A Game-Theoretic Perspective on Risk-Sensitive Reinforcement Learning

Mathieu Godbout [1, 2, 3], **Maxime Heuillet** [1, 2], **Sharath Chandra Raparthy** [3], **Rupali Bhati** [1, 2, 3], **Audrey Durand** [1, 2, 3, *],

[1] Université Laval, [2] Institut Intelligence et Données, [3] Mila, [*] CIFAR AI Chair
{mathieu.godbout.3@, maxime.heuillet.1@, rupali.bhati.1@, audrey.durand@ift}.ulaval.ca, raparths@mila.quebec

Abstract

Most Reinforcement Learning (RL) approaches usually aim to find a policy that maximizes its expected return. However, this objective may be inappropriate in many safety-critical domains such as healthcare or autonomous driving, where it is often preferable to optimize for a risk-sensitive measure of the policy's return as the learning objective, such as the Conditional-Value-at-Risk (CVaR). Although previous literature exists to address the problem of learning CVaR-optimal policies in Markov decision problems, it mostly relies on the distributional RL perspective. In this paper, we solve this problem by rather proposing an approach based on a game theoretic perspective, which can be applied on top of any existing RL algorithm. At the core of our approach is a two-player zero-sum game between a policy player and an adversary that perturbs the policy player's state transitions given a finite budget. We show that, the closer the players are to the game's equilibrium point, the closer the learned policy is to the CVaR-optimal one with a risk tolerance explicitly related to the adversary's budget. We provide a gradient-based training procedure to solve the proposed game by formulating it as a Stackelberg game, enabling the use of deep RL architectures and training algorithms. We illustrate the applicability of our approach on a risky artificial environment, presenting the different policies learned for various adversary budgets.

1 Introduction

Reinforcement Learning (RL) (Sutton, Barto et al. 1998) is a branch of machine learning where the learner (agent) is not told how to behave in an environment, but instead must learn to do so by trial and error. RL approaches usually aim to find agents that maximize their expected return. This expectation maximization objective has led to increased successes in domains like videogames (Vinyals et al. 2019), board games (Silver et al. 2018) or content recommendation (Li et al. 2010). However, in safety-critical domains like healthcare, autonomous driving or financial planning, some erroneous actions may lead to disastrous consequences. In healthcare for instance, an RL agent may be in charge of designing the shortest path to an organ for surgery. In this task, some paths may be shorter but at the same time may be risk endangering the patient as they are too close to an artery, a nerve or a critical region of the brain (Baek et al. 2018). Automation in this context and other safety-critical domains can only come if the agent is able to successfully reach its goal while avoiding the most risky actions (Gottesman et al. 2019). Therefore, naively maximizing the expected return is not a satisfying approach in such scenarios.

This has motivated the community to design risk-sensitive algorithms, where the agent is trained to account for the possibility of catastrophic events. One way to proceed is to include a risk measure (Artzner et al. 1999) in the algorithm's objective. A commonly used risk measure in RL is the Conditional Value-at-Risk (CVaR_α), defined as the expectation over the worst α-quantile of a distribution. The lower the value of α, the more risk-averse the agent. The search for CVaR-optimal policies, usually referred to as CVaR RL, is typically achieved with distributional RL (Schubert et al. 2021; Keramati et al. 2020). In this approach, the agent predicts the whole distribution of returns rather than only its mean (Bellemare, Dabney, and Munos 2017; Dabney et al. 2018). A CVaR_α transform is then applied to the predicted distribution, resulting in more conservative choices depending on the chosen α threshold.

In this work, we rather tackle the CVaR RL problem in a game-theoretic setting. Looking at a problem from a game-theoretic perspective has helped advance other fields like Generative Adversarial Networks (GANs) (Berthelot, Schumm, and Metz 2017; Goodfellow et al. 2020), suggesting that this perspective is a promising new research avenue for risk-sensitivity in RL. Precisely, we suggest that learning the policy of the RL protagonist (agent) can be cast as a two-player zero-sum game between this protagonist and an antagonist. In this game, the protagonist aims at maximizing its reward collection, while the goal of the antagonist is to minimize the rewards obtained by the protagonist. In order to achieve this, the antagonist has access to a fixed budget for interfering with the next state transitions of the protagonist. Our contributions are the following:

- we propose a two-player zero-sum game formulation which yields (approximate) CVaR-optimal policies at its (approximate) equilibrium;
- we provide a gradient-based algorithm to solve the proposed game alongside sufficient conditions to ensure its convergence;

- we illustrate the applicability of our method in a risky artificial experiment.

The rest of the paper is separated as follows. Section 2 first introduces the necessary background and notation. Following is an overview of the related work in Section 3. Next, we present in Section 4 an overview of our proposed game formulation alongside its relevant properties, which include the convergence to a CVaR RL policy at equilibrium point. In Section 5, we develop a gradient-based algorithm based on the Stackelberg game formulation to solve the presented game, presenting both a theoretical analysis of its convergence as well as key practical concerns. Lastly, we illustrate the validity of the method in an artificial gridworld experiment (Section 6).

2 Background and Notation

Reinforcement Learning

In RL, an agent interacts with an environment, trying to identify actions that lead to the highest rewards. The most common RL formulation for the environment is that of a Markov Decision Process (MDP) framework. An MDP $\mathcal{M}$ is represented by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to [0,1]^{\mathcal{S}}$ is the transition probability function which governs the evolution of the system, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0,1)$ is a discount factor and ρ is the initial state distribution. We formalize an agent's decisions as following a policy $\pi_\theta : \mathcal{S} \to [0,1]^{\mathcal{A}}$ that generates distributions over which actions to take for each state from parameters θ[1].

