., the likelihood of playing the action a (that was actually taken) when the mixed strategy x would be used. We noticed that using this factor results in similar agent behaviour but biases the agent toward playing more polarised mixed strategies. This approach is therefore discarded in our results.

It should be noted that the learning equation of Bayesian Ultra-Q no longer tries to solve the original Bellman equation, but a modified version.

1.3 Contributions

The contributions resulting from our work[1] can be summarised as follows. Firstly, we reproduce Tesauro's work behind Hyper-Q Learning applied to the game of rock-paper-scissors, and shed light on certain aspects that require further clarification, such as the definition of y' in the Bayesian update equation. We present Bayesian Ultra-Q Learning, an extension of Tesauro's Bayesian Hyper-Q Learning agent that takes better advantage of the information available to update the entire Q-table at once with each iteration. Bayesian Ultra-Q also introduces a new weighting factor in the learning equation, which determines the intensity with which other similar strategies in the Q-table are updated. Although Bayesian Ultra-Q and Hyper-Q show similar performances in the game of rock-paper-scissors, we argue that this game is not a great fit for benchmarking these agents. The fact that the game's Nash equilibrium is characterized by a completely random strategy, i.e. playing each action with probability $\frac{1}{3}$, means that the agents cannot truly demonstrate their learning capabilities. We chose a more interesting game to benchmark these agents on, namely the Stochastic Hill-Climbing game, which is a cooperation game that requires more complex behaviour from an agent for it to be successful. Our experiments show that Bayesian Ultra-Q strongly outperforms Hyper-Q in this game.

2 EXPERIMENTS

We investigate the convergence and performance of Bayesian Ultra-Q agents by benchmarking them in games against Hyper-Q agents and other, dynamic agents. Two games with an action space of size 3 have been considered: the matrix game of rock-paper-scissors and a stochastic hill-climbing game. The game of rock-paper-scissors helps us investigate the behaviour of these agents in an adversarial zero-sum setting. In contrast, the stochastic hill-climbing game

[1] https://github.com/WardGauderis/Hyper-Q

gives us insights into how these agents behave in a cooperative setting where coordination is difficult.

The dynamic opponent agents in question are agents that use *Infinitesimal Gradient Ascent* [9], and agents that use *Policy Hill-Climbing* [1]. Infinitesimal gradient ascent (IGA) involves carrying out small changes in the agent's strategy in the direction of the gradient of immediate payoff. Strategy updates for IGA are carried out by updating each action probability of the strategy in question as follows:

$$p_{i,j} \leftarrow p_{i,j} + \eta \frac{\delta V}{\delta p_{i,j}} \tag{8}$$

where $p_{i,j}$ is the probability that agent i plays action j, η is the step size parameter, and $\frac{\delta V}{\delta p_{i,j}}$ is the partial derivative of the *expected value*, which is denoted as V. Calculation of this factor requires perfect knowledge of the strategies of all agents involved.

Policy hill-climbing (PHC) is an extension of Q-learning that can be applied to mixed strategies. This learning algorithm keeps track of Q-values and of a policy, each of which is updated at each game iteration. The Q-values are updated according to the following formula:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a')) \tag{9}$$

where α is a learning rate and γ is the discount factor. The policy, meanwhile, is updated according to the following rule:

$$\pi(s,a) \leftarrow \pi(s,a) + \begin{cases} \delta, & \text{if } a = \text{argmax}_{a'} Q(s,a') \\ \frac{-\delta}{|A_i|-1}, & \text{otherwise} \end{cases} \tag{10}$$

where δ is another learning rate and A_i corresponds to the set of actions available to agent i.

For benchmarking purposes, an *Omniscient* variant of the Hyper-Q Learning algorithm is implemented and used for comparison during tests. This agent has perfect knowledge of the opponent's strategy and provides insight into how a Hyper-Q agent ought to behave with perfect information. Finally, we use Monotone agents, which have a static strategy that involves repeatedly performing one fixed action indefinitely.

