Robust temporal difference learning for critical domains

Richard Klima
University of Liverpool, UK
richard.klima@liverpool.ac.uk

Daan Bloembergen
Centrum Wiskunde & Informatica, NL
d.bloembergen@cwi.nl

Michael Kaisers
Centrum Wiskunde & Informatica, NL
m.kaisers@cwi.nl

Karl Tuyls
DeepMind & University of Liverpool, UK
karltuyls@google.com

September 13, 2021

Abstract

We present a new Q-function operator for temporal difference (TD) learning methods that explicitly encodes robustness against significant rare events (SRE) in critical domains. The operator, which we call the κ-operator, allows to learn a safe policy in a model-based fashion without actually observing the SRE. We introduce single- and multi-agent robust TD methods using the operator κ. We prove convergence of the operator to the optimal safe Q-function with respect to the model using the theory of Generalized Markov Decision Processes. In addition we prove convergence to the optimal Q-function of the original MDP given that the probability of SREs vanishes. Empirical evaluations demonstrate the superior performance of κ-based TD methods both in the early learning phase as well as in the final converged stage. In addition we show robustness of the proposed method to small model errors, as well as its applicability in a multi-agent context.

Keywords: reinforcement learning; robust learning; multi-agent learning

1 Introduction

Many critical systems exhibit global system dynamics that are highly sensitive to the local performance of individual components. This holds for example for (air) traffic and transport networks, communication networks, security systems, and (smart) power grids [5, 11, 14, 22]. In each case, the failure of or malicious attack on a small set of nodes may lead to knock-on effects that can potentially destabilise the whole system. Moreover, innovations in critical systems may introduce additional vulnerabilities to such attacks: e.g., in smart grids communication channels are needed for distributed intelligent energy management strategies, while simultaneously forming a potential target that could compromise safety [32]. Our research is motivated precisely by the need for safety in these critical systems, which can be achieved by building in robustness against rare but significant deviations caused by one or more system components failing or being compromised in an attack.

In this article we present a new approach for learning policies in such systems that are safe and robust against a chosen scenario of potential attacks or failures. We accomplish this by introducing a

This paper will appear in the proceedings of AAMAS'19.

new Q-function operator, which we call the κ-operator, that encodes robustness into the bootstrapping update of traditional temporal difference (TD) learning methods. In particular, we design the operator to encode the possibility of significant rare events (SREs) without requiring the learning agent to observe such events in training. Although the κ-operator is model-based with respect to these SREs, it can be combined with any TD method and can thus still be model-free with respect to the environment dynamics.

We prove convergence of our method to the optimal safe Q-function with respect to the model using the theory of Generalized Markov Decision Processes. In addition we prove convergence to the optimal Q-function of the original MDP given that the probability of SREs vanishes. Empirical evaluations demonstrate the superior performance of κ-based TD methods both in the early learning phase as well as in the final converged stage. In addition we show robustness of the proposed method to small model errors, as well as its applicability to multi-agent joint-action learning.

The remainder of this article is structured as follows: The next section embeds our approach in related work, followed in Section 3 by an introduction of background concepts. Section 4 introduces the new TD operator κ, for which we subsequently prove convergence. Section 6 provides empirical results, and Section 7 concludes.

2 Related Work

The aim to find robust policies is relevant to multiple research areas, including security games, robust control/learning, safe reinforcement learning and multi-agent reinforcement learning.

The domain of **security games** has expanded in recent years with many real-world applications in critical domains [18, 21], where the main approach has been computing exact solutions and deriving strong theoretical guarantees, mostly using equilibria concepts such as Nash and Stackelberg equilibria [12, 15]. On the one hand, our work adopts the information asymmetry assumption often used in Stackelberg Security games [12], providing the model of attack types for the *leader*, and allowing leader-strategy-informed best response strategies by attackers. On the other, we base our approach on *reinforcement learning from interactions* with the environment, thus we do not need to know the system model. Until now there has been substantially less work on reinforcement learning for security games than on game-theoretic approaches, exceptions being for example Ruan et al. [20] and Klima et al. [10] who use reinforcement learning in the context of patrolling and illegal rhino poaching problems, respectively.

