Likelihood Quantile Networks for Coordinating Multi-Agent Reinforcement Learning

Xueguang Lyu
Northeastern University
Boston, MA
lu.xue@northeastern.edu

Christopher Amato
Northeastern University
Boston, MA
c.amato@northeastern.edu

ABSTRACT

When multiple agents learn in a decentralized manner, the environment appears non-stationary from the perspective of an individual agent due to the exploration and learning of the other agents. Recently proposed deep multi-agent reinforcement learning methods have tried to mitigate this non-stationarity by attempting to determine which samples are from other agent exploration or suboptimality and take them less into account during learning. Based on the same philosophy, this paper introduces a decentralized quantile estimator, which aims to improve performance by distinguishing non-stationary samples based on the likelihood of returns. In particular, each agent considers the likelihood that other agent exploration and policy changes are occurring, essentially utilizing the agent's own estimations to weigh the learning rate that should be applied towards the given samples. We introduce a formal method of calculating differences of our return distribution representations and methods for utilizing it to guide updates. We also explore the effect of risk-seeking strategies for adjusting learning over time and propose adaptive risk distortion functions which guides risk sensitivity. Our experiments, on traditional benchmarks and new domains, show our methods are more stable, sample efficient and more likely to converge to a joint optimal policy than previous methods.

ACM Reference Format:
Xueguang Lyu and Christopher Amato. 2020. Likelihood Quantile Networks for Coordinating Multi-Agent Reinforcement Learning. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Many multi-agent reinforcement learning (MARL) methods have been developed (e.g., recent deep methods [7–9, 19, 25, 26, 28, 30]), but many of these approaches assume centralized training and decentralized execution. Unfortunately, many realistic multi-agent settings will consist of agents that must (continue to or completely) learn online. In this case, each agent will be an Independent Learner which learns and executes in a decentralized manner using only its own sensor and communication information. This decentralization can be more scalable and is necessary in cases where online (decentralized) learning takes place. Unfortunately, with high probability, the decentralized learning agents will not converge to an optimal joint policy, but only optimal independent policies under the effect of *environment non-stationarity* caused by other agents' optimal independent policies [10]. In other words, without considering other agent exploration, agents will not typically achieve high performance.

Previous work, based on hysteretic Q-Learning [20] and leniency [29], which limit negative value updates (possibly due to exploration) have shown success in Deep reinforcement learning [26, 28] based on Deep Q-Networks (DQN) [22]. Both approaches inject optimism: limiting the decrease of value estimations to alleviate the effect of other agents' exploration strategies and to encourage exploration outside of equilibria which easily trap agents without any injected optimism. The trade-off between environment stochasticity biases and the aforementioned value overestimation, is considered inevitable since domain stochasticity and teammate policy shifts are traditionally indistinguishable. Empirically, leniency shows higher learning stability compared to hysteretic learning, primarily due to a temperature-enabled leniency at different stages of estimation maturity [28]. The leniency decay allows for a more faithful representation of domain dynamics during later stages of training, where it is probable that teammate policies become stable and near-optimal, assuming the rate of decay is appropriate and value maturity is synchronized across all states. Nevertheless, both hysteresis and leniency show only limited performance improvement and leniency introduces hyper-parameters that are hard to tune for new environments. A recent study proposes incorporating negative update intervals [27] to ignore sub-par episodes in a gradually relaxed way. But this method is designed for two-player temporally extended team games and requires an oracle mapping trajectories to domain specific 'meta-actions,' making it inapplicable to the general multi-agent learning setting.

Our work aims to develop a general method for improving decentralized multi-agent reinforcement learning. The method not only automatically identifies transitions involving sub-optimal teammate policies, especially explorations, but also adaptively schedules the amount of optimism applied to each training sample based on estimated value maturity, achieving improved performance without hyper-parameter interventions [26, 28], scheduling tables [28] or specialized experience buffers [27]. In particular, we develop a novel method that extends a state-of-the-art deep distributional single-agent RL method [6], Implicit Quantile Networks (IQN), to multi-agent settings to improve training stability and show how the auxiliary value distribution expectations can be used to identify exploratory teammates through what we call Time Difference Likelihood (TDL). TDL, uses distribution information to identify individual sub-par teammate explorations and guides the amount of optimism injected into the Q distribution; we call the new architecture Likelihood IQN. We show empirically that our method is more robust even in domains that are difficult for previous methods.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

In addition, we propose a Dynamic Risk Distortion operator, in which risk distortion techniques can be applied in a scheduled fashion to produce optimistic policies that are robust to environment non-stationarity.

2 BACKGROUND

We start by providing a summary of the background literature. This section includes introductions to MDPs, decentralized POMDPs and DQN, as well as a brief discussion on the challenges of independent learning in decentralized POMDPs.

2.1 MDPs and Deep Q-Networks

A Markov Decision Process (MDP) is defined with tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ an action space, $\mathcal{T}(s, a, s')$ the probability of transitioning from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$, and $\mathcal{R}(s, a, s')$ is the immediate reward for such a transition. The problem is to find an optimal policy $\pi^\star : \mathcal{S} \to \mathcal{A}$ which maximizes the expected sum of rewards (i.e., values) over time.

Deep Q-Networks [22] consider a common practice where a nonlinear function approximator is used for estimating values by parameterizing the Q function $Q^\theta(s, a)$ with parameters θ using a deep neural network, where $Q(s, a)$ is the expected maximum sum of rewards achievable in the future given state s and action a.

DQN uses experience replay [18] where each transition is stored in a fixed-sized experience buffer

$$D_t = \{(s_1, a_1, r_1, s_2), ..., (s_t, a_t, r_t, s_{t+1})\}$$

from which all training batches for the network are uniformly sampled to balance the network's tendency to bias towards more recent samples. The update of the network follows the following loss function:

$$L_i(\theta_i) = \mathop{\mathbb{E}}_{s,a,r,s' \sim U(D)} [(r + \gamma \max_{a'} Q^{\theta_i^-}(s', a') - Q^{\theta_i}(s, a))^2] \quad (1)$$

where θ_i^- is the parameters for target network, an identical network whose parameters are not updated, but copied from the main network every C steps as to maintain value stability.