A policy's actions not only generate immediate rewards but also influence future rewards because they rule the next state the agent will end up in. This is why the performance of a policy π is measured by balancing immediate and distant rewards, using a measure called the random return:

$$\mathcal{J}(\pi) := \sum_{t=0}^{\infty} \gamma^t r_t, \tag{1}$$

where γ is a discount factor, $a_t \sim \pi(s_t)$, $s_{t+1} \sim \mathcal{P}(s_t, a_t)$ and $r_t \sim \mathcal{R}(s_t, a_t)$. Due to the stochasticity that may be present in either the reward function, the policy's action selection or the environment transitions, the random return $\mathcal{J}(\pi)$ is accurately defined as a random variable. The usual objective in RL is to find the policy π^* that maximizes its expected return, which boils down to finding $\pi^* = \arg\max_\pi \mathbb{E}[\mathcal{J}(\pi)]$.

Conditional-Value-at-Risk

Let Z be a random variable with a bounded expectation $\mathbb{E}[Z] < \infty$ and cumulative distribution function $F(z) := \mathbb{P}(Z \leq z)$. In this paper, we interpret Z as a return to be maximized. The **Value-at-Risk** (VaR) at confidence level $\alpha \in (0,1]$ represents the worst $1-\alpha$ quantile of Z, i.e., $\text{VaR}_\alpha(Z) := \min\{z | F(z) \geq \alpha\}$. Analogously, the **Conditional-Value-at-Risk** (Artzner et al. 1999) at confidence level α is defined as

$$\text{CVaR}_\alpha(Z) := \min_{w \in \mathbb{R}} \left\{ w + \frac{1}{\alpha} \mathbb{E}\left[\max(Z - w, 0)\right] \right\}. \tag{2}$$

If Z has a continuous distribution, the CVaR measure can be written as

$$\text{CVaR}_\alpha(Z) := \mathbb{E}\left[z \,|\, z \leq \text{VaR}(Z)\right]. \tag{3}$$

Intuitively, the $\text{CVaR}_\alpha(Z)$ measure can therefore be viewed as the mean over the α-quantile worst values of Z. Since Z represents returns in our case, the worst values of Z can be seen as the ones where most risk has been incurred. This last interpretation is particularly attractive, as it makes the CVaR_α easy to understand for non-experts who might be involved in the design of any risk-sensitive model in safety-critical domains.

CVaR Reinforcement Learning

To measure the level of risk associated with a policy π, the CVaR_α measure defined in (3) can be applied to the random discounted return $\mathcal{J}(\pi)$ defined in (1). Optimizing for this yields the CVaR RL objective

$$\pi^* = \arg\max_\pi \text{CVaR}_\alpha\left(\mathcal{J}(\pi)\right). \tag{4}$$

This objective, which differs from the traditional expectation maximization goal, can be viewed as finding a risk-sensitive policy. Indeed, by having the optimal CVaR_α return, the optimal policy π^* is maximizing the expectation over its worst α-percentile trajectories. Since the expectation is now taken over a small, disadvantageous subset of all returns rather than over all of them, the optimal policy is more sensible towards avoiding the probability of large negative returns. In practice, this means that optimizing for the CVaR RL objective (4) will produce policies that accept to reduce their expected performance, so long as it means they avoid catastrophic trajectories in return.

Let us take a moment to note here that it is well known that the optimal policy π^* for the CVaR RL objective can be history-dependent (Shapiro, Dentcheva, and Ruszczyński 2014), meaning that there are cases where the agent needs to know what previous rewards were collected to achieve CVaR optimal returns. We follow previous work on CVaR RL (Keramati et al. 2020; Dabney et al. 2018) and limit our analysis to stationary, history independent policies which can be suboptimal but typically achieve high CVaR nonetheless.

3 Related Work

Adversarial Reinforcement Learning

Although typically used in the supervised learning setup, adversarial learning (Kurakin, Goodfellow, and Bengio 2017) has already been applied in RL with the aim of increasing task difficulty. Dennis et al. (2020); Tobin et al. (2017) use an adversary to design hard environment instances in order to optimize the robustness of the policy learned by a RL protagonist. This is achieved by allowing the adversary to select

[1]For ease of notation, we will only write π when it is clear that we are talking about a parametrized policy π_θ. This applies for any parametrized function throughout the paper.

a different environment from a constrained set at the beginning of each episode. The adversary is therefore limited to a single move (at the beginning of each episode) and its actions are not budgeted.

More similar to the current work, Mandlekar et al. (2017) use an adversary to perturb the action selection policy of a RL protagonist at every time step. However, their adversary consists in gradient-based perturbations which do not adapt to the policy of the protagonist. In the formulation tackled in the current work, the antagonist strategy is jointly learned with the protagonist strategy, resulting in an antagonist strategy adapted to battle the protagonist.

Risk-Sensitive RL

Broadly speaking, risk-sensitivity in RL represents the ensemble of methods that aim at reducing the level of risk associated with the learned policy. Different risk measures have been proposed in the literature to address this problem.

First, numerous approaches have been proposed to explicitly balance the expectation and variance of the return of a policy (Tamar and Mannor 2013; Pan et al. 2019). Contrary to our CVaR objective which only assumes the returns to be bounded, such approaches are valid only under the assumption that returns follow a normal distribution.

One other popular alternative to incorporate risk-sensitivity is the use of exponential utility functions (Mihatsch and Neuneier 2002; Fei et al. 2020). In these approaches, the return landscape is reshaped into a convex set, essentially achieving balance between expectation and worst-case maximization of the return. The reshaping is based on applying an exponential function and is regulated by a user-specified λ trade-off parameter. However, it is difficult to interpret what different values of λ represent regarding the balance between mean and minimal return (Gosavi, Das, and Murray 2014), in contrast with the straightforward interpretation of our proposed CVaR measure.

Conditional-Value-at-Risk RL

Many algorithms have been proposed to find optimal policies with respect to the CVaR criterion. Proposed algorithms range from policy gradient (Tamar et al. 2015) and Q-learning (Chow et al. 2015), to actor-critic methods (Tamar and Mannor 2013). Unlike our algorithm that is compatible with modern deep learning, the above approaches are all limited to either tabular or low-dimensional action and state space settings.

