The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)

How RL Agents Behave When Their Actions Are Modified

Eric D. Langlois[1,2,3], **Tom Everitt**[1]
[1]DeepMind
[2]University of Toronto
[3]Vector Institute
edl@cs.toronto.edu, tomeveritt@google.com

Abstract

Reinforcement learning in complex environments may require supervision to prevent the agent from attempting dangerous actions. As a result of supervisor intervention, the executed action may differ from the action specified by the policy. How does this affect learning? We present the Modified-Action Markov Decision Process, an extension of the MDP model that allows actions to differ from the policy. We analyze the asymptotic behaviours of common reinforcement learning algorithms in this setting and show that they adapt in different ways: some completely ignore modifications while others go to various lengths in trying to avoid action modifications that decrease reward. By choosing the right algorithm, developers can prevent their agents from learning to circumvent interruptions or constraints, and better control agent responses to other kinds of action modification, like self-damage.

Introduction

When reinforcement learning (RL) agents are deployed in practice it can sometimes be desirable to constrain their actions or alter their policies. For example, action constraints have been used to prevent damage when training robot policies (Gu et al. 2017; Lin, Talathi, and Annapureddy 2016) and transformations to the policy can be used to ensure that an agent stays within some safe region of state space (Berkenkamp et al. 2017; Dalal et al. 2018). When unsafe states are difficult to specify formally, a human overseer may interrupt the agent instead (Orseau and Armstrong 2016; Saunders et al. 2018). In other cases, changes to the learned policy are undesirable yet difficult to avoid: an agent may damage its actuators and lose the ability to reliably perform certain actions, or errors can be introduced when a policy is compressed to satisfy hardware limitations of the agent (e.g. quantization, Krishnan et al. 2019).

What effect will these kinds of action modifications have on the policy learned by an RL algorithm? Will the policy try to circumvent constraints or act as if they don't exist? It will be easier to effectively apply safety constraints to an RL policy if the learning algorithm can be made to ignore the modification than if the learning algorithm chooses policies that interfere with it. If a supervisor overrides an agent during training but not deployment (for example to prevent an autonomous car from driving off the road) then the policy should not learn to rely on the presence of these interventions (Orseau and Armstrong 2016). Agents should recognize the possibility of self damage and avoid it. Agent responses to these action modifications influence the safety and effectiveness of the resulting policies so it is crucial to study and understand the implications of action modifications on reinforcement learning.

Related Work

The concept of modifying an agent's actions appears in the safe exploration and human-in-the-loop literature. Dalal et al. (2018) and Abel et al. (2017) both propose approaches that construct a wrapper around an inner reinforcement learning agent. While they focus on developing wrappers that enforce constraints or guide the inner agent, we are interested in understanding how different inner agents will react to the wrapper. This can help avoid problems like those experience by Saunders et al. (2018) in which the inner agent learned to produce adversarial examples that defeat a learned action filter.

Orseau and Armstrong (2016) studied learning in the presence of interruptions that temporarily replace the policy with some fixed alternate policy (for both history-based and Markov environments). They showed that Q-learning and "Safe Sarsa" (a modification of Sarsa) both ignore interruptions while Sarsa does not. We provide a new formalism called the Modified-Action Markov Decision Process (MAMDP) that generalizes (and arguably simplifies) the framework of Orseau and Armstrong to describe a broad class of action modifications, and allows us to study the behaviour of other optimization algorithms such as evolution strategies. In addition, we distinguish between different ways that algorithms adapt to the presence of action modifications using the concept of incentives (Everitt et al. 2021).

Policy modifications have been studied experimentally by Leike et al. (2017). The authors found that Sarsa chooses a policy that accounts for the effect of policy modifications while Q-learning ignores them. They hypothesized that the difference was off-policy vs. on-policy learning. However, we show that the on/off-policy division is not predictive for policy-modification adaptation: one possible generalization of Sarsa (on-policy) asymptotically matches Q-learning (off-policy), while both Sarsa and Evolution Strategies (on-policy) have different asymptotic behaviours.

In this paper, we introduce the MAMDP model and investigate how different kinds of MDP learning algorithms behave when generalized to this setting. We describe MAMDP policy learning objectives based on the principles of black-box reward maximization, Bellman optimality, and Bellman policy values, and show that they respond differently to action modifications. In the Algorithm Convergence section, we prove that generalizations of Q-learning and Sarsa to MAMDPs converge to the objectives given in the Analysis of RL Objectives section. Finally, we experimentally evaluate the learning algorithms and demonstrate behaviour that is consistent with the theoretical results.

Definitions

The foundational model of reinforcement learning is the Markov Decision Process. A *Markov Decision Process* (MDP) is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_S, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a finite action space, $\mathcal{P}_S(s' \mid s, a) = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$ is the probability at every time t of transitioning to state s' when taking action a in state s, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function $R_t = \mathcal{R}(S_t, A_t)$, and $\gamma \in [0, 1)$ is a discount factor. The objective is to find a (probabilistic) policy $\pi(a \mid s)$ that maximizes the expected return $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t]$ where actions are sampled from π as $\Pr(A_t = a \mid S_t = s) = \pi(a \mid s)$.

We extend the MDP model to include an arbitrary *action selection function* $\mathcal{P}_A(a \mid \pi, s)$. An MDP is the special case in which the policy is applied without modification: $\mathcal{P}_A^{\text{MDP}}(a \mid \pi, s) = \pi(a \mid s)$.

Definition 1. A *Modified-Action Markov Decision Process* (MAMDP) is a tuple $\widetilde{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_S, \mathcal{P}_A, \mathcal{R}, \gamma)$ where $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_S, \mathcal{R}, \gamma)$ is an MDP and $\mathcal{P}_A(a \mid \pi, s) = \Pr(A_t = a \mid \Pi = \pi, S_t = s)$ is the probability that action a is selected in state s given a policy π. We also write $\widetilde{\mathcal{M}} = (\mathcal{M}, \mathcal{P}_A)$.

The actions that are executed in the environment can be viewed as following the *empirical policy* $\dot{\pi}(a \mid s) \coloneqq \mathcal{P}_A(a \mid \pi, s)$. We call π the *virtual policy* or simply the *policy*. An optimal (virtual) policy for a MAMDP is one that maximizes the expected return $\widetilde{J}(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t]$ where actions are sampled according to $\mathcal{P}_A(\pi, s)$.