Similar to security games, control theory starts with a model of the system to be controlled (the *plant*), and for the purpose of **robust control** assumes a set of possible plants as an explicit model of uncertainty, seeking to design a policy that stabilises all these plants [33]. A slightly weaker assumption is made in related work that assumes control over the number of observations for *significant rare events* (SREs), performing updates by sampling [3]. It introduces a policy gradient variant to improve learning in presence of SREs, using a *proposal distribution* that controls data from which it learns and *importance sampling* to adjust updates. In contrast, our work assumes that the model of this system is not known a priori, and a policy needs to be learned by interacting with it, as in robust learning.

While early work on **robust reinforcement learning** focused on learning within parameterised acceptable policies [24], later work transferred the objective of maximising tolerable disturbances from control theory to reinforcement learning [16]. Our work is similar to the therein defined *Actor-*

disturber-critic, but we replace its model of minimax simultaneous actions with stochastic transitions between multiple controllers (one being in control at any time) with arbitrary objectives for each controller. Similarly, our approach has commonalities with the **multi-agent reinforcement learning** algorithm Minimax-Q [13] for zero-sum games, which assumes minimisation over the opponent action space. However, in contrast, we define an attack to minimise over our own action space, and thus learn (but not enact) simultaneously our optimal policy and the (rare) attacks it is susceptible to. We further cover not only minimising adversaries but also random failures or any other policy encoding other adversaries' agendas (see Section 4.1). While *on-policy* learning algorithms have been shown to perform better than classical Q-learning in perturbed environment [23], and can thus in some sense be considered safer (against mistakes or exploration), our method combines with both on- and off-policy learning, and provides robustness against a chosen target.

3 Background

This work belongs to the field of Reinforcement learning (RL) [27], and makes use of the core concept of a *Markov Decision Process (MDP)*. An MDP is formally defined by a tuple (S, A, R, P), where S is a finite set of states, A is a finite set of actions, $R(s, a) \to r \in \mathbb{R}$ is a reward function for a given state $s \in S$ and an action $a \in A$ and $P(s'|s, a)$ is a transition function giving a probability of reaching state s' after taking action a in state s. In this work we also consider a multi-agent setting, which uses the formulation of the *Stochastic game*, which is a generalization of MDP to multiple agents and is defined by a tuple $(n, S, A_1 \dots A_n, R_1 \dots R_n, P)$, extending the MDP, where n is the number of agents, A_i is the action space of agent i. The joint action space is $A = A_1 \cup \dots \cup A_n$, and a joint action is $\boldsymbol{a} = (a_1, a_2, \dots, a_n)$.[1] $R_i(s, a_i, \boldsymbol{a_{-i}}) \to r_i$ is the reward function of agent i for given state s and joint action $\boldsymbol{a}$, and $P(s'|s, \boldsymbol{a})$ is the state transition function.

The main goal of RL is finding an optimal policy for given MDP. One of the most common methods is *Temporal difference (TD)* learning, which is a one-step bootstrapping method based on Bellman style equations. Of crucial importance in the TD learning is the definition of the TD error, describing the difference between already learnt value and new information about the value obtained from interacting with the environment. TD error, as the difference between the *target* and the current value, prescribes the type of adjustment we make to the already learnt value. In this work we focus on modifying the target, which has the standard form of $r + \gamma V(s')$, where γ is the discount factor and $V(s')$ is the value of the next state s'. The target can be induced by the behaviour policy in which case we are talking about *on-policy* methods or by something else (e.g., maximization over the action space) in which case we arrive to *off-policy* type of learning. For further explanation of common RL concepts used in this paper we refer the reader to Sutton and Barto [27].

4 The Robust TD Operator κ

We now present our robust TD operator κ. Before we formally define the operator, we give an intuitive example. Suppose a Q-learning agent needs to learn a safe policy against a potential

[1] We use the common shorthand $\boldsymbol{a_{-i}}$ to denote the joint action of all agents except agent i, i.e., $\boldsymbol{a_{-i}} = (a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$.

malicious adversary who could, with some probability $\varkappa$, take over control in the next state.[2] The value of the next state s_{t+1}, thus, depends on who is in control[3]: if the agent is in control, she can choose an optimal action that maximizes expected return; or if the adversary is in control he might, in the worst case, aim to minimize the expected return. This can be captured by the following modified TD error

$$\delta_t = r_{t+1} + \gamma \left((1-\varkappa) \max_a Q(s_{t+1}, a) + \varkappa \min_a Q(s_{t+1}, a) \right) - Q(s_t, a_t),$$

where we assume that the agent has knowledge of (or can estimate) the probability $\varkappa$.