More recently, distributional RL (Bellemare, Dabney, and Munos 2017; Dabney et al. 2018) has been used to learn CVaR optimal policies in high-dimensional settings (Keramati et al. 2020; Zhang and Weng 2021). This approach has seen growing usage, largely due to the empirical efficiency of distributional RL and its ability to incorporate a wide variety of risk measures in its objective, not exclusively the CVaR. Instead of relying on distributional RL, our paper presents a game-theoretic perspective which can be used on top of any conventional RL algorithm.

The closest work to ours would be Robust Adversarial Reinforcement Learning (RARL) (Pinto et al. 2017), where a CVaR optimal policy is learned by allowing an antagonist to apply external forces or disturbances to the environment model. Their antagonist's action space is however rather different, as it can only change parameters of the model within a given range at every time step. In contrast, our approach lets the antagonist explicitly change the model's transition probabilities and we limit the antagonist's perturbation amount over a whole trajectory rather than at every time step. Moreover, unlike their CVaR formulation which is essentially only intuitive, we establish a clear, theoretically justified, connection between our antagonist's budget and the α-quantile the protagonist's policy is optimized for.

4 Game Setting

Before diving into the explanation of our proposed adversarial game, let us first present the two assumptions necessary on the target MDP for the game to be applicable.

Assumption 1. *The MDP has stochastic state transitions $\mathcal{P}$.*

The stochasticity is required because it is precisely upon those state transitions that the antagonist will be acting. This assumption is the least restrictive, as a large number of MDPs are already stochastic in nature. Furthermore, any deterministic transition function $\mathcal{P}$ could be made stochastic by adding a noise component, either via ε-random sampling for discrete states or adding a Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma)$ for continuous states.

Assumption 2. *It is possible to access the state transitions $\mathcal{P}$ and arbitrarily modify their non-zero values.*

This assumption is the one that allows the antagonist's actions to be registered. This is a much more restrictive assumption and is only likely to be met in artificial environments over which the user has a lot of control. Notably, given an MDP with added noise as described above, it should be relatively straightforward to meet this assumption by accessing the analytical formulation of the added noise component.

Formulation

We aim to learn a risk-sensitive policy with respect to the CVaR risk measure. To do so, we leverage Chow et al. (2015)'s model perturbation framework. Namely, we design a zero-sum adversarial game where a protagonist tries to learn a policy π_θ to maximize its return while an antagonist tries to learn a perturbation function Λ_ω that instead minimizes the protagonist's return.

To explain our game setting, let us first define time step transition dynamics $\mathcal{P}_t = \mathcal{P}(\cdot \mid s_t, a_t) \in [0,1]^{|\mathcal{S}|}$ from a given trajectory $\tau = \{(s_t, a_t, r_t)\}_{t=1}^{T}$, where T is the trajectory's duration. At time step t, the antagonist is given direct access to $\mathcal{P}_t$ and allowed to perturb it with a *multiplicative probability perturbation* $\delta_t \in (0,\infty)^{|\mathcal{S}|}$, yielding perturbed transitions $\hat{\mathcal{P}}_t = \mathcal{P}_t \circ \delta_t$, for $\circ$ the Hadamard (pointwise) product. A perturbation δ is admissible if it generates a valid probability distribution $\hat{P}$. Let Δ^t be the set of all perturbations δ_t admissible for a given $\mathcal{P}_t$ and $\Delta = \Delta^1 \times ... \times ... \Delta^T$ be the set of all possible perturbations of a trajectory τ. We can define a trajectory perturbation budget $\eta \geq 1$ and constrain Δ_η to contain only perturbations within an η budget

by posing the admissible perturbation envelope

$$\Delta_\eta = \left\{\Delta = \{\delta_t\}_{t=1}^{T} \,|\, \delta_1(s_1) * ... * \delta_T(s_T) \leq \eta\right\}, \quad (5)$$

where $\delta_t(s_i)$ represents the antagonist's perturbation towards next state s_i at time step t. Note here that, since perturbations are multiplicative in nature, $\eta = 1$ represents an empty budget, meaning that the antagonist could not apply any modifications to next state transitions.

This definition and limitation of the antagonist's actions enjoy two principal interesting properties. First, since the antagonist is only given a limited budget for each trajectory, they have to be efficient with their perturbations, trying to maximize the damage to the protagonist's return without wasting their budget. Intuitively, the role of the budget can be seen as a way to ensure that the antagonist cannot force the worst possible scenario to occur at every time step. Secondly, the fact that the antagonist perturbations are done via a Hadamard product constrains the perturbations to states with non-zero transition probability by default. Essentially, this means that the antagonist is only allowed to modify the distribution over next states that were already reachable.

The main idea of this paper is that an antagonist Λ can be learned to select perturbations within the admissible perturbation envelope Δ_η defined in (5), attempting to minimize the protagonist π's reward. Combining the protagonist and antagonist naturally produces the two-player zero-sum game

$$\max_{\pi} \min_{\Lambda} \mathbb{E}\left[\mathcal{J}^\eta(\pi, \Lambda)\right], \quad (6)$$

where $\mathbb{E}[\mathcal{J}^\eta(\pi, \Lambda)]$ represents the expected return collected from a protagonist π with admissible perturbations sampled from an antagonist Λ respecting a budget η. Interestingly enough, this formulation implies that both players' goals are to respectively minimize and maximize the expected return, retrieving the classical RL objective. This notably means that any classical RL algorithm like actor-critic methods or Deep Q-Network may be used without fundamental changes for our game. Moreover, the embedded flexibility of the game also allows for the underlying MDP to have continuous action and/or state spaces.