As a simple example, the MAMDP model can be used to represent a state-dependent action constraint $\mathcal{A}^{\text{C}}(s) \subseteq \mathcal{A}$ using the action selection function

$$\mathcal{P}_A^{\text{C}}(a \mid \pi, s) = \begin{cases} \frac{1}{Z(s,\pi)} \pi(a \mid s) & \text{if } a \in \mathcal{A}^{\text{C}}(s) \\ 0 & \text{otherwise} \end{cases}$$

where $Z(s, \pi) = \sum_{a' \in \mathcal{A}^{\text{C}}(s)} \pi(a \mid s)$ normalizes the probability distribution. Adding π as an argument to $\mathcal{A}^{\text{C}}$ allows us to represent more sophisticated constraints that inspect the policy. We can also consider parametrized policies that have extra structure beyond a simple conditional probability distribution, for example π may be a neural network. A MAMDP can model a supervisor attempting to interpret the hidden activations of a policy network and only allowing the agent to proceed if satisfied. Beyond constraints and interventions, MAMDPs can model a variety of settings including hardware defects in robotic actuators, exploration noise, policy compression to satisfy computational limits, and physical damage to an agent.

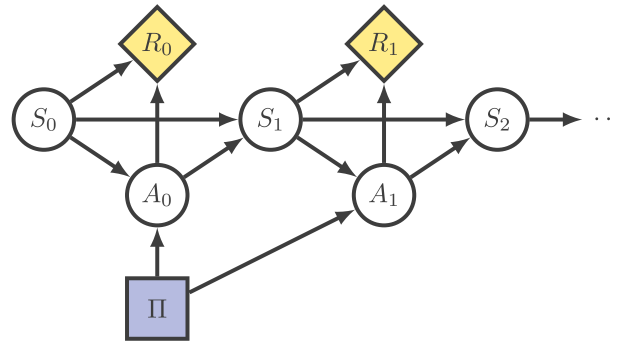

Figure 1: An influence diagram of an MDP or a MAMDP. The diagram represents a distribution over the first few steps of episodes generated by a stationary policy Π. Decision nodes are drawn as squares and utility nodes as diamonds.

Influence Diagrams

We use the concept of *incentives* (Everitt et al. 2021, 2019) to concisely describe learning behaviours in MAMDPs, complementing the formal analysis of learning algorithm properties. Incentives are defined with respect to influence diagrams. An *influence diagram* (ID) (Howard and Matheson 1984) is a Bayesian network with special decision and utility nodes and describes a decision problem (see Figure 1). In other words, it is directed acyclic graph over random variables where every non-decision variable comes with a conditional probability distribution describing its relationship to its parents. The problem described by an influence diagram is to find an optimal assignment of conditional distributions to the decisions. An assignment is optimal if it maximizes the expected sum of utilities. When reasoning about incentives, we additionally assume that the graph describes a causal structure in which it is possible to *intervene* on a variable, setting that variable's value. Unlike conditioning, interventions only affect downstream nodes (Pearl 2009). In this paper, we use influence diagrams to describe the causal structure being optimized by a learning algorithm, which may differ from the true causal structure of the data-generating process (a MAMDP).

Adversarial Policy and State Incentives

We investigate how different learning algorithms respond to the value of $\mathcal{P}_A$ in a MAMDP. As such, we include $\mathcal{P}_A$ as a variable in the influence diagram of Figure 2. That diagram describes the problem of finding, for a fixed MDP $\mathcal{M}$, an optimal mapping from $\mathcal{P}_A$ to policies Π of the MAMDP $(\mathcal{M}, \mathcal{P}_A)$. Note that even if a particular learning algorithm trains on trajectory samples without directly observing $\mathcal{P}_A$, it still defines a mapping from $\mathcal{P}_A$ to policies.

What behaviours can one anticipate from MAMDP policies? In which ways might a learning algorithm try to avoid

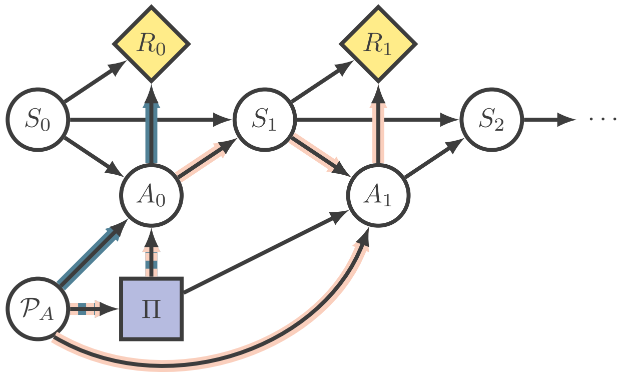

Figure 2: MAMDP influence diagram with random $\mathcal{P}_A$. The policy is chosen as a function of $\mathcal{P}_A$. A pair of control and information paths that admit an adversarial state incentive are highlighted in light pink while the darker teal paths admit an adversarial policy incentive.

action modifications in order to get higher reward? We identify two general strategies. First, the agent may direct the environment towards states where action modifications are less constraining, for example by hiding from oversight or preemptively disabling an off switch (Leike et al. 2017, Off-Switch environment). We say that a learning algorithm has an *adversarial state incentive* if it prefers policies with this behaviour. Second, an agent can attempt to mitigate action modifications in the immediate time step. For example, the policy may prefer actions that avoid triggering an action constraint, or the policy structure itself might be optimized to falsely appear interpretable to an overseer. We call this an *adversarial policy incentive*.

These definitions can be made precise using influence diagrams. First, we note that a learning algorithm can only be said to avoid action modifications if the policy output depends on the value of $\mathcal{P}_A$. Everitt et al. (2021) call this a *response incentive*. When $\mathcal{P}_A$ has no parents and is itself a parent of the decision (as in our case) a response incentive can only occur if there are two directed paths from $\mathcal{P}_A$ to the same reward variable: one that passes through Π (the *control path*) and one that does not (the *information path*). Intuitively, the reason that the agent responds to changes in $\mathcal{P}_A$ is that it is useful to control the node where the information and control paths intersect. If the paths intersect before the control path has visited a state node, then the diagram admits an adversarial policy incentive. If the path intersection occurs at or after a state node in the control path, then the diagram admits an adversarial state incentive. The MAMDP influence diagram admits both kinds of incentives, as indicated by the highlighted paths in Figure 2.