We use the IGA, PHC, and Monotone agents as opponents in our experiments, as well as the three types of Hyper-Q Learning agents: Omniscient, Exponential Moving Average (EMA) and Bayesian.

2.1 Agent Implementations

Since the mixed strategies of our Bayesian Ultra-Q agents span a continuous spectrum, we employ a uniform grid discretisation of the strategy space containing all pairs (x, y), where x is the agent's strategy and y is the (estimated) opponent's strategy. A mixed strategy is represented by its 3 component probabilities for every action. By discretising the strategy space with a grid of size n, every strategy is represented on a simplex grid of size $N = \frac{n(n+1)}{2}$, giving us a Q-table of a total size N^2 for a two-player normal-form game. We chose a grid size of $n = 25$. The Q-table is implemented as a one-dimensional array and indexing mathematics are used to efficiently map every (x, y) pair of mixed strategies to a single index in the array.

To optimize the performance of each learning algorithm, we conducted a comprehensive grid search of hyper-parameter values. By simulating all algorithms against each other over all possible configurations, we identified and selected the optimal set of hyper-parameters that resulted in the highest average performance for each algorithm, similar to Tesauro [11]. For Hyper-Q agents, the hyper-parameters $\alpha = 0.01$ and $\gamma = 0.9$ were used for all experiments. A value of $\mu = 0.005$ was set for all opponent strategy estimation methods. The policies of the agents are always greedy, except for random resets where a random uniform strategy is selected. In the original paper, these resets occur consistently every 1000 iterations, we replace this with a more general ϵ-greedy exploration method with $\epsilon = 0.001$.

It was noted during initial runs against a Monotone agent that the implementation details of these random resets have a large influence on convergence and performance, and these are unfortunately not discussed in Tesauro's work [11]. For this reason, we performed multiple baseline experiments with different restart configurations. Our first hypothesis was that when both agents randomly restart at exactly the same iteration, the exploration steps could become too large, resulting in rewards that are not indicative of the real Q-value for an estimated strategy. By making the different agents restart at different random iterations, this could be avoided. However, the results showed that, except for the Omniscient Hyper-Q Learning agent, this behaviour does not influence the learning process of the agents in the long run. We, therefore, made the choice to employ the more straightforward simultaneous restart method. The experiments also show that although it increases exploration, it is better for convergence not to restart the strategy estimation methods randomly. To promote exploration, the Q-tables are initialised with randomised Q-values instead.

Since the estimation method used by the Bayesian Hyper-Q Learning agent does not provide a single point estimate y of the opponent's strategy but instead provides a probability distribution $P(y|H)$ over all possible (discretised) mixed strategies, the Hyper-Q Learning equations must also be made Bayesian:

$$\Delta Q(y,x) = \alpha P(y|H)\left[r + \gamma \max_{x'} Q(y',x') - Q(y,x)\right] \tag{11}$$

$$\hat{x} = \arg\max_{x} \sum_{y} P(y|H)Q(y,x) \tag{12}$$

This has the advantage over non-Bayesian methods in the way that not just one Hyper-Q value is updated per iteration, but the values of all possible opponent strategies, weighted by their posterior probability. The definition of the greedy policy (12) is now changed as well because the best action over the *expected* Q-values for all opponent strategies is selected. To improve the tractability of this approach, we estimate the posterior $P(y|H)$ by $P(y'|H')$. For this reason, the Hyper-Q agent with Bayesian estimation has been implemented as a separate agent.