We note that both the protagonist and antagonist players are well defined under the MDP framework for a shared environment. On one hand, the protagonist player's MDP remains exactly the same, with the only modification that next states are sampled from $\hat{\mathcal{P}}_t$ rather than $\mathcal{P}_t$. On the other hand, the antagonist receives as input states $s'_t = (\mathcal{P}_t, \eta_t) \in \mathcal{S}' = [0,1]^{|\mathcal{S}|} \times [1, \infty)$ and outputs perturbed transitions $\hat{\mathcal{P}}_t = \mathcal{P}_t \circ \delta_t$ subject to $\hat{\mathcal{P}}_t$ being a distribution over next states and $\delta_t(s) \leq \eta_t$ for all states. Next antagonist states s'_{t+1} are selected by first sampling the next protagonist state s_{t+1} from $\hat{\mathcal{P}}_t$, then sampling the next protagonist action a_{t+1} from $\pi(s_{t+1})$, and finally observing the next transition distribution $\mathcal{P}_{t+1}$. Lastly, since the game is zero-sum, the antagonist's reward is simply $-r_{t+1}$. An overview of the interaction between the protagonist, antagonist and environment is shown in Figure 1.

Connection to CVaR RL

The solution concept to the game (6) is given by its *Minimax Equilibrium*.

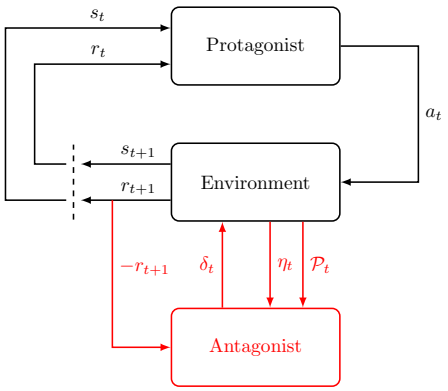


Figure 1: Overview of the proposed game's interaction loop. Components that deviate from the standard RL training loop are in red.

Definition 1. *(Minimax Equilibrium). A Minimax Equilibrium (π^*, Λ^*) is defined as a point where*

$$\mathbb{E}\left[\mathcal{J}^\eta(\pi, \Lambda^*)\right] \leq \mathbb{E}\left[\mathcal{J}^\eta(\pi^*, \Lambda^*)\right] \leq \mathbb{E}\left[\mathcal{J}^\eta(\pi^*, \Lambda)\right], \quad (7)$$

for which the inequalities hold for any other protagonist π or antagonist Λ.

At this equilibrium point, we have the following interesting result.

Lemma 1. *Suppose π_θ and Λ_ω are expressive enough player parametrizations, meaning that they can represent the optimal players π^* and Λ^*. Then, at a Minimax Equilibrium point (π^*, Λ^*), the protagonist's policy π^* is CVaR_α optimal for $\alpha = \frac{1}{\eta}$, where η is the antagonist Λ's perturbation budget:*

$$\max_{\pi} \min_{\Lambda} \mathbb{E}\left[\mathcal{J}^\eta(\pi, \Lambda)\right] = \max_{\pi} \text{CVaR}_{\frac{1}{\eta}}\left[\mathcal{J}(\pi)\right].$$

Proof. Proposition 1 from Chow et al. (2015) established that

$$\inf_{\Lambda'} \mathbb{E}\left[\mathcal{J}^\eta(\pi, \Lambda')\right] = \text{CVaR}_{\frac{1}{\eta}}\left[\mathcal{J}(\pi)\right].$$

If the parametrization of the antagonist Λ is expressive enough, the $\inf$ operation will return an antagonist in the search space, implying that one can look for the best available antagonist for a protagonist to find the CVaR of its returns. The result follows by taking the $\max$ over all possible protagonists, where we once again use the fact that π is expressive enough to guarantee the existence in the parameter search space. □

The result in Lemma 1 is satisfying in two ways. First, it directly connects our adversarial framework to risk-sensitivity for the learned protagonist policy, even though the protagonist and antagonist's objectives remain axed on expectation maximization. Secondly, the equation establishes that the α confidence interval in the CVaR_α objective is directly linked to the budget perturbation η via $\alpha = \frac{1}{\eta}$.

Given our parametrized setting where players are learned rather than exactly computed, we are actually more interested in the notion of approximate equilibrium, which is more likely to be encountered in practice.

Definition 2. *(ε-Minimax Equilibrium). Let $V^* := \mathbb{E}\left[\mathcal{J}^\eta(\pi^*, \Lambda^*)\right]$, the value at the Minimax Equilibrium (π^*, Λ^*). An ε-Minimax equilibrium is a pair of players (π', Λ') where*

$$\begin{aligned} & V^* - \min_{\Lambda} \mathbb{E}\left[\mathcal{J}^\eta(\pi', \Lambda)\right] \leq \varepsilon \\ \textit{and} \quad & \max_{\pi} \mathbb{E}\left[\mathcal{J}^\eta(\pi, \Lambda')\right] - V^* \leq \varepsilon \end{aligned} \tag{8}$$

Interestingly, this approximate equilibrium yields an approximately CVaR-optimal policy.

Theorem 1. *Suppose π_θ and Λ_ω are expressive enough player parametrizations. Then, at an ε-Minimax Equilibrium point (π', Λ'), the protagonist's policy π' is at most ε-suboptimal:*

$$\max_{\pi} \text{CVaR}_{\frac{1}{\eta}}\left[\mathcal{J}(\pi)\right] - \text{CVaR}_{\frac{1}{\eta}}\left[\mathcal{J}(\pi')\right] \leq \varepsilon$$

Proof. Taking the first inequality in (8), we have

$$\begin{aligned} & V^* - \min_{\Lambda} \mathbb{E}\left[\mathcal{J}^\eta(\pi', \Lambda)\right] \leq \varepsilon \\ \iff \quad & V^* - \text{CVaR}_{\frac{1}{\eta}}\left[\mathcal{J}(\pi')\right] \leq \varepsilon \\ \iff \quad & \max_{\pi} \text{CVaR}_{\frac{1}{\eta}}\left[\mathcal{J}(\pi)\right] - \text{CVaR}_{\frac{1}{\eta}}\left[\mathcal{J}(\pi')\right] \leq \varepsilon, \end{aligned}$$

first leveraging Proposition 1 from Chow et al. (2015) and then using Lemma 1. □

5 Gradient-based algorithm

Having established the favorable properties of an approximate equilibrium of our proposed game, we only need to find an algorithm that can reliably reach it. First note that, in order for the algorithm to be able to tackle challenging problems, we wish both players to be used with high-capacity function approximators such as neural networks. Therefore, the algorithm we are looking for should not only converge towards the Minimax Equilibrium, but also do so using gradient-based updates.