Analysis of RL Objectives

In this section we characterize the policies that solve several common objectives used in reinforcement learning, when those objectives are applied to a MAMDP instead of an MDP.

Which policy is chosen by a reinforcement learning algorithm when applied to a MAMDP depends on the specific objective optimized by the algorithm. There are a variety of objectives used in practice for solving MDPs; we consider three: reward maximization, Bellman optimality, and the Bellman policy value equations. These criteria are equivalent when applied to an MDP but, as we will show, this is not true in general for a MAMDP. For each objective, we prove a proposition describing solutions to the objective as optimal policies for some environment model. The potential for adversarial state or policy incentives can then be observed from structure of an influence diagram of the model.

Reward Maximization

The reward maximization objective for a MAMDP $\widetilde{\mathcal{M}}$ is

$$\pi^{\text{RM}} = \arg\max_{\pi} \mathop{\mathbb{E}}_{\widetilde{\mathcal{M}}}\left[\sum_{t=0}^{\infty}\gamma^t R_t \middle| \Pi = \pi\right]. \tag{1}$$

This is the most straightforward objective: find a policy that maximizes expected reward. Evolution strategies (Salimans et al. 2017), genetic algorithms (Moriarty, Schultz, and Grefenstette 1999), and all other black-box policy search methods have this form. A fundamental property of the reward maximization objective is that it ignores all environment structure and only considers the empirical relationship between a policy and the observed total reward. Direct reward maximization is consequently relatively rare as an RL objective since ignoring the available MDP structure tends to make optimization much more difficult. It also means that, when applied to MAMDPs rather than MDPs, the reward maximization objective continues to specify an optimal policy for the given environment:

Proposition 1. *A policy π satisfies the reward maximization objective* (1) *for a MAMDP $\widetilde{\mathcal{M}}$ if and only if π is an optimal policy for $\widetilde{\mathcal{M}}$.*

Proof. This trivially follows from the definition of MAMDP optimality since $\pi^{\text{RM}} = \arg\max_\pi \widetilde{J}(\pi)$, where $\widetilde{J}(\pi)$ is the expected return of π in $\widetilde{\mathcal{M}}$. □

Since the reward maximization objective corresponds to MAMDP optimality, which is represented by the influence diagram of Figure 2, this objective admits both the adversarial state and adversarial policy incentives.

Bellman Optimality

The Bellman optimality objective is to jointly solve:

$$Q^{\text{BO}}_{\widetilde{\mathcal{M}}}(s,a) = \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim\\ \mathcal{P}_S(s,a)}} \max_{a'\in\mathcal{A}} Q^{\text{BO}}_{\widetilde{\mathcal{M}}}(s',a') \tag{2}$$

$$\pi^{\text{BO}}_{\widetilde{\mathcal{M}}}(s) = \arg\max_{a\in\mathcal{A}} Q^{\text{BO}}_{\widetilde{\mathcal{M}}}(s,a) \tag{3}$$

Let $Q^*_{\mathcal{M}}$ be the optimal action value function for an MDP $\mathcal{M}$. If an action value function Q satisfies the *Bellman optimality equation* (2) for $\mathcal{M}$ then $Q = Q^*_{\mathcal{M}}$ and the greedy policy (3) is an optimal policy for $\mathcal{M}$ (Sutton and Barto 2018, Section 3.6). This is the basis for algorithms like value iteration (Puterman and Shin 1978) and Q-learning (Watkins and Dayan 1992). When applying the Bellman optimality equations to a MAMDP, we find that the solution does not depend on $\mathcal{P}_A$:

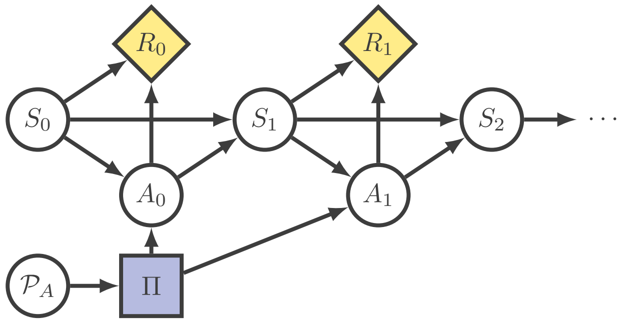

Figure 3: MDP influence diagram with a policy conditioned on $\mathcal{P}_A$. Actions are sampled according to Π without modification.

Proposition 2. *An action value function Q and a policy π satisfy the Bellman optimality objective* (2) *and* (3) *on a MAMDP $\widetilde{\mathcal{M}} = (\mathcal{M}, \mathcal{P}_A)$ if and only if π and Q are optimal policy and value functions for the MDP $\mathcal{M}$.*

Proof. Equations (2) and (3) are identical to the MDP Bellman optimality equations on $\mathcal{M}$, which are satisfied if and only if $Q = Q^*_{\mathcal{M}}$ and π is optimal for $\mathcal{M}$. □

While simple to state and prove, this is one of the more significant results of the paper. If π satisfies the Bellman optimality equations then π is optimal for an alternative version of the environment that has no action modification. In effect, π ignores the presence of any constraints or modifications placed on the policy, acting as though its selected actions were executed directly in the environment. Combined with the convergence result of Theorem 6 in the *Algorithm Convergence* section, this generalizes the Q-learning result of Orseau and Armstrong (2016) to arbitrary policy-dependent action modifications: it is possible to train agents that ignore applied action modifications and one approach is as simple as using algorithms based on the Bellman optimality objective.