In the following we first present a formal, general model of the operator κ, by modifying the target in the classical Bellman style value function. We then present practical implementations of TD(κ) methods that use this operator for both single- and multi-agent settings, based on the classical on- and off-policy TD learning algorithms (Expected) SARSA and Q-learning.

4.1 Formal Model

We consider a set of m possible *control policies* $C = \{\sigma_1, \ldots, \sigma_m\}$. At each time step, one of these policies is in control (and thus decides on the next action) with some probability $p(\sigma_i|s)$ that may depend on the state s. The set C and probability function $p(\cdot)$ are assumed to be (approximately) known by the agent. In our new TD methods, the value of the next state s' then becomes a function of both, the state and the function $p(\cdot)$, which we capture in our proposed operator κ, as $V^\kappa(s')$. Note that the set C includes the focal policy π that we seek to optimise in face of (possibly adversarial) alternative controllers. Such external control policies can represent for example a malicious attacker, aiming to minimize the expected return, or any arbitrary dynamics, such as random failures, represented by, e.g., a uniformly random policy. Based on a prior assumption about the nature of σ we want to optimise the focal policy π without necessarily observing actual attacks or failures. This means learning our robust policy π right from the start.

We define σ in terms of our own Q-value function, for example an attacker that is minimising our expected return. Thus we need to learn only one Q-value function Q^π. This is similar to the standard assumption in Stackelberg games that the attacker is able to fully observe our past actions and thus can enact the informed best response. We define the Q-value function update for our policy π based on standard Bellman equation and given the operator κ as

$$Q^\pi(s,a) \leftarrow Q^\pi(s,a) + \alpha \Big[\underbrace{r + \gamma V^\kappa(s')}_{\text{target}} - Q^\pi(s,a) \Big]. \tag{1}$$

Note that where in the standard Bellman equation we would have $V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a)$, in our case we have

$$V^\kappa(s) = \sum_{\sigma \in C} p(\sigma|s) \sum_a \sigma(s,a) Q^\pi(s,a), \tag{2}$$

[2]We use the symbol κ to denote the proposed TD operator and the symbol $\varkappa$ for the parameter of probability of attack, both are different versions of the letter "kappa".

[3]Note that while a token of control could be included in the state (doubling its size), our approach rather directly applies model-based bootstrap updates. This makes it explicit that the robustness target is a chosen parameter of the operator, and makes it possible to learn robust strategies before observing SREs, or when learning does not occur during SREs due to compromisation.

computed as a weighted sum over all possible control policies $\sigma \in C$. Note that we can learn Q^π without actually experiencing any attack or malfunction, based only on prior assumptions about the possible control policies as captured by the operator κ. We refer to this target modification as the operator κ because it closely resembles the Bellman optimality operator $\mathcal{T}^\star$, which is defined as $\mathcal{T}^\star V(s) = \max_a \big[R(s,a) + \sum_{s'} P(s'|s,a)\gamma V(s')\big]$. Thus, we can then formally define the κ optimality operator $\mathcal{T}^\star_\kappa$ by substituting the value function $V(\cdot)$ with $V^\kappa(\cdot)$.

In the following we present several κ-versions of classical TD methods. For simplicity we assume a scenario in which we have only a single adversarial external policy σ that aims to minimize our value, and thus $C = \{\pi, \sigma\}$. Note however that our model is general, and would work for any C and $p(\cdot)$.

4.2 Examples of TD(κ) Methods

We first present single-agent κ-based learning methods by building on the standard TD methods Q-learning and Expected SARSA. Then we present two-agent joint-action learning approaches. Although a generalization to n agents is relatively straightforward, we choose to focus solely on the single- and two-agent case in this paper for clarity of exposition. In each case, we consider the setting in which either the focal agent, with policy π, is in control, or the external adversary with policy σ aiming to minimize return. We further simplify the model by making the control policy probability function $p(\cdot)$ state-independent, reducing it to a probability vector.

4.2.1 Single-Agent Methods

Before we present the algorithms, it is important to note that we need to distinguish the *target* and *behaviour* policies. The κ-operator is defined on the target (see Eq. (1)), while the behaviour policy is used only for selecting actions. We assume an ϵ-greedy behaviour policy throughout.