2.2 Decentralized POMDPs (Dec-POMDPs)

General cooperative multi-agent reinforcement learning problems can be represented as decentralized partially observable Markov decision processes (Dec-POMDPs) [24]. In a Dec-POMDP, each agent has a set of actions and observations, but there is a joint reward function and agents must choose actions based solely on their local observations. A Dec-POMDP is defined as: $\langle \mathcal{I}, \mathcal{S}, \boldsymbol{A}^{\mathcal{I}}, \mathcal{Z}, \mathcal{T}, \boldsymbol{O}^{\mathcal{I}}, \boldsymbol{R} \rangle$ where $\mathcal{I}$ is a finite set of agents, $\boldsymbol{A}^i$ is the action space for agent $i \in \mathcal{I}$, and $\boldsymbol{O}^i$ is observation space of agent i. At every time step, a joint action $\boldsymbol{a} = \langle a^1, ..., a^{|\mathcal{I}|} \rangle$ is taken, each agent sees it's own observation o^i, and all agents receive joint rewards based on the joint action $\boldsymbol{R}(s, \boldsymbol{a})$.

Earlier work has extended deep RL methods to partially observable MDPs (POMDPs) [14] and Dec-POMDPs. For instance, Deep Recurrent Q-Networks (DRQN) [12] extends DQN to partially observable (single-agent) tasks, where a recurrent layer (LSTM) [13] was used to replace the first post-convolutional fully-connected layer of DQN. Hausknecht and Stone argue that the recurrent layer is able to integrate an arbitrarily long history which can be used to infer the underlying state. DQN and DRQN form the basis of many deep MARL algorithms (e.g., [7, 26, 28]). Our work's basis is IQN (which we discuss later), and the recurrent version that we call IRQN.

Multi-Agent Reinforcement Learning methods are usually classified into two classes: Independent Learners (ILs) and Joint Action Learners (JALs) [5]. ILs observe only local actions a^i for agent i, whereas JALs have access to joint action $\boldsymbol{a}$. Our work is in line with ILs, which may be more difficult, but resembles real-world decentralized learning and may be more scalable.

2.3 Challenges of Independent Learners (ILs)

Even with perfect observability, ILs are non-Markovian due to unpredictable and unobservable teammates' actions, hence the *environment non-stationary problem* [3]. Previous work has highlighted prominent challenges when applying Markovian methods, such as Q-Learning, to ILs: *shadowed equilibria* [10], *stochasticity*, and *alter-exploration* [21].

Shadowed equilibria is the main issue we are addressing, which must be balanced with the *stochasticity* problem. Without communication, independent learners who are maximizing their expected returns are known to be susceptible to sub-optimal Nash equilibria where the suboptimal joint policy can only be improved by changing all agents' policies simultaneously. To battle this issue methods typically put more focus on high reward episodes, with the hope that all agents will be able to pursue the maximum reward possible, forgoing the objective of maximizing the expected return.

Optimistic methods are more robust to *shadowed equilibria*, but give up precise estimation of environment stochasticity. Therefore, these methods can mistake a high reward resulting from stochasticity as a successful cooperation [32]. This challenge is called *stochasticity*. In environments where high reward exists at low probability, the agents will fail to approach a joint optimal policy.

The *alter-exploration* problem arises from unpredictable teammate exploration. In order to estimate state values under stochasticity, ILs have to consider agent exploration. For learners with an ϵ-greedy exploration strategy, the probability of at least 1 out of n agent exploring at an arbitrary time step is $1 - (1 - \epsilon)^n$. The alter-exploration problem amplifies the issue of *shadowed equilibria* [21].

3 RELATED WORK

In a Dec-POMDP, the reward for each agent depends on the joint action chosen by the entire team $\mathcal{I}$; so an agent will likely be punished for an optimal action due to actions from non-optimal teammates. Teammates' policies are not only unobservable and non-stationary, but are often sub-optimal due to exploration strategies. As a result, vanilla Q-Learning would be forced to estimate the exploratory dynamics which is less than ideal. We first discuss related work for adapting independent learners for multi-agent domains, and then discuss Implicit Quantile Networks, which we will extend.

3.1 Hysteretic Q-Learning (HQL)

Hysteretic Q-Learning (HQL) [20] attempts to improve independent learning by injecting overestimation into the value estimation by reducing the learning rate for negative updates. Two learning rates α and β, named the increase rate and the decrease rate, are respectively used for updating overestimated and underestimated TD error δ:

$$Q(x,a) \leftarrow \begin{cases} Q(x,a) + \beta\delta & \text{if } \delta \leq 0 \\ Q(x,a) + \alpha\delta & \text{otherwise} \end{cases} \tag{2}$$

Hysteretic DQN (HDQN) [26] applies hysteresis to DQN, whose TD error is given by

$$\delta_t := Q^{\theta_i}(s_t, a_t) - (r + \gamma f \max_{a'} Q^{\theta_i^-}(s_{t+1}, a')). \tag{3}$$

In practice, HDQN fixes the increase rate α (e.g. $\alpha = 0.001$), and scales the decrease rate as $\beta\alpha$. We thus only discuss the effect of tuning β. In order to reason under partial observability, Hysteretic Deep *Recurrent* Q-Networks (HDRQN) [26], uses a recurrent layer (LSTM) and is trained using replay buffers featuring synchronized agent samples (called CERTs) [26]. Our work utilizes the same buffer structure.