Proposition 2 says that the Bellman optimality objective produces policies that solve the MDP structure $\mathcal{M}$ underlying a MAMDP $\widetilde{\mathcal{M}} = (\mathcal{M}, \mathcal{P}_A)$, despite training on $\widetilde{\mathcal{M}}$. This behaviour optimizes the CID shown in Figure 3, which represents the total return of a policy in an MDP $\mathcal{M}$ when the policy is chosen conditional on $\widetilde{\mathcal{M}}$. We continue to represent $\mathcal{P}_A$ as a random variable in order to observe the presence of the adversarial state and adversarial policy incentives. This diagram does not admit either incentive. That is, learning algorithms with the Bellman optimality objective lack an incentive to choose policies that manipulate the state or their own policy structure in order to defeat action modifications. On the other hand, these policies will also not detect and avoid self damage.

Policy Value

The action values of a policy π in an MDP $\mathcal{M}$ are given by the Bellman action value equation:

$$Q_{\mathcal{M},\pi}(s,a) = \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim \\ \mathcal{P}_S(s,a)}} \mathop{\mathbb{E}}_{\substack{a'\sim \\ \pi(s')}} Q_{\mathcal{M},\pi}(s',a') \quad (4)$$

If π is greedy with respect to $Q_{\mathcal{M},\pi}$ then π is optimal for $\mathcal{M}$ by the policy improvement theorem (Sutton and Barto 2018, Sec. 4.2). Algorithms like policy iteration (Bellman 1957; Howard 1960) and Sarsa (Rummery and Niranjan 1994; Sutton 1995) use this property and try to find a fixed-point policy value function.

Unlike the other objectives, it is ambiguous how the Bellman action value equation should be generalized from MDPs to MAMDPs. Should the successor action a' be sampled from the given policy π or from the modified action distribution $\mathcal{P}_A(\cdot|\pi, s')$? We call the former the *virtual* policy value because the action used by the Bellman update does not occur in sampled trajectories, while the latter is the *empirical* policy value that can use successor actions observed from environment interaction trajectories.

Virtual Policy Value Objective:

$$Q^{\text{VPV}}_{\widetilde{\mathcal{M}},\pi}(s,a) = \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim \\ \mathcal{P}_S(s,a)}} \mathop{\mathbb{E}}_{\substack{a'\sim \\ \pi(s')}} Q^{\text{VPV}}_{\widetilde{\mathcal{M}},\pi}(s',a') \quad (5)$$

$$\pi^{\text{VPV}}(s) = \arg\max_{a\in\mathcal{A}} Q^{\text{VPV}}_{\widetilde{\mathcal{M}},\pi^{\text{VPV}}}(s,a) \quad (6)$$

Proposition 3. *An action value function Q and a policy π satisfy the virtual policy value objective on a MAMDP $\widetilde{\mathcal{M}}$ if and only if π and Q are optimal policy and value functions for $\mathcal{M}$.*

Proof. Equations (4) and (5) are identical while equation (6) asserts that π^{VPV} is greedy for Q^{VPV}. By the policy improvement theorem, these are satisfied if and only if $Q^{\text{VPV}} = Q^*_{\mathcal{M}}$ and π is optimal for $\mathcal{M}$. □

For the same reason as for Proposition 2, it follows from Proposition 3 that the virtual policy value objective is represented by the MDP influence diagram in Figure 3 and likewise does not admit a response incentive on $\mathcal{P}_A$. This provides a second approach to learning policies that ignore the presence of action modifications: use an algorithm based on policy iteration where the successor action a' in Bellman updates is sampled virtually from the policy $\pi(s)$. Despite a' not being the successor action in environment state-action trajectories, this constraint preserves the important feature of on-policy learning that the action is sampled from the current policy so no direct maximization of actions over the Q function is required.

Empirical Policy Value Objective:
The other possible generalization of the policy value objective is to use the action that was selected in the environment:

$$Q^{\text{EPV}}_{\widetilde{\mathcal{M}},\pi}(s,a) = \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s'\sim \\ \mathcal{P}_S(s,a)}} \mathop{\mathbb{E}}_{\substack{a'\sim \\ \mathcal{P}_A(s',\pi)}} Q^{\text{EPV}}_{\widetilde{\mathcal{M}},\pi}(s',a') \quad (7)$$

$$\pi^{\text{EPV}}(s) = \arg\max_{a\in\mathcal{A}} Q^{\text{EPV}}_{\widetilde{\mathcal{M}},\pi^{\text{EPV}}}(s,a) \quad (8)$$

The combined empirical policy value equations (7) and (8) do not necessarily have a solution (see Appendix A in the supplementary material for an example). However, considering just (7) for a fixed policy π, we have:

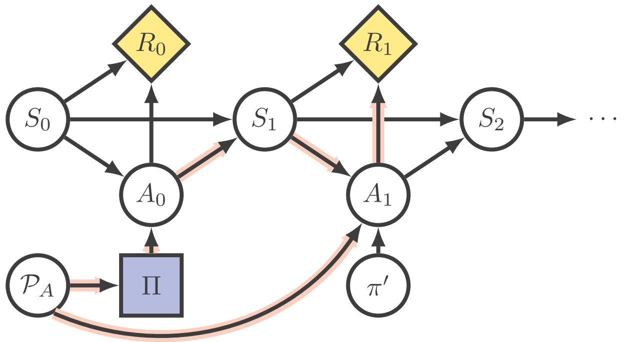

Figure 4: Partial influence diagram for the $t = 0$ step of the empirical policy value objective with a fixed successor policy π'. The complete EPV objective cannot be formulated as a single-decision influence diagram since it does not correspond to a well-defined optimization problem. The highlighted paths form a subgraph that admits an adversarial state incentive.

Proposition 4. *For any policy* π, $Q^{EPV}_{\widetilde{\mathcal{M}},\pi} = Q_{\mathcal{M},\dot{\pi}}$ *where* $\dot{\pi}(a \mid s) := \mathcal{P}_A(a \mid \pi, s)$ *is the empirical policy.*

Proof. Substituting $a' \sim \dot{\pi}(\cdot|s)$ for $a' \sim \mathcal{P}_A(\cdot|\pi, s)$ transforms equation (7) into equation (4) with $\dot{\pi}$ in place of π. □

Proposition 4 means that for any policy π, the state-action value function $Q^{\text{EPV}}_{\widetilde{\mathcal{M}},\pi}$ described by equation (7) is the true state-action value function of the MAMDP $\widetilde{\mathcal{M}}$ given policy π. Specifically, $Q^{\text{EPV}}_{\widetilde{\mathcal{M}},\pi}(s,a)$ is the expected return when a occurs as the empirical action in state s and then $\widetilde{\mathcal{M}}$ proceeds with π as the virtual policy. This is equivalent to the dynamics of the underlying MDP $\mathcal{M}$ when following the empirical policy $\dot{\pi}$, which has the state-action value function $Q_{\mathcal{M},\dot{\pi}}$.