However, computing the solution to the minmax game in (6) for players represented by neural networks is not a simple task. Indeed, the dependence of each player's reward on the other makes the optimization objective non-stationary (Fiez, Chasnov, and Ratliff 2020), hampering naive gradient-based algorithms' convergence properties. To this end, we propose to view our game as a Stackelberg game (Von Stackelberg 2010), which allows us to derive well-defined gradient-based algorithms.

Stackelberg Algorithm

An essential step to achieve stable learning of the game is to take into account the game's structure in the parameter updates. Although there aren't good gradient-based optimization updates specifically designed for minimax games like ours, such updates do exist for Stackelberg games. We therefore cast our game under the Stackelberg formulation, allowing us to derive our desired gradient updates.

A two-player Stackelberg game is an asymmetric game played with a leader l and a follower f. Both players aim to maximize their respective payoff functions $u_l(\theta_l, \theta_f)$ and $u_f(\theta_l, \theta_f)$, where θ represents the parameters of a given player. The particularity in this game is that the follower's parameters always represent a best-response with respect to the leader's parameters. Accordingly, the leader can then pick its parameters by taking for granted that the follower's parameters will be optimal with respect to them. This yields the following optimization problem for the leader

$$\theta_l \in \arg\max_{\theta} \left\{ u_l(\theta, \theta_f^{'}) \,\middle|\, \theta_f^{'} \in \arg\max_{\theta_f} u_f(\theta_l, \theta_f) \right\},$$

and for the follower

$$\theta_f \in \arg\max_{\theta} u_f(\theta_l, \theta).$$

The key intuition to note here is that, since the follower always has optimal parameters with respect to the leader, then the follower parameters may be seen as an implicit function taking as input the leader's parameters $\theta_f(\theta_l)$. In turn, this allows the leader to exploit this optimal-response form of its follower to update its own parameters towards its own goal.

Intuitively, updating both player's parameters may be viewed as a bi-level optimization procedure. In order to solve a Stackelberg game, we can therefore simply alternate between solving each player player's respective optimization problem, producing a simple yet effective algorithm.

As Stackelberg games are a generalization of minimax games such as ours, this Stackelberg algorithm only requires us to establish a leader and a follower to be able to find our game's equilibrium. We arbitrarily select the protagonist as the leader and the antagonist as their follower, but show in the following convergence analysis that this choice can be reversed without problem.

Convergence Analysis

Given some assumptions on the environment MDP, the Stackelberg algorithm described previously converges to the equilibrium of our game.

Assumption 3. *The reward function $\mathcal{R}$ and all transition functions $\mathcal{P}$ are both smooth and convex.*

Assumption 3 is quite restrictive in practice. Indeed, while the smoothness portion of it is likely to hold in practice, e.g. many real-world environments like robotics can be considered smooth (Pirotta, Restelli, and Bascetta 2015) and the same can be said about neural networks with ReLU activations (Petersen and Voigtländer 2018), the convexity assumption is much less likely to hold. We however wish to emphasize that previous work (Pinto et al. 2017; Perolat et al. 2015; Patek 1997) also uses this assumption in the context of minimax objectives and still obtain good empirical results, even when the convexity assumption is not met.

When Assumption 3 is met, we have the following lemma.

Lemma 2. *The expected return $\mathbb{E}[\mathcal{J}^\eta(\pi, \Lambda)]$ is convex with respect to π and concave with respect to Λ.*

Proof. This follows from the convexity of $\mathcal{R}$ and all $\mathcal{P}$. □

Building upon Lemma 2, we have the following theorem.

Theorem 2. *The proposed Stackelberg algorithm will converge towards the Minimax Equilibrium of the game.*

Proof. This is a well-known result for Stackelberg games in the two-player zero-sum setting with a convex-concave playoff function (see Fiez, Chasnov, and Ratliff (2020)), a condition guaranteed by Lemma 2. □

Remember that the (approximate) Minimax Equilibrium of the game contains a CVaR (approximately) optimal policy. It follows that Theorem 2 constitutes the proof that our overall framework (the game and the proposed algorithm), is indeed a theoretically justified way to attain a CVaR optimal policy. A last theoretical result follows from the game's convex-concave objective.

Corollary 1. *For the proposed game, we have*

$$\min_{\Lambda} \max_{\pi} \mathbb{E}\left[\mathcal{J}^{\eta}(\pi, \Lambda)\right] = \max_{\pi} \min_{\Lambda} \mathbb{E}\left[\mathcal{J}^{\eta}(\pi, \Lambda)\right].$$

Proof. This follows from the convexity of the payoff function $\mathbb{E}\left[\mathcal{J}^{\eta}(\pi, \Lambda)\right]$ (Lemma 2), which allows us to apply the minimax theorem (Sion 1958), inversing the $\min$ and $\max$ terms. □

This last corollary is of separate interest, as it notably implies that the Stackelberg algorithm can take any of the players as its leader.

Practical Considerations

In practice, we adopt a few relaxations of the exact Stackelberg algorithm described above. The adopted relaxations have proven effective in domains like GANs (Heusel et al. 2017; Metz et al. 2017) or model-based RL (Rajeswaran, Mordatch, and Kumar 2020), to name a few. Precisely, we (i) implement the *almost optimal* updates of the follower by doing $K_{\text{ant}} > 1$ gradient steps of the antagonist (follower) before updating the protagonist (leader). Also, we (ii) use the first-order approximation of the joint gradient to update the leader rather than the computationally expensive true Jacobian term.