In **off-policy Q(κ)**, the target policy is the greedy policy $\pi(s) = \arg\max_a Q(s,a)$ that maximizes expected return. The adversarial policy on the other hand aims to minimize the return, i.e., $\sigma(s) = \arg\min_a Q(s,a)$. Assuming a probability of attack of $\varkappa$ as before, we have $p(\pi) = (1-\varkappa)$ and $p(\sigma) = \varkappa$. Thus, Eq. (2) becomes

$$V^\kappa(s) = (1-\varkappa)\max_a Q(s,a) + \varkappa \min_a Q(s,a).$$

For **on-policy Expected SARSA(κ)** the target is the (expectation over the) focal policy π, while the adversarial policy σ remains the same as before. Thus, we have

$$\begin{aligned} V^\kappa(s) &= (1-\varkappa)\mathbb{E}_{a\sim\pi}\big[Q(s,a)\big] + \varkappa \min_a Q(s,a) \\ &= (1-\varkappa)\sum_a \pi(a|s)Q(s,a) + \varkappa \min_a Q(s,a). \end{aligned}$$

4.2.2 Multi-Agent Methods

We move from a single-agent setting to a scenario in which multiple agents interact. For sake of exposition we only present a two-agent case, which we further examine in the remainder of this paper.

We assume two agents with different action spaces, A_1 and A_2, but an identical reward function and thus a shared joint action Q-value function $Q : S \times A_1 \times A_2 \to \mathbb{R}$. Moreover, we assume full communication during the learning phase, allowing the agents to take each other's policies into account when selecting the next action.[4] Our algorithms are therefore based on the joint-action learning (JAL) paradigm [4]. We further assume that only one agent can be attacked at each time step.[5] For **multi-agent Q($\boldsymbol{\kappa}$)** we can write Eq. (2) for each individual agent as

$$\begin{aligned} V^{\kappa}(s) = (1-\varkappa) & \max_{A_1} \max_{A_2} Q(s, \langle a_1, a_2 \rangle) \\ + \quad \frac{\varkappa}{2} & \min_{A_1} \max_{A_2} Q(s, \langle a_1, a_2 \rangle) \\ + \quad \frac{\varkappa}{2} & \min_{A_2} \max_{A_1} Q(s, \langle a_1, a_2 \rangle) \end{aligned}$$

with $a_1 \in A_1$ and $a_2 \in A_2$, representing the scenario in which no attack happens with probability $(1-\varkappa)$, and each agent is attacked individually with probability $\varkappa/2$.[6] Analogously, we can define Eq. (2) for **multi-agent Expected SARSA($\boldsymbol{\kappa}$)** as

$$\begin{aligned} V^{\kappa}(s) = (1-\varkappa) & \quad \mathbb{E}_{a_1 \sim \pi_1, a_2 \sim \pi_2}\big[Q(s, \langle a_1, a_2 \rangle)\big] \\ + \quad \frac{\varkappa}{2} & \min_{A_1} \mathbb{E}_{a_2 \sim \pi_2}\big[Q(s, \langle a_1, a_2 \rangle)\big] \\ + \quad \frac{\varkappa}{2} & \min_{A_2} \mathbb{E}_{a_1 \sim \pi_1}\big[Q(s, \langle a_1, a_2 \rangle)\big] \end{aligned}$$

where we now compute an expectation over the actual policy of the agents that are not attacked, while the attacker is still minimizing.

5 Theoretical Analysis

In this section we analyze theoretical properties of the proposed κ-methods. We start by relating the different algorithms to each other in the limit of their respective parameters. Then we proceed to show convergence of both Q(κ) and Expected SARSA(κ) to two different fixed points: (i) to the optimal value function $Q^{\star}$ of the original MDP in the limit where $\varkappa \to 0$; and (ii) to the optimal robust value function $Q^{\star}_{\kappa}$ of the MDP that is *generalized* w.r.t. κ for constant parameter $\varkappa$. Note that *optimality* in this sense is purely induced by the relevant operator. In (i) this is the standard Bellman optimality which maximizes the expected discounted return of the MDP. However, in (ii) we derive optimality in the context of *Generalized MDPs* [29], where optimal simply means the fixed point of a given operator, which can take many forms.

Before proceeding with the convergence proofs, Figure 1 summarizes some relationships between the algorithms in terms of their targets, in the limit of their respective parameters: As is known, Expected SARSA, SARSA, and Q-learning become identical in the limit of a greedy policy [27, 31]. Furthermore, the update targets of our κ-methods approach the update targets of the standard TD methods on which they are based as $\varkappa \to 0$. Finally, Expected SARSA(κ) and Q(κ) share

[4] A common practice in cooperative multi-agent learning settings, see e.g., [7, 26].