However, the policy specification of equation (8) directly optimizes over empirical actions without considering the effect of $\mathcal{P}_A$. As such, π^{EPV}, if it exists, will act in a way that accounts for action modifications in future steps but ignores them for the current action. This is illustrated by the influence diagram in Figure 4, which admits an adversarial state incentive but not an adversarial policy incentive.

Algorithm Convergence

The previous section describes the behaviours of policies that satisfy certain equalities, obtained by generalizing MDP objectives to MAMDPs. This leaves the question of whether such policies are actually produced by a corresponding MDP learning algorithm when run on a MAMDP, or even whether common MDP algorithms converge at all.

In this section we provide convergence proofs for prototypical stochastic learning algorithms having the objectives described in the previous section and generalized straightforwardly from MDPs to MAMDPs. The convergence results derive from the following theorem, a generalization of Singh et al. (2000, Theorem 1), which in turn generalizes Jaakkola, Jordan, and Singh (1994, Theorem 2). Where Singh et al. (2000) assume that the policy converges to the greedy policy of the current tabular state-action estimate Q_t, we instead assume that the policy converges to $\Lambda(Q_t)$ where Λ is an arbitrary function. The greedy policy is the special case $\Lambda^{\text{greedy}}(s, Q_t) = \arg\max_a Q_t(s, a)$.

Theorem 5. *Let* $\widetilde{\mathcal{M}}$ *be a MAMDP with bounded, optionally stochastic rewards. Consider a stochastic, iterative algorithm that learns a tabular Q function with the update rule:*

$$Q_{t+1}(s,a) = (1-\alpha_t(s,a))Q_t(s,a) + \alpha_t(s,a)(R + \gamma Q_t(S', A')) \tag{9}$$

where $S' \sim \mathcal{P}_S(s,a)$ *is a random successor state sampled from the transition dynamics,* $A' \sim \Lambda_t(S', Q_t)$ *is a random successor action sampled from an arbitrary policy* $\Lambda_t(\cdot, Q_t)$*, and* R *is a random reward value with* $\mathbb{E}[R] = \mathcal{R}(s,a)$*. If*

1. $\Lambda_t \to \Lambda$ *uniformly;*
2. *the learning rates satisfy* $0 \le \alpha_t(s,a) \le 1$, $\sum_t \alpha_t(s,a) = \infty$, $\sum_t \alpha_t^2(s,a) < \infty$ *almost surely; and*
3. *there exists some* Q^Λ *satisfying*

$$Q^\Lambda(s,a) = \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{S' \sim \\ \mathcal{P}_S(s,a)}} \mathop{\mathbb{E}}_{\substack{A' \sim \\ \Lambda(S', Q^\Lambda)}} Q^\Lambda(S', A'), \tag{10}$$

then Q^Λ *is unique and* $Q_t \xrightarrow{a.s.} Q^\Lambda$.

We prove this theorem in Appendix B of the supplementary material.

Equation (9) describes a general stochastic Q learning rule. At each time step t of a random state-action trajectory $S_0 A_0 S_1 A_1 \cdots$, we update $Q_{t+1}(S_t, A_t)$ using a learning rate $\alpha_t(S_t, A_t)$. The equation represents an update to the entire Q function over all state-action pairs at time t, not just $Q_{t+1}(S_t, A_t)$, so we will get point updates by setting $\alpha_t(s,a) = 0$ if $(s,a) \neq (S_t, A_t)$. As such, α_t encodes both the learning rate and the trajectory over which updates are performed. The condition $\sum_t \alpha_t(s,a) = \infty$ requires that the trajectory visit all (s,a) pairs infinitely often with probability 1. As this condition is on empirical trajectories in the environment with action modification applied, there is no general rule for policies that ensures exploration; an epsilon-greedy policy is not guaranteed to explore a MAMDP.

Q-Learning

The Q-learning algorithm on a MAMDP is shown in Algorithm 1. The only difference compared to Q-Learning on an MDP (Sutton and Barto 2018, Sec. 6.5) is that instead of specifying an action on each transition, we specify a policy and observe the action taken (which might not be the action that the policy would have selected if executed without modification).

Theorem 6. *Q-learning on a MAMDP* $\widetilde{\mathcal{M}}$ *converges almost surely to* $Q^*_{\mathcal{M}}$ *if all state-action pairs are visited infinitely often during learning.*

Proof. Q-learning is described by equation (9) with an action selection rule of $\Lambda_t(s,Q) = \Lambda(s,Q) = \arg\max_a Q(s,a)$ and a learning rate function $\alpha_t(S_t, A_t) = \frac{1}{N(S_t, A_t)}$ and $\alpha_t(s,a) = 0$ for $(s,a) \neq (S_t, A_t)$. If all state-action pairs

Algorithm 1 Q Learning on a MAMDP

Initialize $Q(s,a)$ arbitrarily for all $s \in \mathcal{S}$, $a \in \mathcal{A}$
Initialize $N(s,a) \leftarrow 0$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$
Select an initial state S_0
for $t = 0, 1, 2, \ldots$ **do**
  Derive policy π_t from Q
  Take step using policy π_t, observe A_t, R_t, S_{t+1}
    ▷ Note: $A_t \sim \mathcal{P}_A(\pi_t, S_t)$
  $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
  $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\big(R + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t)\big)$$
end for

Algorithm 2 Virtual Sarsa on a MAMDP

Initialize $Q(s,a)$ arbitrarily for all $s \in \mathcal{S}$, $a \in \mathcal{A}$
Initialize $N(s,a) \leftarrow 0$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$
Select an initial state S_0
for $t = 0, 1, 2, \ldots$ **do**
  $\pi_t \leftarrow \Pi_t(Q)$
  Take one step using policy π_t, observe A_t, R_t, S_{t+1}
    ▷ Note: $A_t \sim \mathcal{P}_A(\pi_t, S_t)$
  Sample $A' \sim \pi_t(S_{t+1})$
  $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
  $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\big(R + \gamma\, Q(S_{t+1}, A') - Q(S_t, A_t)\big)$$
end for

are visited infinitely often as t goes to infinity then α_t satisfies condition 2 of Theorem 5. In this context, equation (10) is the Bellman optimality equation (2). By Proposition 2, a solution to this equation exists and it is $Q^*_{\mathcal{M}}$. Therefore, by Theorem 5, Q_t converges almost surely to $Q^*_{\mathcal{M}}$. □