Another important consideration is the fact that the protagonist needs to output perturbations within the admissible perturbation envelope for every collected trajectory. To do so, we propose to simply keep track of the remaining antagonist budget after every time step t, updating it according to the realized perturbation (consumed budget) $\delta_t(s_{t+1})$. A convex transformation and a rescaling function are applied to the budget predictions in order to constrain the predictions to give a probability distribution when multiplied with the state probability. A pseudocode of the overall algorithm incorporating all the above relaxations can be seen in Algorithm 1.

Algorithm 1: CVaR Adversarial Stackelberg Algorithm

Require: π_θ (protagonist), Λ_ω (antagonist), η (perturbation budget), K_{ant} (number of intermediate antagonist steps)

$N_{\text{updates}} = 0$
while training not done **do**
  Get initial state s_t
  $\eta_\tau = \eta$ ▷ Remaining antagonist budget
  while s_t not terminal **do**
    $a_t \sim \pi_\theta(s_t), \mathcal{P}_t = \mathcal{P}(s_t, a_t)$
    $\delta_t = \Lambda_\omega(\mathcal{P}_t, \eta_\tau)$
    $\hat{\mathcal{P}}_t = \mathcal{P}_t \circ \delta$
    $s_{t+1} \sim \hat{\mathcal{P}}_t, r_{t+1} \sim \mathcal{R}(s_{t+1})$
    $\eta_\tau = \frac{\eta_\tau}{\delta_t(s_{t+1})}$ ▷ Update remaining budget
  end while
  Update θ or ω according to N_{updates} and K_{ant}.
  $N_{\text{updates}} = N_{\text{updates}} + 1$
end while

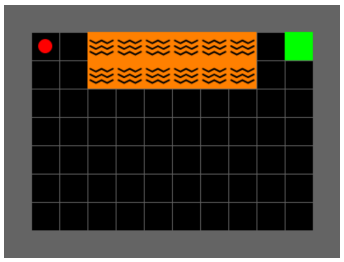


Figure 2: Overview of our Lava gridworld environment.

6 Experiments

We conduct experiments on a variation of the Gym Minigrid Lava environment (Chevalier-Boisvert, Willems, and Pal 2018) displayed in Figure 2. In our environment, a protagonist player walks around in a 7×10 grid with the aim to reach a goal location (top right corner of the grid) as fast as possible given their initial location (top left corner of the grid). A lava zone lies between the starting point and the goal (wavy orange squares). A state in this game corresponds to the (x, y) coordinates of the protagonist and the state space $\mathcal{S}$ corresponds to the set of all possible coordinates in the grid. The action space $\mathcal{A}$ for the protagonist corresponds to moving into one of the four cardinal directions (up, down, right, left). The environment is stochastic with probability ($p = 0.05$) for the protagonist to perform a random action instead of the chosen action a_t. At each time step t of a game episode, the protagonist incurs a reward penalty of -0.035, motivating the protagonist to reach the goal using the shortest path. The episode ends when the protagonist either (i) reaches the goal, resulting into a reward of $+1$, (ii) falls into the lava, resulting into a reward of -1, or (iii) when the maximum number of 40 steps is reached within the episode.

As the protagonist aims to reach the goal location, the antagonist rather aims to push the protagonist into the lava zone. Since the antagonist acts upon a limited budget, the protagonist is able to reduce the chances of falling into the lava by just moving farther away from the lava. As the protagonist also incurs penalties for every step taken, they nonetheless want to avoid making unnecessary steps, eventually needing to be close to the lava zone. This dilemma between minimizing the chances of falling into the lava while also making sure that no unnecessary steps are taken serves as a good minimal illustration of a safety-critical environment. Indeed, the more weight the protagonist will put on

avoiding large negative rewards, the more it will be willing to take the time to get to the lower cases of the grid before crossing from left to right.

Implementation details

Both players are trained using Actor-Critic PPO algorithms (Schulman et al. 2017). To keep our antagonist's state and action spaces reasonable, we represent next state transitions $\mathcal{P}_t$ as the concatenation of the protagonist current coordinates (x_t, y_t), the probability vector p_t of each action a being executed and the value of their remaining perturbation budget η_t. The players are trained jointly over 5M time steps, following Algorithm 1 and the antagonist's budget constraint is enforced using a convex rescaling method. The number K_{ant} of antagonist updates between the policy updates has been set to 2, after evaluating over $K_{\text{ant}} = 2, 4, 8$. An update of the parameters occurs every 10k time steps.

Both algorithms use an Adam optimizer with their own decaying learning rate that decreases linearly from 0.005 to 0.0001 each time their parameters are updated. Further details on the hyper-parameters and implementation are discussed in our publicly available implementation. Our codebase is made publicly available[2] for full reproduction of the experiments and figures.

Evaluation

We consider three instances of the described game: the case where there is no antagonist, which corresponds to having an antagonist with a perturbation budget $\eta = 1$ (as a reference for comparison), as well as having an antagonist with two different budgets, $\eta \in \{25, 100\}$. Without an antagonist, the game corresponds to the classical problem of maximizing the excepted returns. When the antagonist has a positive perturbation budget, this corresponds to optimizing a CVaR-optimal policy, i.e. $\text{CVaR}_{0.04}$ for $\eta = 25$ and $\text{CVaR}_{0.01}$ for $\eta = 100$. In order to evaluate the robustness of the proposed approach, we replicate each instance of the game 10 times, each time with a different random seed for the environment (governing its randomness), for the protagonist (governing the initialization of its policy and the action selection process), and for the antagonist (governing the initialization of its policy and the perturbation prediction process).