3.2 Lenient Deep Q-Network (LDQN)

Lenient Deep Q-Network (LDQN) [28] incorporates lenient learning [29] with DQN by encoding the high-dimensional state space into lower dimensions where temperature values are feasible to be stored and updated. Leniency, used to determine the probability of negative value updates, is obtained from exponentially decaying temperature values for each *state encoding and action* pair using a decay schedule with a step limit n, the schedule β is given by:

$$\beta_t = e^{\rho \times d^t}$$

for each t, $0 \leq t < n$, where ρ is a decay exponent which is decayed using a decay rate d. The decay schedule aims to prevent the temperature from premature cooling. Given the schedule, the temperature T is folded and updated as follows:

$$T_{t+1}(\phi(s_t), a_t) = \beta_t \Big((1-\upsilon) T_t(\phi(s_t), a_t) + \upsilon \mathop{\mathbb{E}}_{a \in A} T_t(\phi(s_{t+1}), a) \Big)$$

where υ is a fold-in constant. Then, the leniency of a state-action pair is calculated by look up in the temperature table and given by:

$$leniency(s,a) = 1 - e^{-K \times T(\phi(s), a)} \tag{4}$$

where K is a leniency moderation constant.

LDQN schedules optimism injected in state-action estimates, mitigating *shadowed equilibria*, and is able to be robust against *over optimism* as leniency decreases over time. On the other hand, successfully applying LDQN requires careful consideration for decay and moderation parameters, whereas our approach requires fewer hyper-parameters and is robust to different parameter values, yet yields higher performance in terms of improved sample efficiency.

3.3 Implicit Quantile Network (IQN)

IQN [6] is a single-agent Deep RL method which we extend to multi-agent partially observable settings. As a distributional RL method, quantile networks represent a distribution over returns, denoted Z^π for some policy π, where $\mathbb{E}(Z^\pi) = Q^\pi$, by estimating the inverse c.d.f. of Z^π, denoted F_π^{-1}. Implicit Quantile Networks estimate $F_{\pi,\tau}^{-1}(s,a)$ for a given state-action pair, s, a, from samples drawn from some base distribution ranging from 0 to 1: $\tau \sim U([0,1])$, where τ is the quantile value that the network aims to estimate. The estimated expected return can be obtained by averaging over multiple quantile estimates:

$$Q_\omega(s,a) := \mathop{\mathbb{E}}_{\tau \sim U([0,1])} [F_{\pi,\omega(\tau)}^{-1}(s,a)] \tag{5}$$

where $\omega : [0,1] \to [0,1]$ distorts risk sensitivity. Risk neutrality is achieved when $\omega = \mathbb{1}$. In Section 4.3 we will discuss how we distort risk in multi-agent domains and do so in a dynamic fashion where risk approaches neutral as exploration probability approaches 0.

The quantile regression loss [17] for estimating quantile at τ and error δ is defined using Huber loss $\mathcal{H}_\kappa$ with threshold κ

$$\rho_\tau(\delta) = (\tau - \mathbb{1}_{\delta \leq 0}) \frac{\mathcal{H}_\kappa(\delta)}{\kappa} \tag{6}$$

which weighs overestimation by $1-\tau$ and underestimation by τ, $\kappa = 1$ is used for linear loss. Given two sampled $\tau, \tau' \sim \omega(U([0,1]))$ and policy π_ω, the sampled TD error for time step t follows distributional Bellman operator:

$$\delta_t^{\tau,\tau'} = F_\tau^{-1}(s_t, a_t) - (r_t + \gamma F_{\tau'}^{-1}(s_{t+1}, \pi_\beta(s_{t+1}))).$$

Thus, with sampled quantiles $\tau_{1:N}$ and $\tau_{1:N'}$, the loss is given by:

$$L = \frac{1}{N'} \sum_{i=1}^{N} \sum_{j=1}^{N'} \rho_{\tau_i}(\delta^{\tau_i, \tau'_j}) \tag{7}$$

Distributional learning have long been considered a promising approach due to reduced *chattering* [11, 15]. Furthermore, distributional RL methods have shown, in single agent settings, robustness to hyperparameter variation and to have superior sample efficiency and performance [2].

4 OUR APPROACH

We use IQN as the basis of our method since it has shown state-of-the-art performance in single-agent benchmarks, but more importantly, because we believe that learning a distribution over returns provides a richer representation of transitional stochasticity *and* exploratory teammates in MARL. Consequently, the distributional information can be utilized to encourage coordination, but also properly distribute blames among agents, which has historically been difficult to balance. We propose Time Difference Likelihood (TDL) in this section and Dynamic Risk Distortion (DRD), both utilize distributional information to foster cooperation.

Time Difference Likelihood (TDL) is a granular approach for controlling the learning rate in a state-action specific fashion, but without an explicit encoder. Instead, TDL measures the likelihood of a return distribution produced by the target network given the distribution produced by the main network. The motivation is twofold: first, for similar distribution estimations, even with drastic difference in specific quantile location, the learning rate should remain relatively high to capture local differences and improve sample efficiency; second, for teammate explorations, TDL will more likely to be low, hence applying more hysteresis on non-Markovian dynamics. Also, as we show from empirical evaluations, TDL acts as a state-specific scheduler which causes the learning rate to increase over time for states which have received enough training, resulting in more recognition of environment stochasticity, thus converging more robustly towards a joint optimal policy.

Dynamic Risk Distortion (DRD), on the other hand, does not impose value overestimation like hysteresis and leniency; instead, DRD controls the way in which policies are derived from value estimates, by distorting the base distribution from which quantile estimation points τ are sampled. Empirically, DRD is robust to different scheduling hyper-parameters, and allows for faster learning and better performances. Both approaches can be combined to gain better performances.

4.1 Time Difference Likelihood (TDL)

We first discuss TDL, a measure which we propose to later guide the magnitude of the network's learning rate contingent on each update. Motivated to reduce the learning rate when encountering exploratory teammates, but properly updating for local mistakes, we would like to find an indicator value distinguishing the two scenarios. TDL is such an indicator. We scale the learning rate using TDL as discussed later in section 4.2.

At a high level, to calculate the TDL, we first sample from estimated return distributions (using both the main network and the target network) for given observation-action pairs. For simplicity, we denote these as $d_{1:M} := F^{-1}_{\tau_{1:M}}(s_t, a_t)$ and $t_{1:M'} := r_t + \gamma \underset{a'}{max} F^{-1}_{\tau'_{1:M'}}(s_t, a')$, where M and M' are the number of samples drawn from the base distribution. We call them distribution samples and target samples. Observe that obtaining these samples does not add computational complexity, since we can reuse the samples that were used for calculating losses.