One important aspect that the work of Tesauro [11] does not touch upon, is the definition of y' in its update equation (Equation 11). Since the Bayesian estimation does not provide a single-point estimate of the opponent's strategy, it is unclear how y' is determined in the original paper. One could opt for a mean or median strategy over $P(y|H)$, but we have chosen a more principled approach in the Bayesian spirit, similar to 12, by taking the maximum

expected Q-value over all opponent strategies. Equation 11 becomes:

$$\Delta Q(y,x) = \alpha P(y|H)\Big[r + \gamma \max_{x'} \sum_{y'} P(y'|H')Q(y',x') - Q(y,x) \Big] \quad (13)$$

Regarding the implementation of the opponent agents, the IGA agent starts off with a randomly generated strategy and, as described in equation 8, iteratively updates this strategy using a particular step-size η. Our step-size parameter has been set to $\eta = 0.01$. As for the PHC agent, it starts off with a randomly generated policy and iteratively updates this policy according to equation 10, where the parameters α, δ, and γ have been set to $\alpha = 0.05$, $\delta = 0.01$, and $\gamma = 0.95$. According to Bowling and Veloso [1], the convergence of the PHC algorithm requires a suitable exploration component. To achieve this, our implementation applies a fixed ϵ-greedy exploration with $\epsilon = 0.01$.

2.2 Game Implementation

As previously mentioned, our research considers both the game of rock-paper-scissors and a cooperation game. Since the Hyper-Q agents have the means to estimate their opponent's strategy, we investigate whether the agents can use this knowledge in a cooperative setting. We challenge the agents to compete for a common goal. Agents play together in a stochastic stateless environment and are required to cooperate together to maximize their shared reward.

Kapetanakis et al. [5] propose the Stochastic Hill-Climbing Game as a cooperation game. In this game, two agents are required to collaborate to obtain a shared reward, which depends on their joint action. The agents have to learn to select the optimal joint action while taking into account the probabilities associated with the rewards. The two reward matrices and the probabilities associated with these rewards are described in Table 1. However, this game is susceptible to the issue of *relative overgeneralisation*, which occurs when agents learn to repeat an optimal action less frequently because other agents took the wrong action, leading to a negative team reward. This problem arises because the agents learn relative to their opponents' strategies, rather than learning absolute policies.

The Stochastic Hill-Climbing Game was used to observe how independent agents, which are unaware of their opponent's actions, learn to cooperate. The authors conclude that Independent Q-learners (IQLs) do not converge to an optimal Nash equilibrium and propose a different learning scheme based on reward estimation with a shared action-selection protocol, called the *Commitment agent*. Although the independent learners are not able to converge to an optimal Nash equilibrium, the Commitment learners *do* converge to the optimal Nash equilibrium.

We challenged each opponent estimation strategy of Hyper-Q (i.e. Bayesian, EMA, and Omniscient) against itself. Each agent used the same parameters as in the rock-paper-scissors game described above, except the exploration parameters are different. Instead of ϵ-greedy, we observed slightly better results by applying random restarts every 1000 timesteps. Furthermore, the agents' Q-tables are initialised optimistically to 10 to encourage exploration, rather than the random initialisation in the rock-paper-scissors game. Each pair of agents played for a total of 600,000 steps. We repeated the experiment twenty times.

	b_0	b_1	b_2
a_0	0.4 : -3.5, 0.6 : 4	0.25 : -46, 0.75 : -38	0.6 : -6, 0.4 : -16
a_1	0.25 : -46, 0.75 : -38	0.8 : -5, 0.2 : 5	0.8 : -5, 0.2 : 0
a_2	0.7 : -4, 0.3 : -17	0.6 : -6, 0.4 : -16	0.8 : -6, 0.2 : -1

Table 1: Stochastic Hill-Climbing Game. 'a : x, b : y' means probability 'a' of getting payoff 'x', and probability 'b' of getting payoff 'y'.

3 RESULTS

We present the results of the experiments outlined in Section 2 for the game of rock-paper-scissors first, and the cooperation game second.

3.1 Results for the Rock-Paper-Scissors Game

All performance results are obtained by taking the average result across 20 experiments. We start by comparing each agent's reward against a Monotone (static) agent. The Monotone agent has a fixed strategy, meaning it will always play the same action. Figure 1 displays the average reward for each agent when playing against the Monotone agent. An average reward of 1 corresponds to always winning, while an average reward of -1 corresponds to always losing against the Monotone agent. As expected, all agents perform well and converge to a near 100% win rate (with occasional errors due to forced exploration). We have also tested each agent against a random agent. This random agent is equivalent to a Monotone agent with fixed strategy $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. In this case, each agent converges to an average reward of zero, indicating it has found the Nash equilibrium and cannot exploit the random agent any further.

