The aim of this document is to describe the CVaR Value Iteration algorithm, it's distributional version and the extension to Q-learning. This is done in order to formulate the CVaR Q-learning convergence problem.
Let's start with some definitions.

0.1 Return

We define the return $Z^\pi(x)$ as a random variable representing the discounted reward along a trajectory generated by the MDP by following policy π, starting at state x:

$$\begin{aligned} Z^\pi(x) &= \sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) \\ x_t &\sim p(\cdot|x_{t-1}, a_{t-1}), a_t \sim \pi, x_0 = x \end{aligned} \tag{1}$$

As a useful notation, we denote $Z^\pi(x,a)$ as the random variable representing the discounted reward along a trajectory generated by first selecting action a and then following policy π.

$$\begin{aligned} Z^\pi(x,a) &= \sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) \\ x_t &\sim p(\cdot|x_{t-1}, a_{t-1}), a_t \sim \pi, x_0 = x, a_0 = a \end{aligned} \tag{2}$$

0.2 CVaR

Let Z be a random variable representing reward, with cumulative distribution function (c.d.f.) $F(z) = \mathbb{P}(Z \le z)$. The Value-at-Risk at confidence level $\alpha \in (0,1)$ is the α-quantile of Z, i.e.

$$\mathrm{VaR}_\alpha(Z) = F^{-1}(\alpha) = \inf\{z | \alpha \le F(z)\} \tag{3}$$

The Conditional Value-at-Risk (CVaR) at confidence level $\alpha \in (0,1)$ is defined as the expected reward of of outcomes worse than the α-quantile (VaR_α):

$$\mathrm{CVaR}_\alpha(Z) = \frac{1}{\alpha}\int_0^\alpha F_Z^{-1}(\beta)\mathrm{d}\beta = \frac{1}{\alpha}\int_0^\alpha \mathrm{VaR}_\beta(Z)\mathrm{d}\beta \tag{4}$$

[1] also showed that CVaR is equivalent to the solution of

$$\mathrm{CVaR}_\alpha(Z) = \max_s \left\{ \frac{1}{\alpha}\mathbb{E}\left[(Z-s)^-\right] + s \right\} \tag{5}$$

where $(x)^- = \min(x,0)$ represents the negative part of x and in the optimal point $s^* = VaR_\alpha(Z)$

$$\mathrm{CVaR}_\alpha(Z) = \frac{1}{\alpha}\mathbb{E}\left[(Z - VaR_\alpha(Z))^-\right] + VaR_\alpha(Z) \tag{6}$$

See Figure 1 for a visualization.
The risk-averse objective we wish to address for a given confidence level α is the following

$$\max_{\pi \in \Pi_H} \mathrm{CVaR}_\alpha(Z^\pi(x_0)) \tag{7}$$

where $Z^\pi(x_0)$ coincides with definition (1).
In words, our goal is to find a general policy $\pi^* \in \Pi_H$, that maximizes conditional value-at-risk of the return, starting in state x_0. We emphasize the importance of the starting state since, unlike the expected value, the CVaR objective is not time-consistent.

0.3 Vale Iteration

Value iteration [2] is a well-known algorithm for computing the optimal (action-)value function and hereby finding the optimal policy. Let us remind ourselves of the Bellman optimality operator $\mathcal{T}$ (**??**):

$$\mathcal{T}Q(x,a) := \mathbb{E}\left[R(x,a)\right] + \gamma\,\mathbb{E}_P\left[\max_{a' \in \mathcal{A}} Q(x', a')\right]$$

or rewritten for the value function V

$$\mathcal{T}V(x) = \max_a \left\{ r(x,a) + \gamma \sum_{s'} p(s'|x,a)V(x') \right\} \tag{8}$$