Next, we formalize an approximation method for estimating the likelihood of a set of samples, given a distribution constituted by another set of samples. TDL, in particular, is the likelihood of target samples given the distribution constituted by distribution samples. We denote the probability density function given by the distribution samples as $\mathcal{P}(X) := P(X \mid d_{1:M})$. The intuition of calculating TDL is to treat the discrete distribution samples as a continuous p.d.f. on which the proximity intervals of target samples are calculated for their likelihoods. More specifically, if given $\mathcal{P}$, we estimate the likelihood of target samples as follows:

$$l_{t_{1:M'}, d_{1:M}} = \sum_{j \in 1:M'} \mathcal{P}\Big(\frac{t_{j-1} + t_j}{2} \leq X \leq \frac{t_j + t_{j+1}}{2}\Big). \tag{8}$$

Now we only need an approximation of the continuous p.d.f. $\mathcal{P}$ which is represented by discrete samples. Our continuous representation is constructed by assuming the density between neighboring samples d_i and d_{i+1} is linear for generalizability and implementation simplicity. We therefore obtain a set of continuous functions $F_i(X)$ each with domain $(d_i, d_{i+1}]$, where F_i denotes a linearity fits (d_i, τ_i) and (d_{i+1}, τ_{i+1}).

Let $\mathcal{F}(X) = F_i(X)$ iff $X \in (d_i, d_{i+1}]$. In other words, $\mathcal{F}$ is obtained by connecting all the distribution samples into a continuous monotonically increasing probability density function, which consists of $M - 1$ connected linear segments. Using $\mathcal{F}$ as the c.d.f approximation for $\mathcal{P}$, by definition, for arbitrary a and b: $\mathcal{P}(a < X \leq b) = \mathcal{F}(b) - \mathcal{F}(a)$, which can be obtained using the linearity property we defined for $\mathcal{F}$:

$$\mathcal{P}(a < X \leq b) = \sum_{i=1}^{M-1} \frac{|(a, b] \cap (d_i, d_{i+1}]|}{d_{i+1} - d_i} (\tau_{i+1} - \tau_i). \tag{9}$$

Note that intervals $(-\infty, d_1]$ and $(d_i, \infty]$ have no probability density, hence are omitted. TDL can be calculated using an arbitrary number of samples for all $M > 1$ and $M' > 0$.

We can view TDL as not only a noisy consistency measurement between the main and target networks, but also an indicator of information sufficiency in the return distribution estimation. The latter is important for training MADRL agents because it aims to differentiate stochasticity from non-stationary, which allows agent to obtain a more faithful value estimates based on the environment alone, not other peers.

4.2 Likelihood Hysteretic IQN (LH-IQN)

Hysteretic Learning [20] incorporates low returns in a delayed fashion, by updating value estimations at a slower rate when decreasing. Hysteretic approaches show strong performance in both tabular and deep learning evaluations, yet fail to delay value estimations synchronously across the state-action space. Leniency [29] addresses this issue by recording temperature values in the state-action space. Temperature values control the negative update probability, which decrease when an update happens to the corresponding state-action pair. However, when applied in large or continuous state and action spaces, not only is state-action encoding required for computational tractability, but extra care is required for scheduling the temperature [28]; Palmer et al. found it necessary to apply temperature folding techniques to prevent the temperature from prematurely extinguishing.

To combat these issues, we introduce Likelihood Hysteretic IQN (LH-IQN) which incorporates TDL with hysteretic learning. Intuitively, LH-IQN is able to automatically schedule the amount of leniency applied in the state-action space without careful tuning of temperature values thanks to state-action specific TDL measurements. While deep hysteretic learning uses $0 < \beta < \alpha \leq 1$ to scale learning rates, our LH-IQN uses the *max* of β and TDL as the *decrease rate.* More specifically, the learning rate μ_t is given by:

$$\mu_t = \begin{cases} max(\beta, l_{t_{1:M'}, d_{1:M}})\bar{\mu}, & \text{if } \delta_t^{\tau, \tau'} \leq 0 \\ \bar{\mu}, & \text{otherwise} \end{cases}. \tag{10}$$

where $\bar{\mu}$ is a base learning rate suitable for learning assuming a stationary environment (e.g. 0.001), $l_{t_{1:M'}, d_{1:M}}$ is the likelihood defined in Section 4.1 and $\delta_t^{\tau, \tau'}$ is the TD error defined in Section 3.3. To explore the effect of likelihood and hysteresis during evaluation, we also define L-IQN as an IQN architecture which only uses TDL $l_{t_{1:M'}, d_{1:M}}$ as the decrease rate, and H-IQN which only uses β as the decrease rate. Empirically, β ranging from 0.2 to 0.4 yields high performance.

Since TDL generally increases as the network trains toward consistency, the amount of optimism/overestimation added by hysteretic updates is reduced over time, which is analogous to leniency. The key difference is that for domain non-stationarity (caused by stochasticity and/or shifts in teammate policies), which remains unpredictable forever, TDL remains small, effectively employing a low learning rate toward such transitions.

4.3 Dynamic Risk Sensitive IQN

Distributional RL has also been studied for designing risk sensitive algorithms [23]. We introduce dynamic risk sensitive IQN which utilizes what we call *dynamic risk distortion operators*. IQN has shown to be able to easily produce risk-averse and risk-seeking policies by integrating different *risk distortion measures* $\omega : [0,1] \rightarrow [0,1]$ [6, 33]. In single agent positive-sum games, risk-averse policies are sometimes preferred to actively avoid terminal states for more efficient exploration. In MARL, however, agents may benefit from risk-seeking policies as seeking the highest possible utility helps the team break out of sub-optimal shadowed equilibria. As we are not boosting the value estimations directly, we say this approach injects *hope* instead of optimism. In our work, we let IQN learn to reflect the true perceived domain dynamics (no learning rate adjustments), but consider generally higher quantile locations (larger values) when making decisions, producing optimistic policies without raising value estimations. Again, to be robust to environment stochasticity, we anneal the amount of distortion we apply so that in the end we produce policies based on realistic (non-optimistic) value estimations. We discuss two such distortion operators: CVnaR and Wang [31].