[5] Although relaxing this assumption is straightforward, we opt to keep it for clarity.

[6] Note the order of the min max, which follows the Stackelberg assumption of an all-knowing attacker who moves last.

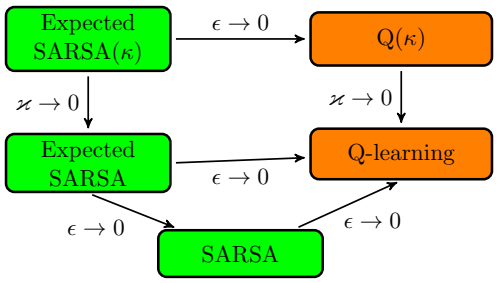

Figure 1: The relationship between the learning targets of different algorithms in the limits of their parameters. On-policy methods are in green, off-policy methods in orange.

the same relationship as their original versions, and thus Expected SARSA(κ) approaches Q(κ) as $\epsilon \to 0$. Note that the algorithms' equivalence in the limit does not hold in the transient phase of the learning process, and hence in practice they may converge on different paths and to different policies that share the same value function. For a comprehensive understanding of the algorithms introduced in Section 4.2, the following sections provide proofs for both convergence of κ methods for $\varkappa \to 0$, as well as their convergence when $\varkappa$ stays constant.

While we focus on the adversarial targets considered in Section 4.2, a previous proof of convergence under persistent exploration [29] can be interpreted as a model of random failures with fixed kappa.

5.1 Convergence to the Optimal $Q^\star$

There exist several proofs of convergence for the temporal difference algorithms Q-learning [9, 30], SARSA [23], and Expected SARSA [31]. Each of these proofs hinges on linking the studied algorithm to a stochastic process, and then using convergence results from stochastic approximation theory [6, 19]. These proofs are based on the following lemma, presented as Theorem 1 in Jaakkola et al. [9] and as Lemma 1 in Singh et al. [23]. These differ in the third condition, which describes the contraction mapping of the operator. The contraction property used for the Q-learning proof [9] has the form $||\mathbb{E}\{F_t(\cdot)|P_t\}|| \leq \gamma||\Delta_t||$, where $\gamma \in [0, 1)$. We show the lemma as it was used for the SARSA proof provided by Singh et al. [23], who show that the contraction property does not need to be strict; strict contraction is required to hold only asymptotically.

Lemma 5.1. *Consider a stochastic process $(\alpha_t, \Delta_t, F_t), t \geq 0$, where $\alpha_t, \Delta_t, F_t : X \to \mathbb{R}$ satisfy the equations*

$$\Delta_{t+1}(x) = \big(1 - \alpha_t(x)\big)\Delta_t(x) + \alpha_t(x)F_t(x),\ x \in X,\ t = 0, 1, 2, \ldots$$

Let P_t be a sequence of increasing σ-fields such that α_0 and Δ_0 are P_0-measurable and α_t, Δ_t and F_{t-1} are P_t-measurable, $t = 1, 2, \ldots$. Then, Δ_t converges to zero with probability one (w.p.1) under the following assumptions:

1. *the set X is finite,*
2. *$0 \leq \alpha_t(x_t) \leq 1$, $\sum_t \alpha_t(x_t) = \infty$, $\sum_t \alpha_t^2(x_t) < \infty$ w.p.1,*
3. *$||\mathbb{E}\{F_t(\cdot)|P_t\}|| \leq \gamma||\Delta_t|| + c_t$, where $\gamma \in [0, 1)$ and c_t converges to zero w.p.1,*
4. *$Var\{F_t(x_t)|P_t\} \leq K(1 + ||\Delta_t||)^2$, where K is some constant,*

where $||\cdot||$ denotes a maximum norm.

The proof continues by relating Lemma 5.1 to the temporal difference algorithm, following the same reasoning as Van Seijen et al. [31] in their convergence proof for Expected SARSA. We define $X = S \times A$, $P_t = \{Q_0, s_0, a_0, r_0, \alpha_0, s_1, a_1, \ldots, s_t, a_t\}$, $x_t = (s_t, a_t)$, which represents the past at step t and $\alpha_t(x_t) = \alpha_t(s_t, a_t)$ is a learning rate for state s_t and action a_t. To show the convergence of Q to the optimal fixed point $Q^\star$ we set $\Delta_t(x_t) = Q_t(s_t, a_t) - Q^\star(s_t, a_t)$, therefore when Δ_t converges to zero, then the Q values converge to $Q^\star$. The maximum norm $||\cdot||$ can be expressed as maximizing over states and actions as $||\Delta_t|| = \max_s \max_a |Q_t(s,a) - Q^\star(s,a)|$.