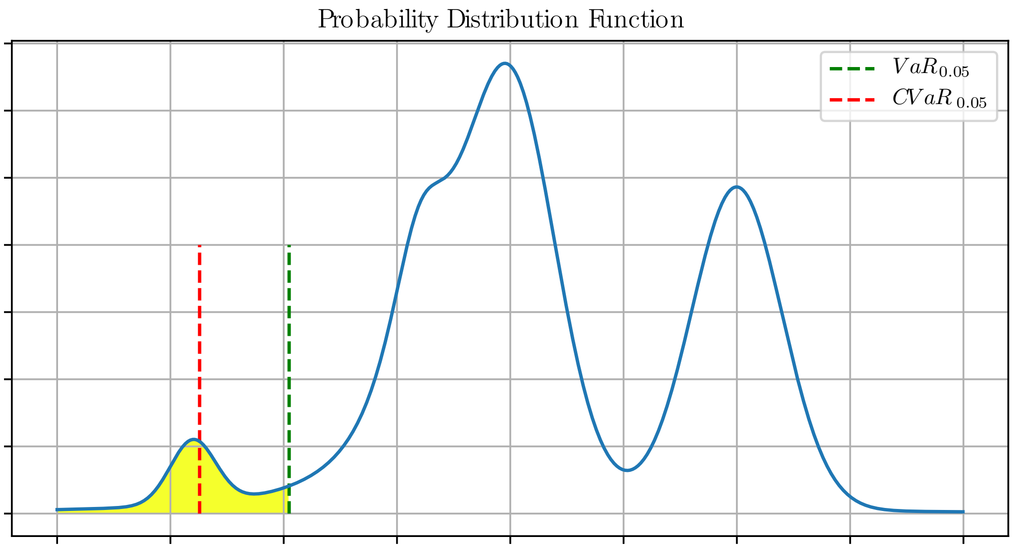

Figure 1: Value-at-Risk and Conditional Value-at-Risk of a general probability distribution with the integral $\alpha = 0.05$ marked in yellow. The main flaw of the VaR metric is clearly visible here, as we could shift the leftmost 'mode' of the distribution into minus infinity and the VaR would remain unchanged, while CVaR would change with the shift.

1 CVaR Value Iteration

[3] present a dynamic programming formulation for the CVaR MDP problem (7). As CVaR is a time-inconsistent measure, their method requires an extension of the state space. A Value Iteration type algorithm is then applied on this extended space and [3] proved it's convergence.
We repeat their key ideas and results bellow, as they form a basis for our contributions presented in later sections.

1.1 Bellman Equation for CVaR

The results of [3] heavily rely on the CVaR decomposition theorem (Lemma 22, [4]):

$$\mathrm{CVaR}_\alpha\left(Z^\pi(x)\right) = \min_{\xi \in \mathcal{U}_{\mathrm{CVaR}}(\alpha, P(\cdot|x,a))} \sum_{x'} p(x'|x,\pi(x))\xi(x')\mathrm{CVaR}_{\xi(x')\alpha}\left(Z^\pi(x')\right) \tag{9}$$

where $\mathcal{U}_{\mathrm{CVaR}}(\alpha, P(\cdot|x,a)) = \left\{\xi : \xi(\omega) \in \left[0, \frac{1}{\alpha}\right], \int_\omega \xi(\omega)\mathbb{P}(\omega)\mathrm{d}\omega = 1\right\}$. The theorem states that we can compute the $\mathrm{CVaR}_\alpha\left(Z^\pi(x,a)\right)$ as the minimal weighted combination of $\mathrm{CVaR}_\alpha\left(Z^\pi(x')\right)$ under a probability distribution perturbed by $\xi(x')$.
Note that the decomposition requires only the representation of CVaR at different confidence levels and not the whole distribution at each level, which we might be tempted to think because of the time-inconsistency issue.
[3] extend the decomposition theorem by defining the *CVaR value function* $C(x,y)$ with an augmented state-space $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y} = (0,1]$ is an additional continuous state that represents the different confidence levels.

$$C(x,y) = \max_{\pi \in \Pi_H} \mathrm{CVaR}_y\left(Z^\pi(x)\right) \tag{10}$$

Similar to standard dynamic programming, it is convenient to work with with operators defined on the space of value functions. This leads to the following definition of the CVaR Bellman operator $\mathbf{T} : \mathcal{X} \times \mathcal{Y} \to \mathcal{X} \times \mathcal{Y}$:

$$\mathbf{T}C(x,y) = \max_a \left[R(x,a) + \gamma \min_{\xi \in \mathcal{U}_{\mathrm{CVaR}}(\alpha, P(\cdot|x,a))} \sum_{x'} p(x'|x,a)\xi(x')C\left(x', y\xi(x')\right)\right] \tag{11}$$

or in our simplified notation, this describes the following relationship:

$$\mathbf{T}\mathrm{CVaR}_y(Z(x)) = \max_a \left[R(x,a) + \gamma \mathrm{CVaR}_y(Z(x,a))\right] \tag{12}$$