CVnaR, Conditional Value-not-at-Risk, is inspired by well studied risk-averse operator Conditional Value-at-Risk ($\text{CVaR}(\eta,\tau) = \eta\tau$) [4]. Our CVnaR is defined as follows:

$$\text{CVnaR}(\eta,\tau) = 1 - \eta\tau. \tag{11}$$

CVnaR maps $\tau \sim U([0,1])$ to $\text{CVnaR}(\eta,\tau) \sim U([\eta,1])$, and as η reduces, CVnaR become less risk-seeking.

Wang [31] is a distortion operator whose range always remains $[0,1]$, but becomes exponentially increasing (probability density shifted towards 1) when given positive bias parameter η. Wang is defined as:

$$\text{Wang}(\eta,\tau) = \Phi(\Phi^{-1}(\tau) + \eta) \tag{12}$$

where Φ is the standard Normal cumulative distribution function. Observe that when $\eta \longrightarrow 1$, Wang almost always returns 1, becoming the most risk-seeking distortion operator possible. Also, like CVnaR, as $\eta \longrightarrow 0$, risk-neutrality is observed.

We found it suitable to linearly anneal η (for both Wang and CVnaR) during training to achieve better stability as the agent becomes more and more risk-neutral, but behaves like a maximization approach in the beginning. The aim is that during the initial risk-seeking period when η is high, agents are encouraged to explore highly rewarding spaces, which supports them to better break out of shadowed equilibra; whereas in the end, the risk-neutral distortion produces an unbiased policy which is unlikely to fall for domain stochasticity. We show that, empirically, risk distortion not only improves overall performance when applied alone, but also when used in conjunction with TDL.

5 EVALUATION

In this section, we compare our methods with previous state-of-the-art methods in multiple domains. We begin by comparing likelihood hysteretic IQN (LH-IQN) with the previous state-of-the-art, HDRQN and LDQN, and then analyze the effect of TDL as well as risk distortion operators. Results shown in all Figures use decentralized training with 20 random seeds.

5.1 Evaluation on meeting-in-a-grid

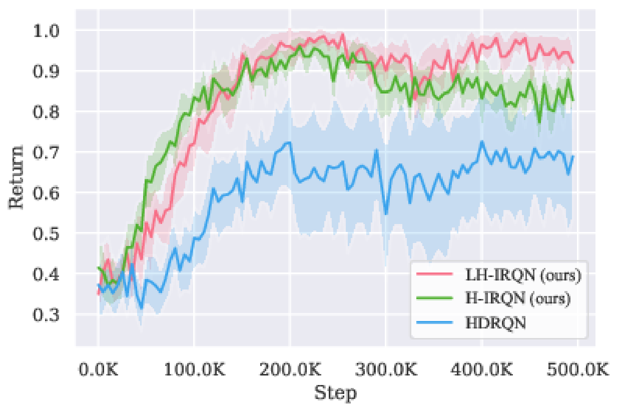
We first conduct experiments on a standard partially observable meeting-in-a-grid domain [1] to be consistent with previous work [26]. The meeting-in-a-grid task consists of one moving target and two agents in a grid world. Agents get reward of 1 for simultaneously landing on the target location and 0 otherwise. Episodes terminate after 40 transitions or upon successful meeting at the target. Agents have noisy transition probability of 0.1, where agents' moves end up in an unintended position (uniformly left, right and still) at the rate of 0.1. Observations include flickering locations of the agents themselves and the target.

Again, to be consistent with previous work [26], we use recurrent versions of our methods. We label the architecture with added Recurrency as LH-I**R**QN. The network starts with 2 fully connected layers of 32 and 64 neurons respectively, then has an LSTM layer with 64 memory cells and a fully connected layer with 32 neurons which then maps onto value estimates for each action. We use $\beta = 0.4, \gamma = 0.95$ and Adam [16] for training. For quantile estimators, we sample 16 for τ and τ' to approximate return distributions, and τ embeddings are combined with the LSTM output.

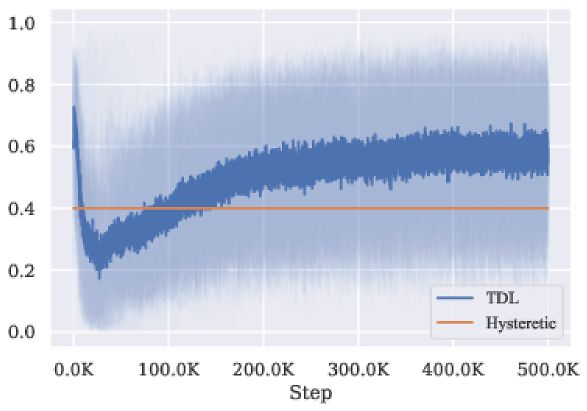
We first evaluate LH-IRQN's performance against HDRQN [26] and H-IRQN on a 4×4 grid (Fig. 1(a)). H-IRQN is a version of LH-IRQN that does not use TDL, but uses IRQN with hysteresis (Eq 10). Both of our IRQN methods outperform HDRQN in both learning rate and final performance. HDRQN has a large variance because it does not robustly solve the task—only a portion of seeds reached near-optimal policies. Our IRQN-based methods show more stability concerning reaching optimality, but not utilizing TDL makes agents susceptible to environment stochasticity, producing fewer near-optimal joint policies over time. Our methods similarly outperform HDRQN in higher dimensional (5×5, 6×6) variations of the benchmark , except for 3×3 which is too simple to differentiate the methods.