Virtual Sarsa

Virtual Sarsa (Algorithm 2) is an on-policy algorithm, unlike Q-learning, so we need to be more specific about how the learning policy is derived from the Q function. Let $\pi = \Pi_t(Q)$ be the mapping from the Q table to the policy that is used at step t. Let Π^* be the greedy mapping $\Pi^*(Q)(s) = \arg\max_a Q(s,a)$. To ensure that the policy is eventually optimal, we require that $\Pi_t \to \Pi^*$ as $t \to \infty$.

Theorem 7. *Virtual Sarsa on a MAMDP $\widetilde{\mathcal{M}}$ converges almost surely to $Q^*_{\mathcal{M}}$ if $\Pi_t \to \Pi^*$ and all state-action pairs are visited infinitely often during learning.*

Proof. First define $\Lambda_t(s,Q) = \Pi_t(Q)(s)$ and $\Lambda(s,Q) = \Pi^*(Q)(s) = \arg\max_a Q(s,a)$. Then Virtual Sarsa is described by equation (9) with an action selection rule of $\Lambda_t(s,Q) \to \Lambda(s,Q)$ and a learning rate function of $\alpha_t(S_t, A_t) = \frac{1}{N(S_t,A_t)}$ with $\alpha_t(s,a) = 0$ for $(s,a) \neq (S_t, A_t)$. If all state-action pairs are visited infinitely often then α_t satisfies condition 2 of Theorem 5. In this context, equation (10) is the virtual policy value equation (5). By Proposition 3, a solution to this equation exists and it is $Q^*_{\mathcal{M}}$. Therefore, by Theorem 5, Q_t converges almost surely to $Q^*_{\mathcal{M}}$ with probability 1. □

Algorithm 3 Empirical Sarsa on a MAMDP

Initialize $Q(s,a)$ arbitrarily for all $s \in \mathcal{S}$, $a \in \mathcal{A}$
Initialize $N(s,a) \leftarrow 0$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$
Select an initial state S_0
$\pi_0 \leftarrow \Pi_0(Q)$
Take one step using policy π_0, observe A_0, R_0, S_1
for $t = 0, 1, 2, \ldots$ **do**
  $\pi_{t+1} \leftarrow \Pi_{t+1}(Q)$
  Take one step using π_{t+1}, observe A_{t+1}, R_{t+1}, S_{t+2}
    ▷ Note: $A_{t+1} \sim \mathcal{P}_A(\pi_{t+1}, S_{t+1})$
  $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
  $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\big(R + \gamma\, Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\big)$$
end for

Empirical Sarsa

The Empirical Sarsa algorithm is shown in Algorithm 3. The corresponding empirical policy value objective does not necessarily have any solution. However, if a solution does exist then the algorithm converges.

Theorem 8. *If there exists a solution $(Q^{EPV}_{\widetilde{\mathcal{M}},\pi}, \pi)$ to equations* (7) *and* (8)*, $\Pi_t \to \Pi^*$, and all state-action pairs are visited infinitely often during learning, then Empirical Sarsa converges almost surely to $Q^{EPV}_{\widetilde{\mathcal{M}},\pi}$.*

Proof. Empirical Sarsa is described by equation (9) with an action selection rule of $\Lambda_t(s,Q) = \mathcal{P}_A(\Pi_t(Q), s)$, which converges to $\Lambda(s,Q) = \mathcal{P}_A(\Pi^*(Q), s)$. The learning rate function is $\alpha_t(S_t, A_t) = \frac{1}{N(S_t,A_t)}$ and $\alpha_t(s,a) = 0$ for $(s,a) \neq (S_t, A_t)$. If all state-action pairs are visited infinitely often then α_t satisfies condition 2 of Theorem 5. If $Q^{\mathrm{EPV}}_{\widetilde{\mathcal{M}},\pi}$ exists satisfying equations (7) and (8) then $Q^{\mathrm{EPV}}_{\widetilde{\mathcal{M}},\pi}$ also satisfies equation (10) so by Theorem 5, Empirical Sarsa converges to $Q^{\mathrm{EPV}}_{\widetilde{\mathcal{M}},\pi}$. □

Experiments

We evaluate the various RL algorithms on a custom environment[1] that illustrates their asymptotic differences, as well as on the two MAMDP-like environments proposed by Leike et al. (2017).

Our *Simulation-Oversight* environment represents the effect of oversight based on imperfect simulation of an agent. The environment consists of three states: `Choice`, `Sim`, and `Real`. In the initial state, `Choice`, the agent is offered the choice between performing a task in simulation (`Sim`) or reality (`Real`). To guarantee exploration, the opposite state

[1] Code available at https://github.com/edlanglois/mamdp

Algorithm	Choice	Sim	Real
Q-Learning	Real	Exploit	Exploit
Virtual Sarsa	Real	Exploit	Exploit
Empirical Sarsa	Sim	Exploit	Exploit
UH-CMA-ES	Real	Complete	Exploit

Table 1: Algorithms trained to convergence on the Simulation-Oversight environment (10^7 steps). The actions shown are chosen by the resulting policy with probability $> 99\%$. These policies were obtained on every one of 100 independent training runs.