[3] further showed (Lemma 3) that the operator $\mathbf{T}$ is a contraction and also preserves the convexity of $y\mathrm{CVaR}_t$. The optimization problem (9) is a convex one and therefore has a unique solution. Additionally, the fixed point of this contraction is the optimal $C^*(x,y) = \max_{\pi\in\Pi} \mathrm{CVaR}_y(Z^\pi(x,y))$ ([3], Theorem 4).
Naive value iteration with operator $\mathbf{T}$ is unfortunately unusable in practice, as the state space is continuous in y. The solution proposed in [3] is then to represent the convex $y\mathrm{CVaR}_y$ as a piecewise linear function.

1.2 Value Iteration with Linear Interpolation

Given a set of $N(x)$ interpolation points $\mathbf{Y}(x) = \left\{y_1,\dots,y_{N(x)}\right\}$, we can interpolate the $yC(x,y)$ function on these points, i.e.

$$\mathcal{I}_x[C](y) = y_i C(x,y_i) + \frac{y_{i+1}C(x,y_{i+1}) - y_i C(x,y_i)}{y_{i+1}-y_i}(y-y_i),$$

where $y_i = \max\{y' \in \mathbf{Y}(x) : y' \le y\}$ The interpolated Bellman operator $\mathbf{T}_\mathcal{I}$ is then also a contraction and has a bounded error ([3], Theorem 7).

$$\mathbf{T}_\mathcal{I} C(x,y) = \max_a \left[R(x,a) + \gamma \min_{\xi\in\mathcal{U}_{\mathrm{CVaR}}(\alpha,P(\cdot|x,a))} \sum_{x'} p(x'|x,a) \frac{\mathcal{I}_{x'}[C](y\xi(x'))}{y} \right] \tag{13}$$

The full value iteration procedure is presented in Algorithm 1.
This algorithm can be used to find an approximate global optimum in any MDP. There is however the issue of computational complexity. As the algorithm stands, the straightforward approach is to solve each iteration of (13) as a linear program, since the problem is convex and piecewise linear, but this is not practical, as the LP computation can be demanding and is therefore not suitable for large state-spaces.

Algorithm 1 CVaR Value Iteration with Linear Interpolation (Algorithm 1 in [3])

```
1: Given:
   • $N(x)$ interpolation points $\mathbf{Y}(x) = \left\{y_1,\dots,y_{N(x)}\right\} \in [0,1]^{N(x)}$ for every $x \in \mathcal{X}$ with $y_i < y_{i+1}$, $y_1 = 0$ and $y_{N(x)} = 1$.
   • Initial value function $C_0(x,y)$ that satisfies:
       1. $yC_0(x,y)$ is convex in $y$ for all $x$
       2. $yC_0(x,y)$ is continuous in $y$ for all $x$
2: Repeat until convergence:
   • For each $x \in \mathcal{X}$ and each $y_i \in \mathbf{Y}(x)$, update the value function estimate as follows:
         $C_{k+1}(x,y_i) = \mathbf{T}_\mathcal{I}[C_k](x,y_i),$
3: Set the converged value iteration estimate as $\widehat{C}^*(x,y_i)$, for any $x \in \mathcal{X}$, and $y_i \in \mathbf{Y}(x)$.
```

2 Efficient computation using quantile representation

We present our original contributions in this section. We first describe a connection between the $y\mathrm{CVaR}_y$ function and the quantile function of the underlying distribution. We then use this connection to formulate a faster computation of the value iteration step, resulting in the first linear-time algorithm for solving CVaR MDPs with bounded error.

Lemma 1. *Any discrete distribution has a piecewise linear and convex $yCVaR_y$ function. Similarly, any piecewise linear convex function can be seen as representing a certain discrete distribution.*
Particularly, the integral of the quantile function is the $yCVaR_y$ function