We now present the performance results of the Hyper-Q agents when they are challenged against the IGA agent. The performance results of our own experiments are provided in Figure 2.

We observe that the average return for all agents hovers between -0.01 and 0.01. This means that all the agents seem to obtain an average reward of approximately 0, and no agent clearly wins against the IGA agent. This is in contrast to the original findings of Tesauro [11], where the three Hyper-Q agents play slightly better than the IGA agent. Regarding our own results, we cannot clearly distinguish a better Hyper-Q agent from our experiments. This could be explained by the fact that our hyperparameters for the IGA agent most likely differ from those used by Tesauro, since hyperparameter values are not disclosed there. A direct look at the policies of our agents reveals that the IGA agent converges to the mixed strategy Nash equilibrium play of $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, making it impossible for the Hyper-Q agents to beat the IGA agent. Hyper-Q agents correctly respond to this with the same mixed strategy to sustain the Nash equilibrium. Finally, our own Bayesian Ultra-Q agent performs on par with the other Hyper-Q estimation strategies, which is to be expected since it cannot beat the Nash equilibrium either.

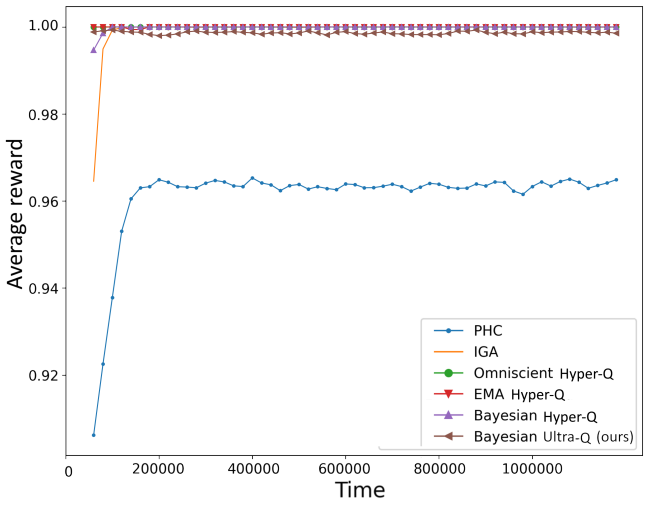

Figure 1: Performance of the different Hyper-Q agents, IGA, and PHC vs Monotone in the game of rock-paper-scissors. The results are smoothed over a rolling window with a window size of 5,000 across all figures.

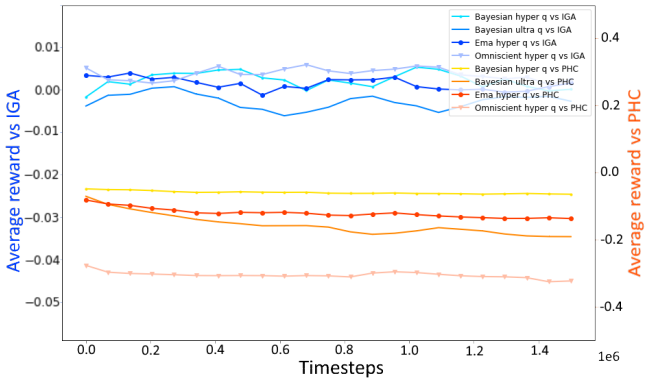

Figure 2: Average reward per time-step for Hyper-Q vs IGA and PHC. The left axis represents the average reward per time-step for Hyper-Q against IGA in the game of rock-paper-scissors. The right axis represents the average reward per time-step for Hyper-Q against PHC. The axes have different scales.