Directly applying LDQN, with convolution layers replaced by fully-connected layers to better suit the observations, on meeting-in-a-grid failed to solve the tasks due to the high flickering probability and the observation encoding. Additional comparisons with LDQN are given in section 5.3 and 5.2, but this shows the sensitivity of LDQN to the task and encoding.

As shown in Fig. 1(b), TDL increases over time during training, while maintaining a high variance which resulted from domain non-stationary as expected. Overall, the usage of TDL versus hysteresis β increases significantly; as TDL is used when it is larger than β, which can be considered a lower cap. The overall learning rate for negative samples (i.e., those with non-positive TD error) is increased over time, thus adding less hysteresis and optimism to experiences deemed predictable by TDL, eventually theoretically learning unbiased state value estimations [20]. While one would expect methods with less optimism to be susceptible to action shadowing, our Likelihood method nonetheless achieves better stability and performance as shown in Fig. 1(a), from which we can deduce that TDL is able to distinguish domain non-stationary from stochasticity as we theorized. The spike (and dip) at the beginning seen in Fig 1(b) is due to immature quantile estimations being used to calculate TDL; during the start of training, these quantile values are not guaranteed to represent a valid distribution—they may be



(a) Performance on 4 × 4 meeting-in-a-grid benchmark



(b) TDL Value

Figure 1: (a) IRQN models perform better than HDRQN, especially with TDL (b) TDL values during training of LH-IRQN, shows clear increase of usage of TDL over hysteresis.

aggregated together or even reversed depending on the network weight initializations. As a result, it is unstable to solely use TDL as a decrease rate, a problem which we solved with maximizing with hysteresis parameter β, which is essentially a lower bound to prevent the network from terminating the learning process when estimations differ drastically from target estimates. This issue can also also be mitigated using Dynamic Risk Distortion (DRD) which can be used to achieve extremely optimistic distortion during the beginning phase of training. We discuss empirical improvements of DRD in Section 4.3.

5.2 Multi-Agent Object Transportation Problems (CMOTPs)

We also evaluate LH-IQN on variations of Coordinated Multi-Agent Object Transportation Problems (CMOTPs) [28], consistent with Palmer et al.'s work on LDQN. CMOTPs require two agents carrying a box to a desired location for a terminal reward; the box moves when agents are adjacent and move in the same direction. Variations of the task include obstacles and stochastic rewards. CMOTPs have 16 × 16 observations with added noise.

Our network architecture mimics that of LDQN for comparability: two convolutional layers with 32 and 64 kernels, a fully connected layers with size 1024 which combines quantile embedding, followed by another fully connected layer with size 1024, which then maps onto value estimates for each action. Hyper-parameters remain the same as original work which were found suitable for training in CMOTPs.

As seen in Fig. 2, although both methods converge to a policy that solves the problem consistently, our method shows an improved sample efficiency. We hypothesize that the temperature is decaying less aggressively than it should be in LDQN, which is likely due to temperature folding techniques and/or that the hashing space of the autoencoder is larger than the theoretical minimum.

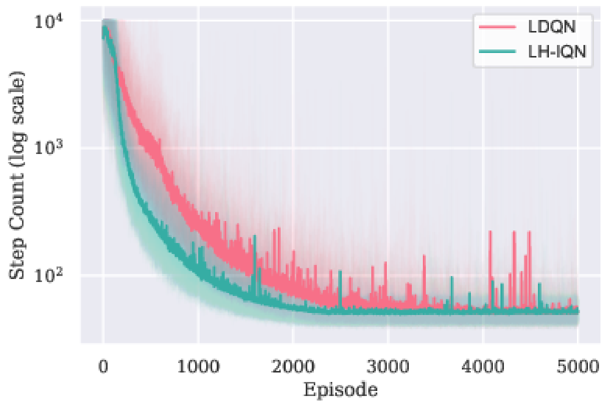
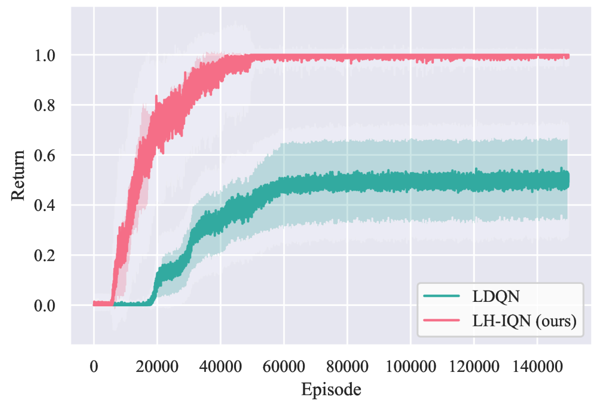


Figure 2: CMOTP benchmark results with a total of 60 runs, aggregated over all three CMOTP variants.

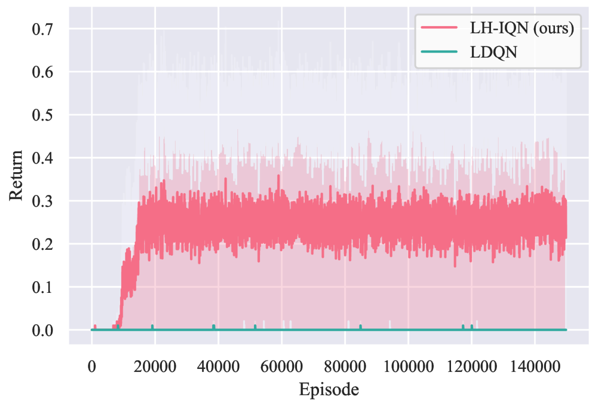
On the other hand, our method utilizes TDL to scale negative updates and shows better sample efficiency. Initially the value estimations do not seem optimistic enough to perform coordinated actions or to propagate to an earlier-stage state, but the likelihood estimation has the added benefit of being able to produce small values in under-explored state-action space, while hesitating less to update negatively in explored spaces. TDL also helps to synchronize optimism across state-action space; in other words, the ability to estimate a distribution consistency adds less optimism to state-action pairs which have received enough training to be able to produce consistent distributions.