$$yCVaR_y(Z) = \int_0^y VaR_\beta(Z)\,d\beta \tag{14}$$

and the derivative of the $yCVaR_y$ function is the quantile function

$$\frac{\partial}{\partial y} yCVaR_y(Z) = VaR_y(Z) \tag{15}$$

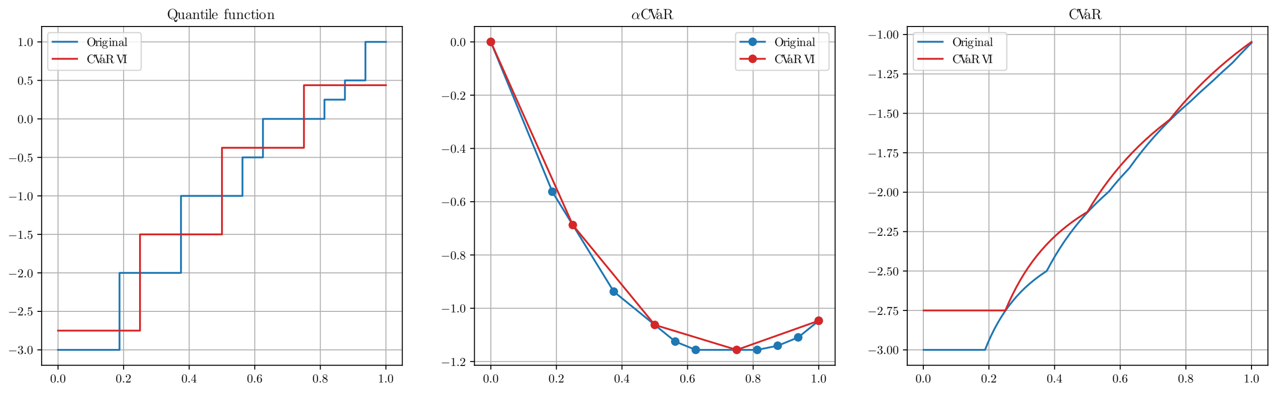

Figure 2: Comparison of a discrete distribution and it's approximation according to the CVaR linear interpolation operator.

Proof. The fact that discrete distributions have a piecewise has already been shown by [1].
According to definition (4) we have

$$y\mathrm{CVaR}_y(Z) = y\frac{1}{y}\int_0^y \mathrm{VaR}_\beta(Z)\mathrm{d}\beta = \int_0^y \mathrm{VaR}_\beta(Z)\mathrm{d}\beta$$

by taking the y derivative, we have

$$\frac{\partial}{\partial y} y\mathrm{CVaR}_y(Z) = \frac{\partial}{\partial y}\int_0^y \mathrm{VaR}_\beta(Z)d\beta = \mathrm{VaR}_y(Z)$$

□

You can get some intuition from Figure 2, where the integral-derivation relationship is clearly visible.
According to Lemma 1, we can reconstruct the $y\mathrm{CVaR}_y$ from the underlying distribution and vice-versa. We utilize the fact that the conversion is linear in the number of probability atoms to formulate a fast way of computing the $\mathbf{T}_\mathcal{I}$ operator.

2.1 CVaR Computation via Quantile Representation

We propose the following procedure: instead of using linear programming for the CVaR computation, we use the underlying distributions represented by the $\alpha\mathrm{CVaR}_\alpha$ function to compute CVaR at each atom. The general steps of the computation are as follows

1. transform $y\mathrm{CVaR}_y$ of each possible state transition to a discrete probability distribution using (15)
2. combine these to to a distribution representing the full state-action distribution
3. compute $y\mathrm{CVaR}_y$ for all atoms using (14)

See Figure 3 for a visualization of the procedure.
Note that this procedure is linear for discrete distributions. To show the correctness of this approach, we formulate it as a solution to the problem (9). Note that we skip the reward and gamma scaling for readability's sake. The extension to the bellman operator is trivial.

2.2 ξ-computation

Similarly to Theorem **??**, we need a way to compute the $y_{k+1} = y_k\xi^*(x_k)$ to extract the optimal policy. We compute $\xi^*(x_k)$ by using the following intuition: y_{k+1} is the portion of $Z(x_{k+1})$ that is present in $\mathrm{CVaR}_{y_k}(Z(x_k))$. In the continuous case, it is the probability in $Z(x_{k+1})$ before the $\mathrm{VaR}_{y_k}(Z(x_k))$ as we show bellow.