Let us now consider the performance against PHC agents. According to Tesauro [11], the average rewards of the Omniscient and EMA Hyper-Q variants are similar, converging to approximately 0.1. Meanwhile, Tesauro's Bayesian agent made strong improvements, converging to an average reward of 0.17. The author mentions that he cannot explain why the Bayesian agent outperforms the Omniscient agent. This is indeed unexpected since the Omniscient agent has perfect information, giving it a considerable advantage over the Bayesian agent.

Our own reproduction shows that, on average, the Bayesian Hyper-Q agent is indeed the best-performing agent of all the Hyper-Q variants. Despite this, however, the Bayesian Hyper-Q agent converges at an average reward of -0.10 against the PHC agent, meaning that, on average, Hyper-Q agents lose against PHC agents. This finding strongly contradicts the findings of Tesauro [11]. The EMA Hyper-Q agent converges to an approximate average reward of -0.15, and our own Bayesian Ultra-Q agent converges to an average reward of approximately -0.20. Lastly, the Omniscient Hyper-Q agent performs the worst of all Hyper-Q agents, similar to Tesauro's findings. Our reasoning behind this result is that the PHC agent learns to change its strategy drastically at every step. This means that the strategy that the Omniscient agent observes from the PHC agent after one iteration is meaningless because the PHC agent will use a very different strategy in the next iteration. The Omniscient agent is not able to counter this behaviour and therefore performs the worst.

3.2 Results for the Cooperation Game

The reward matrix portrayed by Table1 shows that good cooperation skills are needed to excel since an error can result in severe punishments for both agents. For instance, the optimal Nash equilibrium, corresponding to joint actions (a_0, b_0) corresponds with an expected reward of $0.4 * -3.5 + 0.6 * 4 = 1$. When agent 1 explores action a1 for example, the joint action pair becomes (a1, b0) which has an expected reward of $0.25 * -46 + 0.75 * -38 = -40$, resulting in a large penalty for both agents.

The average reward for each agent is shown in Figure 3. In our experiments, we observed that the two Omniscient agents both converged to the non-optimal joint strategy ((0, 0, 1), (0, 0, 1)), which corresponds to the strategy that is least punishing towards non-cooperation. This is a similar behaviour to what one can expect from the traditional independent Q-learners. Similar to when two independent agents play with each other, the agents have no means of communication, and thus the agents are forced to play a suboptimal, but less punishing strategy. This results in an average reward of $0.8 * -6 + 0.2 * -1 = -5$. This indicates that although each Omniscient agent has perfect knowledge of their opponent, the agents are unable to use this information to coordinate effectively and achieve a better outcome. For reference, two random agents playing the cooperation game result in an expected reward of -13.2.

In figure 3 we observe that the EMA and Bayesian Hyper-Q agents demonstrate considerably worse cooperation than the other Hyper-Q agents, obtaining average rewards around -12. When observing the evolution of the mixed strategies, the agents started from random strategies $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, and evolved to the mixed strategy joint strategy $(\frac{3}{10}, \frac{3}{10}, \frac{4}{10})$ after 600,000 steps. These are small changes after a large number of timesteps. The inaccurate convergence of Hyper-Q agents can be explained by two aspects. First of all, the Exponential Moving Average (EMA) Hyper-Q agent's estimation of its opponent's strategy is based on an average of the opponent's previous 200 mixed strategies, which is a significant delay that can lead to inaccurate estimations. By the time the EMA Hyper-Q agent's estimate is formed, the opponent may have already changed strategy. Secondly, due to the large Q-table corresponding to all possible mixed strategies of the opponent and its own strategies, the convergence simply happens very slowly. These two factors contribute to inaccurate resulting convergence.