We follow the reasoning of Theorem 1 from Van Seijen et al. [31], where we repeat the conditions (1), (2) and (4) and modify the condition **(3)** for the κ methods as:

Theorem 5.2. *Q(κ) and Expected SARSA(κ) as defined in Section 4.2.1 using the respective value function V^κ, defined by*

$$Q_{t+1}(s_t, a_t) = (1 - \alpha_t(s_t, a_t))Q_t(s_t, a_t) + \alpha_t(s_t, a_t)[r_t + \gamma V_t^\kappa(s_{t+1})]$$

converge to the optimal Q function $Q^\star(s,a)$ if:

1. *the state space S and action space A are finite,*
2. *$\alpha_t(s_t, a_t) \in (0,1)$, $\sum_t \alpha_t(s_t, a_t) = \infty$ and $\sum_t \alpha_t^2(s_t, a_t) < \infty$ w.p.1,*
3. *$\varkappa$ converges to zero w.p.1,*
3a. *for Expected SARSA(κ) the policy is greedy in the limit with infinite exploration (GLIE assumption),*
4. *the reward function is bounded.*

Proof. Convergence of Q(κ): To prove convergence of Q(κ) we have to show that the conditions from Lemma 5.1 hold. Conditions (1), (2) and (4) of Theorem 5.2 correspond to conditions (1), (2) and (4) of Lemma 5.1 [31]. We now need to show that the contraction property holds as well, using condition (3) of Theorem 5.2. Adapting the proof of Van Seijen et al. [31], we set $F_t(x) = F_t(s,a) = r_t(s,a) + \gamma V_t^\kappa(s') - Q^\star(s,a)$ to show that $F_t(s,a)$ is a contraction mapping, i.e., condition (3) in Lemma 5.1. For Q(κ) we write:

$$F_t = r_t + \gamma\big((1-\varkappa)\max_a Q_t(s_{t+1}, a) + \varkappa \min_a Q_t(s_{t+1}, a)\big) - Q^\star(s_t, a_t).$$

We want to show that $||\mathbb{E}\{F_t\}|| \leq \gamma||\Delta_t|| + c_t$ to prove the convergence of Q(κ) to the optimal value $Q^\star$.

$$\begin{aligned}
||\mathbb{E}\{F_t\}|| &= ||\mathbb{E}\{r_t + \gamma\big((1-\varkappa)\max_a Q_t(s_{t+1}, a) + \varkappa \min_a Q_t(s_{t+1}, a)\big) - Q^\star(s_t, a_t)\}|| \\
&\leq ||\mathbb{E}\{r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q^\star(s_t, a_t)\}|| + \\
&\quad \gamma||\mathbb{E}\{\varkappa \min_a Q_t(s_{t+1}, a) - \varkappa \max_a Q_t(s_{t+1}, a)\}|| \\
&\leq \gamma \max_s |\max_a Q_t(s,a) - \max_a Q^\star(s,a)| + \\
&\quad \gamma \max_s |\varkappa \min_a Q_t(s,a) - \varkappa \max_a Q_t(s,a)| \\
&\leq \gamma||\Delta_t|| + \\
&\quad \gamma\varkappa \max_s |\min_a Q_t(s,a) - \max_a Q_t(s,a)|,
\end{aligned}$$

where the first inequality follows from standard algebra and the fact that splitting the maximum norm yields at least as large a number, the second inequality follows from the definition of $Q^\star$[7] and the maximal difference in values over all states being at least as large as a difference between values given in state s_{t+1}, and the third inequality follows from the definition of $||\Delta_t||$ above. We can see that if we set $c_t = \gamma\varkappa \max_s |\min_a Q_t(s,a) - \max_a Q_t(s,a)|$, then for $\varkappa \to 0$ we get c_t converging to zero w.p.1, thus proving convergence of Q(κ). ■