5.3 High dimensional meeting-in-a-grid task

Motivated to most fairly compare the performance of our approach with LDQN and to compare on more than two agents, we modify



(a) 4-agent high-dimensional meeting-in-a-grid benchmark



(b) 5-agent high-dimensional meeting-in-a-grid benchmark

Figure 3: High dimensional meeting-in-a-grid 4×4 benchmark with more agents.

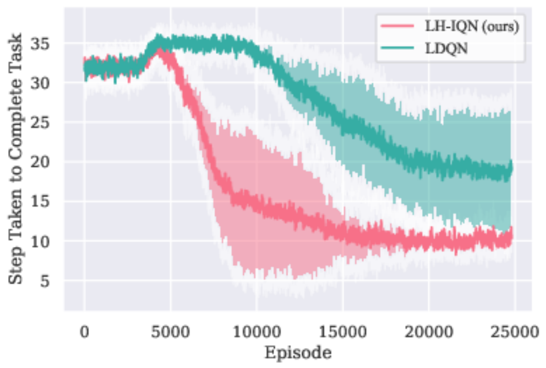


Figure 4: Evaluation of LH-IQN compared with LDQN on the high-dimensional meeting-in-a-grid 4×4 benchmark. Figure shows the number of steps taken to complete tasks (small values preferred).

the existing 4×4 meeting-in-a-grid benchmark to produce graphical observations (16 × 16) with added noise, a type of task on which LDQN is originally evaluated. Due to the difficulty of grid searching numerous hyper-parameters for LDQN, the parameters used were linearly searched individually while fixing others based on the original work. We found reducing temperature schedule decay rate d from 0.9 to 0.8 helps with convergence in our task, possibly due to meeting-in-a-grid's shorter scenarios. We used: $K = 3.0, d = 0.8, \xi = 0.25$ and $\mu = 0.9995$ along with the autoencoder, where ξ is the exponent for temperature-based exploration, and μ is the decrease rate for maximum temperature.

As seen in Fig. 4, our method shows higher sample efficiency and performance. Noticing the y-axis is the number of steps needed to complete the task. We see that LDQN was able to solve the task, however it is not as stable and has less ideal performance compared to LH-IQN. As the task becomes reliably solvable, LDQN slows down learning and has a high variance, whereas LH-IQN achieves the optimal solution on every run. We notice that the temperature values of LDQN are low during the final stages of training, suggesting minimal leniency is applied. Therefore, it appears the joint policy reaches a shadowed equilibrum with less effective explorations.

Benchmarking with more than two agents. Since the probability of effective explorations decreases exponentially as the number of agents increases due to *alter-exploration* [21], independent learners often suffer from poor scalability in the number of agents. LH-IQN mitigates this issue by putting more emphasis on non-exploratory episodic samples which we show in 5.3 and Fig. 5. We evaluate our method against LDQN in those scenarios involving more agents; and as shown in Fig. 3(a), LDQN failed to solve the 4-agent environment reliably, persisting at a 50% fail rate, whereas our method's fail rate converges to 0 (return of 1). Moreover, in the 5-agent environment, shown in Fig. 3(b) LDQN failed to learn any effective policies even though it does encounter cooperatively successful episodes. On the contrary, while noisy, our method is able to successfully learn in this large domain.

Explorations and TDL values. We inspect the TDL values during training the 4 agent high-dimensional meeting-in-a-grid task and plot training TDL values in Fig. 3(a) depending on whether the agent was exploring or not. The TDL trends are shown in Fig. 5. *Self-exploring TDL* refers to TDL values produced from training batch transitions in which the agent is actually exploring (uniformly action selection as opposed to greedily maximizing expected return), and *non-self-exploring TDL* refers to that of when teammates were exploring but the agent is not. We observe that self-exploring TDL values are much higher than that of non-self-exploring during the active policy improvement period. This divergence of TDL values when the agent is versus is not exploring suggests that TDL is able to distinguish local mistakes (where it applies higher learning rate)

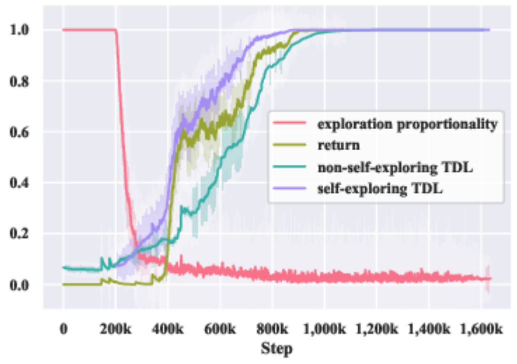


Figure 5: TDL values in different exploration situations, where self-exploring TDL refers to TDLs produced from training sample transitions in which the agent is exploring, and non-self-exploring TDL refer to teammates' explorations. Exploration proportionality shows how much agents are actively exploring.

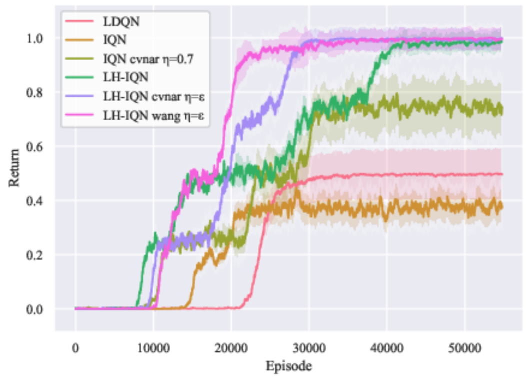


Figure 6: Performance of Risk Distortion methods applied to IQN with and without Likelihood-Hysteresis on the 4×4 high-dimensional meeting-in-a-grid benchmark with high environment stochasticity (sliding probability of 0.3 for moving); shows that distortion operator works well with and without TDL.

from teammate explorations as we theorized. Note that the agent does not have access to ground-truth exploration information in any stage of training, therefore TDL is inferring exploratory information only from the given transitions.