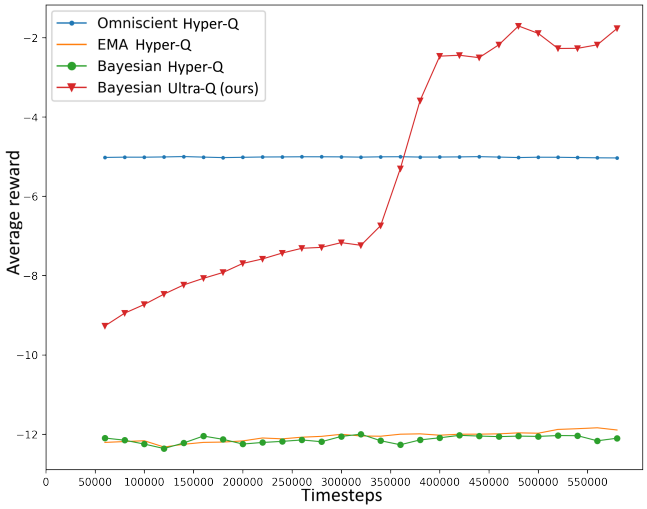

Figure 3: Average reward per timestep for Hyper-Q agents in the cooperation game. Each time an agent plays against its own kind. The rewards are identical for both agents in a single game.

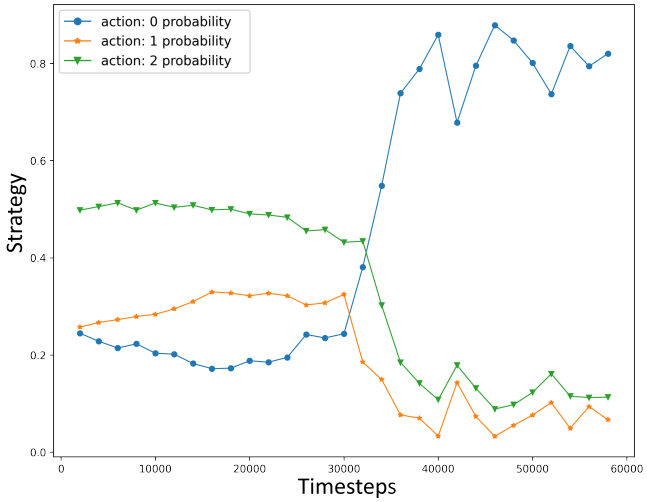

Figure 4: The strategy evolution of one of the two agents in the cooperation game where both agents are Bayesian Ultra-Q agents. Initially, action a_2 has the highest probability, but after a while, the agent recognizes that a_0 is a better option and goes for that action more frequently.

In Figure 3, we observe that the Bayesian Ultra-Q agent outperforms the Hyper-Q variants significantly, getting substantially closer to the optimal expected reward of 1, which corresponds to the optimal joint actions (a_0, b_0). To understand this, recall that any Q-learning agent contains a Q-table where each value represents an estimated expected reward for a particular state. After each environment interaction, the Q-learning agent updates the Q-value corresponding to that state. If a particular state has been explored multiple times and has a low Q-value, a typical Q-learning agent is not likely to exploit the corresponding action even if it suddenly receives a high reward. The Bayesian Ultra-Q agent works differently: if it encounters a state that has previously resulted in low rewards but suddenly receives a high reward, results show that it is more likely to repeat that action in the future. This behaviour is beneficial in difficult cooperation games where relative overgeneralisation is often problematic.

This behaviour is observed at timestep 300,000 in figure 3, where the two Bayesian Ultra-Q agents coincidentally explore a more successful strategy and immediately become more likely to select it again. This behaviour can also be seen in Figures 4, which shows the strategic evolution of a single Bayesian Ultra-Q agent against its equivalent counterpart over 600,000 timesteps. The strategy evolution of the Bayesian Ultra-Q counterpart agent is almost identical, so we only show the strategies for the first agent.

Similar to traditional Q-learning agents, the Bayesian Ultra-Q agents initially converge to joint actions (a_2, b_2). However, after 300,000 iterations, they coincidentally explore joint strategies {(1, 0, 0), (1, 0, 0)} and are rewarded with a large payoff. As a result of this event, the Bayesian Ultra-Q agents become more inclined to choose this particular strategy and other similar strategies, due to their sensitive nature. This preference drives the agents towards convergence to a joint action that returns a higher average reward over time.