Theorem 1. *Let x'_1, x'_2 be two states reachable from state x in a single transition. Let the cumulative distribution functions of the state's underlying distributions $Z(x'_1), Z(x_2)$ be strictly increasing with unbounded support. Then the solution to minimization problem (9) can be computed setting*

$$\xi(x'_i) = \frac{F_{x'_i}(F_x^{-1}(\alpha))}{\alpha} \tag{16}$$

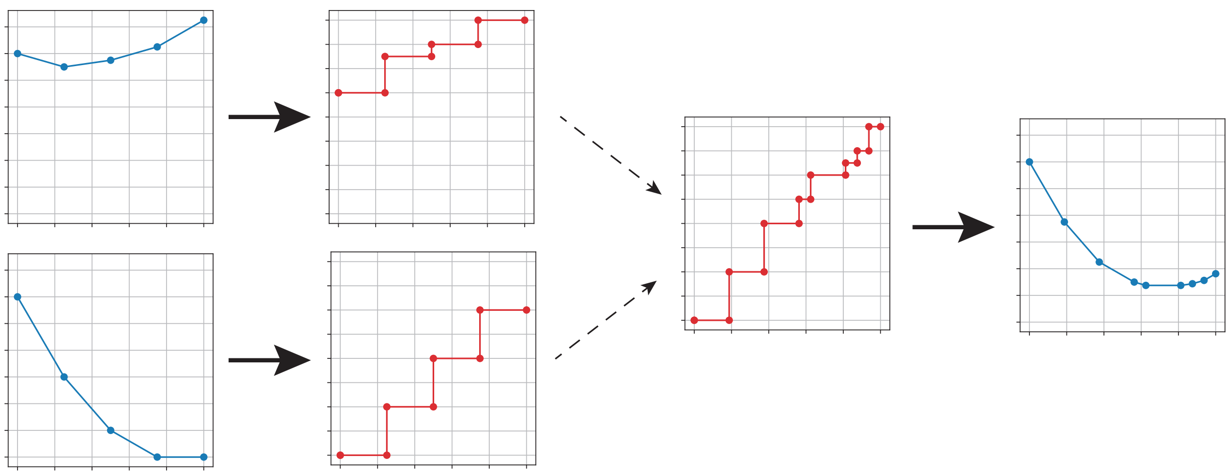

Figure 3: Visualization of the CVaR computation for a single state and action with two transition states. Thick arrows represent the conversion between $y\text{CVaR}_y$ and the quantile function.

2.3 Q-learning

Q-learning ([5]) is an important off-policy temporal difference control algorithm, that works by repeatedly updating the Q value estimate according to the sampled rewards and states using a moving exponential average.

$$\begin{aligned} &Q_{t+1}(x,a) = (1-\beta)Q_t(x,a) + \beta\left[r + \gamma \max_{a'} Q_t(x',a')\right] \\ &x' \sim p(\cdot|x,a) \end{aligned} \tag{17}$$

Here Q is an estimate of the optimal action-value function and β is the learning rate. The order of the visited states is unimportant, as long as all reachable states are updated infinitely often and the learning rate meets a standard condition used in stochastic approximation.

$$\sum_{t=0}^{\infty} \beta_t = \infty \qquad \sum_{t=0}^{\infty} \beta_t^2 < \infty \tag{18}$$

See [6] for details.

2.4 CVaR estimation

Before formulating a CVaR version of Q-learning, we must first talk about simply *estimating* CVaR, as it is not as straightforward as the estimation of expected value.
Let us remind ourselves of the primal definition of CVaR (5):

$$\text{CVaR}_\alpha(Z) = \max_s \left\{ \frac{1}{\alpha}\mathbb{E}\left[(Z-s)^-\right] + s \right\}$$

If we knew the exact $s^* = \text{VaR}_\alpha$, we could estimate the CVaR as a simple expectation of the $\frac{1}{\alpha}(Z-s^*)^- + s^*$ function. As we do not know this value in advance, a common approach is to first approximate VaR_α from data, then use this estimate to compute it's CVaR_α. This is usually done with a full data vector, requiring the whole data history to be saved in memory.
When dealing with reinforcement learning, we would like to store our current estimate as a scalar instead. This requires finding a recursive expression whose expectation is the CVaR value. Fortunately, similar methods have been thoroughly investigated in the stochastic approximation literature by [7].
The RM theorem has also been applied directly to CVaR estimation by [8], who used it to formulate a recursive importance sampling procedure useful for estimating CVaR of long-tailed distributions.
First let us describe the method for a one step estimation, meaning we sample values (or rewards in our case) r from some distribution and our goal is to estimate CVaR at a given confidence level α. The procedure requires

us to maintain two separate estimates V and C, being our VaR and CVaR estimates respectively.