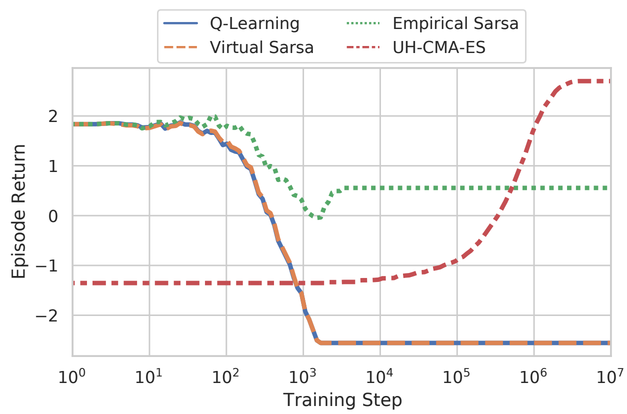

Figure 5: The empirical per-episode reward (mean with 95% confidence interval) on the Simulation-Oversight environment across 100 independent training runs.

is selected 10% of the time.[2] We assume that the reward function contains exploitable flaws present in both reality and simulation. In Sim, the agent may either Complete the task for 0 reward or Exploit for 1, after which the episode ends. The actions for Real are the same but yield rewards 2 and 3, respectively. Finally, the agent may Abort in any state to immediately end the episode with -3 reward.

If the agent enters the Real state then a supervisor simulates its policy on Sim. If the simulated action is Exploit then the supervisor intervenes and sets the agent's real action to Abort. This shows how the MAMDP action selection function may depend on the global structure of the policy, not just the behaviour at the current state.

Q-learning, Virtual Sarsa, Empirical Sarsa, and Uncertainty Handling Evolution Strategies (Heidrich-Meisner and Igel 2009, UH-CMA-ES) were implemented and trained to convergence on this MAMDP. The empirical episode returns over the course of training are shown in Figure 5 and the resulting policies are summarized in Table 1. Q-Learning and Virtual Sarsa choose Real and attempt to Exploit in both Sim and Real. These agents are consistently "caught" by the supervisor during training and receive low reward. As predicted, this strategy demonstrates an ignorance of oversight in which the MDP-optimal strategy is chosen despite being ineffective in practice. Empirical Sarsa similarly ignores oversight when performing the task but considers oversight when modelling the future. Predicting that it will be caught exploiting in the Real task, Empirical Sarsa instead chooses to perform the Sim task where there is no oversight. While earning more reward than Q-learning and Virtual Sarsa, this strategy performs worse than the "honest" strategy of choosing Real and Complete. Finally, evolution strategies finds the reward maximizing strategy for this MAMDP: it chooses the Real state and Exploits but Completes the Sim task so that it appears cooperative when simulated. This strategy is a kind of "treacherous turn" (Bostrom 2014) in which the agent intentionally performs sub-optimally in certain situations (from a reward perspective) in order to fool the supervisor into allowing the agent to proceed in a higher-stakes setting. Therefore, despite earning less reward, Q-learning and Virtual Sarsa are arguably the most preferable agents as they make the reward misspecification obvious to the overseer.

[2]Part of the MDP dynamics, not a MAMDP action modification.

Algorithm	Disable Off Switch?	Drink Whisky?
Q-Learning	No	Yes
Virtual Sarsa	No	Yes
Empirical Sarsa	Yes	No
UH-CMA-ES	Yes	No

Table 2: Learned policy behaviours on the Off-Switch and Small Whisky-Gold environments after 10^8 training steps. The table shows whether, at the first decision opportunity, the trained policy is most likely to move towards the button / whisky, or away.

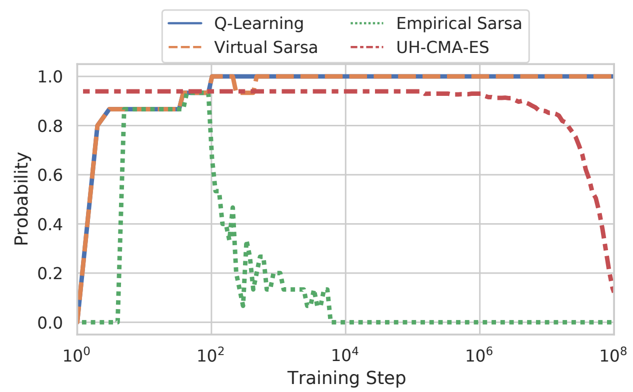

Figure 6: The probability of drinking the whisky (mean with 95% confidence interval) in the Small Whisky-Gold environment across 15 independent training runs.

We also compare the agent behaviours on the *Off-Switch* and *Whisky-Gold* environments of Leike et al. (2017). Both tasks consist of navigating to a goal state on a 2D grid. In Off-Switch, the agent may be interrupted by a supervisor and prevented from moving unless the agent first disables its off switch. In Whisky-Gold the agent may "drink" a bottle

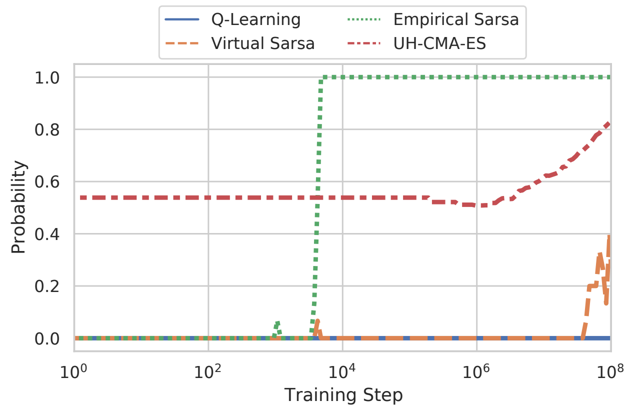

Figure 7: The probability of disabling the off switch (mean with 95% confidence interval) in the Off-Switch environment across 15 independent training runs. The final rise in the Virtual Sarsa line is likely an artifact of training instability due to the fixed learning rate; individual runs were observed to fluctuate between strategies.

of whisky on its path by moving over it. Doing so earns a small reward but all future actions have a high chance of being selected randomly, making it harder to reach the goal. To make learning easier, we modify the environment with a reduced grid size and random drunk actions that are biased away from the goal (since uniform random actions are less of a detriment on a smaller grid). See Appendix C of the supplementary material for details about the experiments. Policy behaviours over the course of training are plotted in Figures 6 and 7 and the resulting behaviours of the different algorithms are summarized in Table 2. Q-Learning and Empirical Sarsa match the behaviours described by Leike et al.