4 DISCUSSION

Tesauro [11] proposed Hyper-Q Learning, where agents learn to estimate the value of joint mixed strategies rather than simple deterministic actions. This means that the Hyper-Q table corresponds to a large grid where each joint mixed strategy is evaluated. This approach is beneficial for games with no pure-strategy Nash equilibria, such as the game of rock-paper-scissors. However, the downside of the approach lies in the fact that one must work with a much larger Q-table, resulting in significantly slower convergence.

In this paper, we introduced our Bayesian Ultra-Q agent, which tackles the slow convergence of the Hyper-Q agents by using additional knowledge about the Hyper-Q values of similar strategy pairs to update the entire Hyper-Q table instead of one single entry at every iteration. As a possible measure of similarity, we employ the cosine similarity. We also addressed open questions and irregular results from Tesauro's original work that required more clarification. We provide a clearer definition of the opponent's mixed strategy estimation y' in the Bayesian update equation, revise the results from Tesauro's Hyper-Q agents against PHC and IGA agents, explain the poor performance of Tesauro's Omniscient agent against PHC, and present hyperparameter settings for reproducibility.

We tested all agents in a second setting, distinct from the rock-paper-scissors game in Tesauro's paper, due to the ease of achieving the mixed-strategy equilibrium in that setting. The stochastic hill-climbing game provided a more nuanced and challenging environment for the agents, where clearer distinctions between the different algorithms emerged, with our Ultra-Q learners performing the best.

When Hyper-Q agents play against IGA agents, the IGA agents converge toward the mixed strategy $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ for the **competitive** rock-paper-scissors game. This makes it very hard for Hyper-Q agents to exploit the IGA agents, resulting in an average reward

of nearly zero. When the Hyper-Q agents are challenged against the PHC agent, the Omniscient agent performs the best. This is to be expected since the agent has perfect knowledge of the opponent's mixed strategy. The EMA agent performs the worst, which is caused by the fact that the estimation of the opponent's strategy lags behind the actual opponent's strategy. Indeed, the moving average estimation simply cannot keep up with the continuous changes in the opponent's strategy. The Bayesian Hyper-Q agent performs better than the EMA Hyper-Q agent. Whereas the EMA agent makes a simple estimate of what the opponent's mixed strategy is, the Bayesian agent will instead consider a discrete *range* of different possible mixed strategies, and then attribute a probability distribution across that range of mixed strategies. At every iteration, the Hyper-Q agents observe a reward as a result of their action based on their own mixed strategy. The agents then learn from this observation by updating their Q-tables to better estimate the expected reward for that mixed strategy. However, the Hyper-Q agents only update the Q-table for that particular state (the combination of its mixed strategy and its opponent's mixed strategy), without taking into account that similar strategies will most likely result in similar rewards. Our Bayesian Ultra-Q agent attempts to leverage this feature by also updating the Q-values for all similar joint mixed strategies inside the Q-table, resulting in more efficient convergence to the Nash equilibrium.

The improved performance of the Bayesian Ultra-Q agents in a **cooperative** setting can be explained by two factors. Firstly, Bayesian Ultra-Q displays greater *sensitivity* to changes in the rewards compared to the other Hyper-Q agents. We hypothesize that the ability of the Ultra-Q agent to update the entire Q-table for each interaction with the environment, may explain this phenomenon, but further research is necessary to confirm this hypothesis. In a cooperative game in general, agents will most likely not perform cooperative actions at the same time during exploration, resulting in bad average rewards for those cooperative actions. However, by chance, agents will sometimes perform these cooperative actions together, resulting in a very positive reward. The sensitive nature of Bayesian Ultra-Q Learning helps in picking up on these sudden increases in rewards, drastically increasing the likelihood of trying those cooperative actions again and converging towards the optimal joint strategy. This is in contrast to traditional Q-learners, where these occasional high rewards will not have enough impact on the Q-table, causing the agents to converge to sub-optimal equilibria. Secondly, a Bayesian Ultra-Q agent has the ability to simultaneously update Q-values over all possible opponent strategies and over all strategies of its own. This means that the agent, besides immediately adjusting the Q-values for all possible opponent strategies responsible for the cooperative incident, also adjusts the Q-values for its own strategies that could have led to this cooperative event. This guides both agents *simultaneously* in adjusting their behaviour towards a more promising cooperative strategy.