Proof. Convergence of Expected SARSA(κ): Similarly as in the proof of Q(κ) we need to show that the contraction property holds as well, this time using conditions (3) and (3a) of Theorem 5.2. We first define:

$$F_t = r_t + \gamma\big((1-\varkappa)\sum_a \pi_t(a|s_{t+1})Q_t(s_{t+1},a) + \varkappa \min_a Q_t(s_{t+1},a)\big) - Q^\star(s_t,a_t)$$

and then show the following:

$$\begin{aligned}
||\mathbb{E}\{F_t\}|| &= ||\mathbb{E}\{r_t + \gamma\big((1-\varkappa)\sum_a \pi_t(a|s_{t+1})Q_t(s_{t+1},a) + \varkappa \min_a Q_t(s_{t+1},a)\big) - Q^\star(s_t,a_t)\}|| \\
&\leq ||\mathbb{E}\{r_t + \gamma \max_a Q_t(s_{t+1},a) - Q^\star(s_t,a_t)\}|| + \\
&\quad \gamma||\mathbb{E}\{(1-\varkappa)\sum_a \pi_t(a|s_{t+1})Q_t(s_{t+1},a) + \varkappa \min_a Q_t(s_{t+1},a) - \max_a Q_t(s_{t+1},a)\}|| \\
&\leq \gamma \max_s |\max_a Q_t(s,a) - \max_a Q^\star(s,a)| + \\
&\quad \gamma \max_s |(1-\varkappa)\sum_a \pi_t(a|s)Q_t(s,a) + \varkappa \min_a Q_t(s,a) - \max_a Q_t(s,a)|,
\end{aligned}$$

where the inequalities use the same operations as above in the proof of Q(κ). If we set $c_t = \gamma \max_s |(1-\varkappa)\sum_a \pi_t(a|s)Q_t(s,a) + \varkappa \min_a Q_t(s,a) - \max_a Q_t(s,a)|$ and assume that the policy is greedy in the limit with infinite exploration (GLIE assumption) and parameter $\varkappa \to 0$ w.p.1 (conditions (3) and (3a)), it follows that c_t converges to zero w.p.1, thereby proving that Expected SARSA(κ) converges to optimal fixed point $Q^\star$. ■

5.2 Convergence to the Robust $Q^\star_\kappa$

In this section we show convergence to the robust value function $Q^\star_\kappa$ which is optimal w.r.t. the operator κ. The main difference with the proof of Theorem 5.2 is that here we do not require $\varkappa \to 0$ but instead assume it remains constant over time. We base our reasoning on the theory of *Generalized MDPs* [29]. A Generalized MDP is defined using operator-based notation as

$$\Big(\bigotimes \bigoplus (R + \gamma V)\Big)(s) = \max_a \sum_{s'} P(s'|s,a)\big(R(s,a) + \gamma V(s')\big),$$

where the operator $\bigotimes$ defines how an optimal agent chooses her actions (in the classic Bellman equation this denotes maximization) and operator $\bigoplus$ defines how the value of the current state is updated by the value of the next state (in the classic Bellman equation this denotes a probability

[7] Recall that we set out in this section to show convergence to the same optimal Q-value as classical Q-learning $Q^\star(s_t,a) = r_t + \gamma \max_{a'} Q^\star(s_{t+1},a')$, even if we do so by our new operator.

weighted average over the transition function). These operators can be chosen to model various different scenarios. The generalized Bellman equation can now be written as $V^\star = \bigotimes \bigoplus (R + \gamma V^\star)$. The main result of Szepesvari and Littman [29] is that if $\bigotimes$ and $\bigoplus$ are non-expansions, then there is a unique optimal solution to which the generalized Bellman equation converges, given certain assumptions. For $0 \leq \gamma < 1$ and non-expansion properties of $\bigotimes$ and $\bigoplus$ we get a contraction mapping of the Bellman operator $\mathcal{T}$ defined as $\mathcal{T}V = \bigotimes \bigoplus (R + \gamma V)$. Then, the operator $\mathcal{T}$ has a unique fixed point by the Banach fixed-point theorem [25].