In order to visualize the protagonist strategy in any game instance, we execute the policy learned by the protagonist agent in a deterministic environment ($p = 0$) without the antagonist. This allows to observe the decisions made by the protagonist agent without perturbations. To visualize the antagonist strategy in a given game instance, we execute the policy learned by a protagonist under this instance, i.e. the stochastic environment ($p = 0.05$) in presence of the antagonist. For each state in the grid, we estimate the probability that the protagonist performs a different action than their preferred action in that state. A probability estimate equal to 0.05 indicates that the antagonist does not hamper the protagonist on this location since this corresponds to the environment randomness (p). A probability greater or lesser than 0.05 indicates that the antagonist is disturbing the transition probabilities at this location since the probability differs from the natural environment randomness.

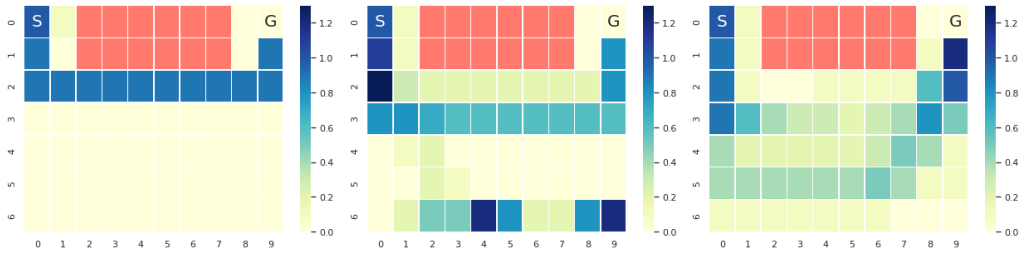
[2]https://github.com/TortillasAlfred/CvarAdversarialRL

Results

Figure 3 displays the learned protagonist policy averaged over the 10 seeds for each of the considered game instances. As expected, Figure 3a shows that the protagonist learns the shortest path to the goal in the absence of an antagonist ($\eta = 1$). However, when there is an antagonist ($\eta > 1$), Figures 3b and 3c show that the protagonist learns a careful path that deviates from the lava. More specifically, the stronger the antagonist (the larger the perturbation budget η), the more careful the protagonist has to be in order to avoid getting pushed into lava. This confirms that the presence of an antagonist agent when learning the protagonist policy results in a safer policy.

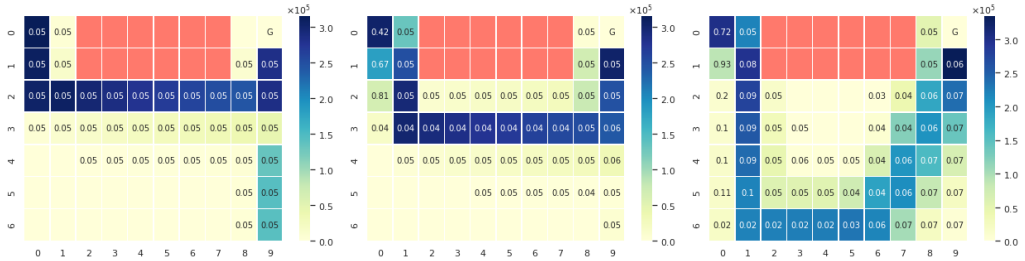
Note: The noticeable noise in the averaged protagonist policy displayed in Figure 3b is due to convergence instabilities, which caused the protagonist to fail reaching the goal in 2 out of the 10 tested seeds. This may be due to the protagonist being particularly unlucky in these realizations of the environment and/or in the initialization of their policy. In other applications of game theory with gradient-based algorithms (e.g. Generative Adversarial Networks (Goodfellow et al. 2020)), it is known that systems learning in an adversarial manner are subject to instabilities (Berthelot, Schumm, and Metz 2017). Despite these challenges inherent to the training procedure, the protagonist consistently displayed convergence to the same policy in 8/10 seeds, still demonstrating the robustness of the method to diverse training scenarios.

Figure 4 shows 300k trajectories obtained by executing a protagonist (single seed) in a stochastic environment ($p = 0.05$) in presence of the antagonist used during their training. The number on a given square indicates the probability that the protagonist will perform a different action than a_t when s_t corresponds to that location. This includes both the natural environment randomness as well as the influence of the antagonist. Figures 4b and 4c show that the antagonist is mostly active at the beginning of a game episode, regardless of the amount of perturbation budget. The beginning of the game is indeed the riskiest situation for the protagonist since the starting point is very close to the lava (top-left corner). More sophisticated adversarial strategies emerge when the antagonist benefits from a higher budget (see Figure 4c). For instance, when the protagonist gets far from the lava, we see that the antagonist exploits the borders of the grid to force the protagonist into accumulating moving reward penalties. Also, when the protagonist is at the extreme bottom of the grid (far from both the lava and the goal), the perturbation probability drops below 0.05 (the natural environment randomness), meaning that the antagonist tends to give more leverage to the protagonist in this region. The antagonist does this in order to recover its budget to get more powerful when the protagonist gets closer to the goal.



(a) No antagonist ($\eta = 1$) (b) Perturbation budget $\eta = 25$ (c) Perturbation budget $\eta = 100$

Figure 3: Protagonist policy (averaged over 10 random seeds) learned on a stochastic environment with different antagonist perturbation budget. **S** and **G** respectively denote the starting location of the protagonist and the goal to reach.



(a) No antagonist ($\eta = 1$) (b) Perturbation budget $\eta = 25$ (c) Perturbation budget $\eta = 100$

Figure 4: Antagonist policy averaged over 300k trajectories of a protagonist learned on a stochastic environment with different antagonist perturbation budget. Numbers indicate the probabilities that the protagonist may perform an action different from the selected action a_t when their state s_t corresponds to the given location. In order to display accurate estimators, only the probabilities for the locations that the agent visited at least 500 times are displayed. **G** denotes the goal.