In conclusion, the Bayesian Ultra-Q learning agent could present an interesting approach to learning mixed strategies in multi-agent games while addressing the issue of relative overgeneralisation. There are multiple possibilities for further research in this direction. To enhance the agent's performance even further, it may be beneficial to analytically derive a more effective weighting factor. By doing so, the agent would be able to accurately solve the original Bellman equation, while still leveraging the similarities between strategies. Additionally, to improve the performance of the Bayesian Ultra-Q agent, it would be interesting to investigate the use of more efficient exploration strategies. A concrete example would be to use an intrinsic reward [3] approach proposed by Chentanez et al.. The intrinsic reward will reward the agent for exploring new states or actions. This will enhance the exploration capabilities of the Bayesian Ultra-Q agent, especially in complex tasks. Finally, exploring the applicability of the Bayesian Ultra-Q agent in other game settings and investigating its scalability to larger games with more complex strategies could also be interesting for future research.

REFERENCES

[1] Michael Bowling and Manuela Veloso. 2001. Rational and Convergent Learning in Stochastic Games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2* (Seattle, WA). Morgan Kaufmann Publishers Inc., San Francisco, CA, 1021–1026.

[2] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. 2019. Deep counterfactual regret minimization. In *International conference on machine learning*. PMLR, 793–802.

[3] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. 2004. Intrinsically Motivated Reinforcement Learning. In *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou (Eds.), Vol. 17. MIT Press, Cambridge, MA. https://proceedings.neurips.cc/paper_files/paper/2004/file/4be5a36cbaca8ab9d2066debfe4e65c1-Paper.pdf

[4] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. 2019. Q-Learning Algorithms: A Comprehensive Classification and Applications. *IEEE Access* 7 (2019), 133653–133667.

[5] Spiros Kapetanakis, Daniel Kudenko, and Malcolm J. A. Strens. 2005. Learning to Coordinate Using Commitment Sequences in Cooperative Multi-agent Systems. In *Adaptive Agents and Multi-Agent Systems II*, Daniel Kudenko, Dimitar Kazakov, and Eduardo Alonso (Eds.). Springer Berlin Heidelberg, Berlin, 106–118.

[6] Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*, William W. Cohen and Haym Hirsh (Eds.). Morgan Kaufmann, San Francisco, CA, 157–163.

[7] Martin J Osborne et al. 2004. *An introduction to game theory*. Vol. 3. Oxford University Press, New York, New York, NY.

[8] Christos H. Papadimitriou. 2007. *The Complexity of Finding Nash Equilibria*. Cambridge University Press, Cambridge, 29–52.

[9] Satinder Singh, Michael Kearns, and Yishay Mansour. 2000. Nash Convergence of Gradient Dynamics in General-Sum Games. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence* (Stanford, CA). Morgan Kaufmann Publishers Inc., San Francisco, CA, 541–548.

[10] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press, Cambridge, MA.

[11] Gerald Tesauro. 2003. Extending Q-Learning to General Adaptive Multi-Agent Systems. In *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Schölkopf (Eds.), Vol. 16. MIT Press, Cambridge, MA, 871–878.

[12] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8 (1992), 279–292.

[13] Daochen Zha, Jingru Xie, Wenye Ma, Sheng Zhang, Xiangru Lian, Xia Hu, and Ji Liu. 2021. Douzero: Mastering doudizhu with self-play deep reinforcement learning. In *International Conference on Machine Learning*. MIT Press, Cambridge, MA, 12333–12344.