5.4 Dynamic Risk Sensitive IQN

We also demonstrate usage of dynamic risk distortion (DRD) operators to produce optimistic policies in an environment with even more environment stochasticity, highlighting that DRD can be used in conjunction with TDL for policy stability. As shown in Fig. 6, both operators, CVnaR and Wang with $\eta = \epsilon$, when applied on top of our IH-IQN approach, reach optimality more efficiently. Notice that the environment is more challenging as the sliding probability (probability of ending up in an unintended adjacent position upon moving) increased from the previously used 0.1 to 0.3. TDL is unstable initially depending the weight initializations, often taking on extreme values such as 0 or 1 in practice. Therefore, we reason that the TDL-based agents are likely to fall for environment stochasticity at the beginning of training like previous methods because value estimations across states are not in the same learning stage initially. However, risk distortion solves the issue that TDL can take on extremely low TDL value in early stages of learning; since in the early stage, high η shifts the density of the distribution from τ towards 1, making the policy resemble a maximization-based approach, yet still learning in an unbiased manner in terms of value estimations. We also found that LH-DQN is robust to different η values when using both Wang and CVnaR. We simply used the exploration parameter ϵ as the value for η for our dynamic distortion operator in our evaluation. Instead of using ϵ, a separate scheduling can be adapted for η, and we found it to be more appropriate to linearly anneal η from 0.9 to 0.4, but we found that performance differences are small in the benchmarking environment.

Also shown in Fig. 6 is that vanilla IQN yields limited performance, but exceeds LDQN when simply applying a fixed risk distortion (CVnaR) without annealing. Overall, we observe that DRD usually leads to faster policy improvements due to initially high η making it overly optimistic, breaking shadowed equilibria and reducing the initial inconsistencies in quantile estimates; on the other hand, if applied without annealing, the agents' policies are subjected to environment stochasticity, since the derived policies are not maximizing expected return. Therefore, it would be advisable to reduce (either annealing η or reduce the likelihood of distortion) or turn off risk distortion in later stages of training.

6 CONCLUSION

This paper describes a novel distributional RL method for improving performance in cooperative multi-agent reinforcement learning settings. In particular, we propose a likelihood measurement applicable in distributional RL, TDL, that is used for comparing return distributions in order to adaptively update an agent's value estimates. Through inspecting TDL values and usages, we conclude that TDL plays a part in distinguishing domain non-stationary (e.g., from other agent learning and exploration) and domain stochasticity (including teammate policy shifts), a long standing difficulty. We compare and analyze our method along side state-of-the-art methods on various benchmarks, and our approach demonstrates improved stability, performance and sample efficiency. Furthermore, we demonstrate the effectiveness and adaptiveness of our method when incorporating dynamic risk distortion operators, and show risk distortion can also be applied to foster cooperation even without incorporating TDL.

7 ACKNOWLEDGEMENTS

We gratefully acknowledge funding from ONR grant N00014-17-1-2072 and NSF grant #1816382.

REFERENCES

[1] Christopher Amato, Jilles Steeve Dibangoye, and Shlomo Zilberstein. 2009. Incremental Policy Generation for Finite-Horizon DEC-POMDPs. In *ICAPS*.

[2] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. 2018. Distributed Distributional Deterministic Policy Gradients. *arXiv preprint arXiv:1804.08617* (2018).

[3] Michael Bowling and Manuela Veloso. 2002. Multiagent learning using a variable learning rate. *Artificial Intelligence* 136, 2 (2002), 215–250.

[4] Yinlam Chow and Mohammad Ghavamzadeh. 2014. Algorithms for CVaR optimization in MDPs. In *Advances in neural information processing systems*. 3509–3517.

[5] Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* 1998 (1998), 746–752.

[6] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. 2018. Implicit Quantile Networks for Distributional Reinforcement Learning. *arXiv preprint arXiv:1806.06923* (2018).

[7] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning.

[8] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Philip Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

[9] Jakob N 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*.

[10] Nancy Fulda and Dan Ventura. 2007. Predicting and Preventing Coordination Problems in Cooperative Q-learning Systems.. In *IJCAI*, Vol. 2007. 780–785.

[11] Geoffrey J Gordon. 1995. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*. Elsevier, 261–268.

[12] Matthew Hausknecht and Peter Stone. 2015. Deep recurrent Q-learning for partially observable MDPs. *CoRR, abs/1507.06527* 7, 1 (2015).

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[14] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence* 101 (1998), 1–45.

[15] Sham Kakade and John Langford. 2002. Approximately optimal approximate reinforcement learning. In *ICML*, Vol. 2. 267–274.

[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] Roger Koenker and Kevin Hallock. 2001. Quantile regression: An introduction. *Journal of Economic Perspectives* 15, 4 (2001), 43–56.

[18] Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Technical Report. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

[19] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments.

[20] Laëtitia Matignon, Guillaume Laurent, and Nadine Le Fort-Piat. 2007. 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'07*. 64–69.

[21] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. 2012. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review* 27, 1 (2012), 1–31.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[23] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. 2010. Nonparametric return distribution approximation for reinforcement learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 799–806.

[24] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer.

[25] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan How. 2019. Learning to Teach in Cooperative Multiagent Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[26] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the International Conference on Machine Learning*.

[27] Gregory Palmer, Rahul Savani, and Karl Tuyls. 2018. Negative Update Intervals in Deep Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1809.05096* (2018).

[28] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. 2018. Lenient multi-agent deep reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 443–451.

[29] Liviu Panait, Keith Sullivan, and Sean Luke. 2006. Lenient learners in cooperative multiagent systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, 801–803.

[30] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*. 4295–4304.

[31] Shaun S Wang. 2000. A class of distortion operators for pricing financial and insurance risks. *Journal of risk and insurance* (2000), 15–36.

[32] Ermo Wei and Sean Luke. 2016. Lenient learning in independent-learner stochastic cooperative games. *The Journal of Machine Learning Research* 17, 1 (2016), 2914–2955.

[33] Menahem E Yaari. 1987. The dual theory of choice under risk. *Econometrica: Journal of the Econometric Society* (1987), 95–115.