7 Conclusion

In this work, we presented an adversarial game designed to compute risk-sensitive policies with respect to the Conditional Value-at-Risk (CVaR) risk measure. Our adversarial game is particular in the sense that it limits an antagonist's perturbations to be within a specified budget and to only change next state transitions for a protagonist. We then presented a gradient-based algorithm based on the Stackelberg formulation of the game to solve it, both proving its convergence under sufficient conditions and presenting key considerations for a practical implementation. We put our method to the test on an artificial risky environment, illustrating that increasing the antagonist's budget indeed leads a more cautious protagonist policy. Given the fact that this is the first game-theoretic perspective on CVaR RL, we expect this paper to be a building block towards connecting risk-sensitivity in RL with the field of Game Theory.

One possibly interesting application of our method is with regards to the Sim2Real domain, where one wishes to optimize to train an RL agent on a high-quality simulation environment before real-life deployment. Since this domain is naturally concerned with risk-sensitivity and supposes access to a simulator to access and perturbate the true next state transitions, it appears highly likely that our proposed approach can be of use in this context.

In future work, we would like to apply our game to a concrete Sim2Real application to try and leverage the empirical potential of the method. Also, in light of the instability observed in our own empirical results, we would like to address the issue of sensibility to hyperparameters in our practical gradient-based algorithm.

References

Artzner, P.; Delbaen, F.; Eber, J.-M.; and Heath, D. 1999. Coherent measures of risk. *Mathematical finance*, 9(3): 203–228.

Baek, D.; Hwang, M.; Kim, H.; and Kwon, D.-S. 2018. Path Planning for Automation of Surgery Robot based on Probabilistic Roadmap and Reinforcement Learning. In *2018 15th International Conference on Ubiquitous Robots (UR)*, 342–347.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 449–458. PMLR.

Berthelot, D.; Schumm, T.; and Metz, L. 2017. BEGAN: Boundary Equilibrium Generative Adversarial Networks. *CoRR*, abs/1703.10717.

Chevalier-Boisvert, M.; Willems, L.; and Pal, S. 2018. Minimalistic Gridworld Environment for OpenAI Gym. https://github.com/maximecb/gym-minigrid.

Chow, Y.; Tamar, A.; Mannor, S.; and Pavone, M. 2015. Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach. In *NIPS*, 1522–1530.

Dabney, W.; Ostrovski, G.; Silver, D.; and Munos, R. 2018. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, 1096–1105. PMLR.

Dennis, M.; Jaques, N.; Vinitsky, E.; Bayen, A. M.; Russell, S.; Critch, A.; and Levine, S. 2020. Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Fei, Y.; Yang, Z.; Chen, Y.; Wang, Z.; and Xie, Q. 2020. Risk-Sensitive Reinforcement Learning: Near-Optimal Risk-Sample Tradeoff in Regret. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Fiez, T.; Chasnov, B.; and Ratliff, L. 2020. Implicit Learning Dynamics in Stackelberg Games: Equilibria Characterization, Convergence Analysis, and Empirical Study. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 3133–3144. PMLR.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144.

Gosavi, A. A.; Das, S. K.; and Murray, S. L. 2014. Beyond exponential utility functions: A variance-adjusted approach for risk-averse reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 1–8. IEEE.

Gottesman, O.; Johansson, F.; Komorowski, M.; Faisal, A.; Sontag, D.; Doshi-Velez, F.; and Celi, L. A. 2019. Guidelines for reinforcement learning in healthcare. *Nature medicine*, 25(1): 16–18.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Keramati, R.; Dann, C.; Tamkin, A.; and Brunskill, E. 2020. Being optimistic to be conservative: Quickly learning a cvar policy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 4436–4443.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2017. Adversarial Machine Learning at Scale. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, 661–670. New York, NY, USA: Association for Computing Machinery. ISBN 9781605587998.

Mandlekar, A.; Zhu, Y.; Garg, A.; Fei-Fei, L.; and Savarese, S. 2017. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3932–3939. IEEE.

Metz, L.; Poole, B.; Pfau, D.; and Sohl-Dickstein, J. 2017. Unrolled Generative Adversarial Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Mihatsch, O.; and Neuneier, R. 2002. Risk-sensitive reinforcement learning. *Machine learning*, 49(2): 267–290.

Pan, X.; Seita, D.; Gao, Y.; and Canny, J. 2019. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, 8522–8528. IEEE.

Patek, S. D. 1997. *Stochastic and shortest path games: theory and algorithms*. Ph.D. thesis, Massachusetts Institute of Technology.

Perolat, J.; Scherrer, B.; Piot, B.; and Pietquin, O. 2015. Approximate dynamic programming for two-player zero-sum markov games. In *International Conference on Machine Learning*, 1321–1329. PMLR.

Petersen, P.; and Voigtländer, F. 2018. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108: 296–330.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, 2817–2826. PMLR.

Pirotta, M.; Restelli, M.; and Bascetta, L. 2015. Policy gradient in lipschitz markov decision processes. *Machine Learning*, 100(2): 255–283.

Rajeswaran, A.; Mordatch, I.; and Kumar, V. 2020. A Game Theoretic Framework for Model Based Reinforcement Learning. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 7953–7963. PMLR.

Schubert, F.; Eimer, T.; Rosenhahn, B.; and Lindauer, M. 2021. Automatic Risk Adaptation in Distributional Reinforcement Learning. *CoRR*, abs/2106.06317.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Shapiro, A.; Dentcheva, D.; and Ruszczyński, A. 2014. *Lectures on stochastic programming: modeling and theory*. SIAM.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Sion, M. 1958. On general minimax theorems. *Pacific Journal of mathematics*, 8(1): 171–176.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Tamar, A.; Chow, Y.; Ghavamzadeh, M.; and Mannor, S. 2015. Policy Gradient for Coherent Risk Measures. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 1468–1476.

Tamar, A.; and Mannor, S. 2013. Variance Adjusted Actor Critic Algorithms. *CoRR*, abs/1310.3697.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 23–30. IEEE.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Von Stackelberg, H. 2010. *Market structure and equilibrium*. Springer Science & Business Media.

Zhang, J.; and Weng, P. 2021. Safe Distributional Reinforcement Learning. *CoRR*, abs/2102.13446.