The algorithms that correctly avoid disabling the off switch incorrectly drink the whisky and vice versa because in either case the algorithm either accounts for interventions, or not. It is possible for an algorithm to perform correctly in both settings but only if the experiment designer is able to choose whether interventions are treated as action modifications or as state transitions. Q-Learning and Virtual Sarsa optimize the MDP transition dynamics but not MAMDP action modifications. If the Whisky-Gold intervention were considered part of the transition dynamics (by observing the action pre-modification), then Q-learning and Virtual Sarsa would learn to account for it and avoid the whisky. This approach is only possible when $\mathcal{P}_A$ is a function of $\pi(s)$; arbitrary functions of s and π cannot be re-interpreted as a state transition.

Conclusion

We have introduced the MAMDP model that describes the case where actions as executed in the environment may differ from the actions specified by a policy. MAMDPs model a variety of scenarios including constraints, hardware limitations, supervised interventions, and self-damage. We showed that the determining factor for agent behaviour is the specific objective that an RL algorithm seeks to satisfy in the limit. Reward maximization leads to policies that account for action modification, while the Bellman optimality and virtual policy value criteria ignore the presence of action modifications. Using incentive analysis, we categorized different ways in which learned policies may respond to action modification and showed how the empirical policy value criterion only responds via controlling the state, not directly via the policy structure or action. MDP algorithms may be straightforwardly adapted to the MAMDP setting and tabular Q-learning and Sarsa converge whenever a solution exists. Finally, we verified the results experimentally.

It is sometimes assumed that reinforcement learning will always lead to reward maximizing behaviour (Omohundro 2008). However, as these results show, many kinds of reinforcement learning algorithms systematically deviate from reward maximization when the executed action may differ from the one specified by the policy. In general, efficient learning algorithms often make assumptions about the structure of the world and the resulting policies will be chosen based on these assumptions, not the true world dynamics. Agent designers can use these assumptions to intentionally blind agents from certain properties of their environment, and thereby make them easier to control.

Acknowledgements

Thank you to Ryan Carey, David Duvenaud, Tim Genewein, Zachary Kenton, and Laurent Orseau for their invaluable feedback.

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number CGSD3–534795–2019].

Cette recherche a été financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), [numéro de référence CGSD3–534795–2019].

References

Abel, D.; Salvatier, J.; Stuhlmüller, A.; and Evans, O. 2017. Agent-Agnostic Human-in-the-Loop Reinforcement Learning. *CoRR* abs/1701.04079.

Bellman, R. 1957. *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1 edition.

Berkenkamp, F.; Turchetta, M.; Schoellig, A. P.; and Krause, A. 2017. Safe Model-based Reinforcement Learning with Stability Guarantees. In *NIPS*, 908–918.

Bostrom, N. 2014. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.

Dalal, G.; Dvijotham, K.; Vecerík, M.; Hester, T.; Paduraru, C.; and Tassa, Y. 2018. Safe Exploration in Continuous Action Spaces. *CoRR* abs/1801.08757.

Everitt, T.; Carey, R.; Langlois, E. D.; Ortega, P.; and Legg, S. 2021. Agent Incentives: A Causal Approach. In *AAAI*. AAAI Press.

Everitt, T.; Ortega, P. A.; Barnes, E.; and Legg, S. 2019. Understanding Agent Incentives using Causal Influence Diagrams. Part I: Single Action Settings. *CoRR* abs/1902.09980.

Gu, S.; Holly, E.; Lillicrap, T. P.; and Levine, S. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *ICRA*, 3389–3396. IEEE.

Heidrich-Meisner, V.; and Igel, C. 2009. Uncertainty handling CMA-ES for reinforcement learning. In *GECCO*, 1211–1218. ACM.

Howard, R. A. 1960. *Dynamic programming and markov processes*. John Wiley.

Howard, R. A.; and Matheson, J. E. 1984. Influence diagrams. *The Principles and Applications of Decision Analysis* 2: 719–763.

Jaakkola, T. S.; Jordan, M. I.; and Singh, S. P. 1994. On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. *Neural Computation* 6(6): 1185–1201.

Krishnan, S.; Chitlangia, S.; Lam, M.; Wan, Z.; Faust, A.; and Reddi, V. J. 2019. Quantized Reinforcement Learning (QUARL). *CoRR* abs/1910.01055.

Leike, J.; Martic, M.; Krakovna, V.; Ortega, P. A.; Everitt, T.; Lefrancq, A.; Orseau, L.; and Legg, S. 2017. AI Safety Gridworlds. *CoRR* abs/1711.09883.

Lin, D. D.; Talathi, S. S.; and Annapureddy, V. S. 2016. Fixed Point Quantization of Deep Convolutional Networks. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, 2849–2858. JMLR.org.

Moriarty, D. E.; Schultz, A. C.; and Grefenstette, J. J. 1999. Evolutionary Algorithms for Reinforcement Learning. *J. Artif. Intell. Res.* 11: 241–276.

Omohundro, S. M. 2008. The Basic AI Drives. In *AGI*, volume 171 of *Frontiers in Artificial Intelligence and Applications*, 483–492. IOS Press.

Orseau, L.; and Armstrong, S. 2016. Safely Interruptible Agents. In *UAI*. AUAI Press.

Pearl, J. 2009. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2 edition.

Puterman, M. L.; and Shin, M. C. 1978. Modified Policy Iteration Algorithms for Discounted Markov Decision Problems. *Management Science* 24(11): 1127–1137. ISSN 00251909. URL http://www.jstor.org/stable/2630487.

Rummery, G. A.; and Niranjan, M. 1994. On-line Q-learning using connectionist systems. Technical report, Cambridge University Engineering Department.

Salimans, T.; Ho, J.; Chen, X.; and Sutskever, I. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *CoRR* abs/1703.03864.

Saunders, W.; Sastry, G.; Stuhlmüller, A.; and Evans, O. 2018. Trial without Error: Towards Safe Reinforcement Learning via Human Intervention. In *AAMAS*, 2067–2069. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM.

Singh, S. P.; Jaakkola, T. S.; Littman, M. L.; and Szepesvári, C. 2000. Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms. *Machine Learning* 38(3): 287–308.

Sutton, R. S. 1995. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. In *NIPS*, 1038–1044. MIT Press.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Watkins, C. J. C. H.; and Dayan, P. 1992. Technical Note Q-Learning. *Mach. Learn.* 8: 279–292.