$$V_{t+1} = V_t + \beta \left[1 - \frac{1}{\alpha} \mathbb{1}_{(V_t \geq r)} \right] \tag{19}$$

$$C_{t+1} = (1-\beta) C_t + \beta \left[V_t + \frac{1}{\alpha} (r - V_t)^- \right] \tag{20}$$

An observant reader may recognize a standard equation for quantile estimation in equation (19) (see e.g. [9] for more information on quantile estimation/regression) and equation (20) is also quite intuitive, representing the moving exponential average of the primal CVaR definition (5). The estimations are proven to converge, given the usual requirements on the learning rate (18) [8].

3 CVaR Q-learning

We now extend the previously established CVaR value iteration and combine it with the recursive CVaR estimation techniques to formulate a new algorithm we call CVaR Q-learning.

3.1 Temporal Difference update

We first define two separate values for each state, action and atom $V, C : \mathcal{X} \times \mathcal{A} \times \mathcal{Y} \to \mathbb{R}$ where $C(x,a,y)$ represents $\mathrm{CVaR}_y(Z(x,a))$ of the distribution, similar to the definition (10). $V(x,a,y)$ represents the one-step VaR_y estimate, or the estimate of the $y-$quantile of a distribution recovered from CVaR_y by Lemma 1.
A key to any TD algorithm is it's update rule. The CVaR TD update rule extends the improved VI procedure and we present the full rule in Algorithm 2.
Let us now go through the algorithm and compare the two versions. We first construct a new CVaR (line 3) by greedily selecting actions that yield the highest CVaR for each atom. This step is somewhat skipped in the VI process since we are not working with action-value functions. These values are then transformed to their underlying distributions (line 5) and used to generate 'samples' v from this distribution. Quotes are used here since we know the distributions and do not have to actually sample - instead we use the quantile values proportionally to their probabilities. Similar technique has been used to minimize the Wasserstein distance of a learned distribution by [10].
If the atoms aren't uniform, we perform basic importance sampling (line) to weight each 'sample' according to it's probability. Finally, these ***samples*** are used to update the respective VaR and CVaR estimates.

Algorithm 2 CVaR TD update

```
1:  input: $x, a, x', r$
2:  for each $y_i$ do
3:      $C(x', y_i) = \max_{a'} C(x', a', y_i)$
4:  end for
5:  todo: introduce notation to C, v
6:  for each $v$, $y_i$ do
7:      $\beta_i = \beta(y_i - y_{i-1})$
8:      $V(x,a,y_i) = V(x,a,y_i) + \beta_i \left[ 1 - \frac{1}{y_i} \mathbb{1}_{(V(x,a,y_i) \geq r + \gamma v)} \right]$
9:      $C(x,a,y_i) = (1-\beta_i) C(x,a,y_i) + \beta_i \left[ V(x,a,y_i) + \frac{1}{y_i} (r + \gamma v - V(x,a,y_i))^- \right]$
10: end for
```

4 Convergence

The main problem faced here is one of convergence. We would like to know if the proposed Q-learning algorithm converges.
Positive results: 1) The interpolated VI is a contraction, 2) the 1-step VaR-CVaR estimation converges w.p. 1(unclear if it only converges for continuous distributions)
Unclear: does the whole procedure converge?

References

[1] R Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[2] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[3] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. In *Advances in Neural Information Processing Systems*, pages 1522–1530, 2015.

[4] Georg Ch Pflug and Alois Pichler. Time-consistent decisions and temporal decomposition of coherent risk functionals. *Mathematics of Operations Research*, 41(2):682–699, 2016.

[5] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[6] Tommi Jaakkola, Michael I Jordan, and Satinder P Singh. Convergence of stochastic iterative dynamic programming algorithms. In *Advances in neural information processing systems*, pages 703–710, 1994.

[7] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[8] Olivier Bardou, Noufel Frikha, and Gilles Pages. Recursive computation of value-at-risk and conditional value-at-risk using mc and qmc. In *Monte Carlo and quasi-Monte Carlo methods 2008*, pages 193–208. Springer, 2009.

[9] Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001.

[10] Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. *arXiv preprint arXiv:1710.10044*, 2017.