Building on the stochastic approximation theory results (as we also used in the Section 5.1), Szepesvari and Littman [29] show the following:

Lemma 5.3. *Generalized Q-learning with operator $\bigotimes$ using Bellman operator*

$$\mathcal{T}_t(Q', Q)(s, a) = \begin{cases} \big(1 - \alpha_t(s,a)\big) Q'(s,a) + \alpha_t(s,a)\big(r_t + \gamma(\bigotimes Q)(s'_t)\big) & \text{if } s = s_t, a = a_t \\ Q'(s,a) & \text{otherwise} \end{cases}$$

converges to the optimal Q function w.p.1, if

1. *s'_t is randomly selected according to the probability distribution defined by $P(s_t, a_t, \cdot)$,*
2. *$\alpha_t(s_t, a_t) \in (0, 1)$, $\sum_t \alpha_t(s_t, a_t) = \infty$ and $\sum_t \alpha_t^2(s_t, a_t) < \infty$ w.p.1,*
3. *$\bigotimes$ is a non-expansion,*
4. *the reward function is bounded.*

We base our convergence proofs for Q(κ) and Expected SARSA(κ) on the insights of Szepesvari and Littman [29] given in Lemma 5.3.

Theorem 5.4. *Q(κ) and Expected SARSA(κ) as defined in Section 4.2.1 converge to the robust Q function $Q^\star_\kappa$ for any fixed $\varkappa$.*

Proof. Convergence of Q(κ) to $Q^\star_\kappa$: To prove convergence of Q(κ) we follow the proof of Generalized Q-learning in Lemma 5.3. The only condition we need to guarantee is the non-expansion property of the operator in the value function update, which for Q(κ) is a weighted average of the operators *min* and *max*. We write the operator $\bigotimes$ for Q(κ) as $\bigotimes^\kappa$ and define it as

$$(\overset{\kappa}{\bigotimes} Q)(s, a) = (1 - \varkappa) \max_a Q(s, a) + \varkappa \min_a Q(s, a).$$

In Appendix B of Szepesvari and Littman [29], Theorem 9 states that any linear combination of non-expansion operators is also a non-expansion operator. Moreover Theorem 8 states that the summary operators max and min are also non-expansions. Therefore, $\bigotimes^\kappa$ is a non-expansion as well, thus proving the convergence of Q(κ) to the robust fixed point $Q^\star_\kappa$ induced by the operator κ. ■

Proof. Convergence of Expected SARSA(κ) to $Q^\star_\kappa$: We base our convergence proof of Expected SARSA(κ) again on the work of Szepesvari and Littman [29], this time on their insights regarding

persistent exploration (Section 4.5 in their paper). They show that Generalized Q-learning with ϵ-greedy action selection converges, for a fixed ϵ, in the Generalized MDP. Following similar reasoning, we define the operator $\bigotimes$ for Expected SARSA(κ) with fixed ϵ as

$$(\overset{\kappa}{\bigotimes} Q)(s,a) = (1-\varkappa)\left(\epsilon\frac{1}{|A|}\sum_a Q(s,a) + (1-\epsilon)\max_a Q(s,a)\right) + \varkappa \min_a Q(s,a).$$

Again, from repeated application of Theorems 8 and 9 in Appendix B of Szepesvari and Littman [29] it follows that $\bigotimes^\kappa$ is a non-expansion as well. Therefore, by Lemma 5.3, Expected SARSA(κ) converges to $Q^\star_\kappa$ for fixed exploration ϵ. ■

It remains an open question whether Expected SARSA(κ) also converges for decreasing ϵ, e.g., under the GLIE assumption, even though we conjecture that it might.

5.3 Convergence in the Multi-Agent Case

We now prove convergence of the cooperative multi-agent variant of the κ methods presented in Section 4.2.2. This proof builds on the theory of Generalised MDPs, similar to the proofs presented in Section 5.2. Therefore this proof also assumes a fixed probability of attack $\varkappa$. In addition, we make use of the assumption that agents can communicate freely in the learning phase, and thus receive identical information and can build a common joint-action Q-table.

Theorem 5.5. *Multi-agent Q(κ) and Expected SARSA(κ) as defined in Section 4.2.2 converge to the robust Q function $Q^\star_\kappa$ for any fixed $\varkappa$.*

Proof. The $\bigotimes^\kappa$ operator for our multi-agent versions of Q(κ) and Expected SARSA(κ) consists of a nested combination of different components, in particular $\max_a Q(s,a)$, $\min_a Q(s,a)$, and $\sum_a \pi^\epsilon(s,a)Q(s,a)$ where π^ϵ is the ϵ-greedy policy. By Theorem 8 of Szepesvari and Littman [29], max and min are non-expansions. By Theorem 9 of [29], linear combinations of non-expansion operators are also non-expansion operators. Finally, by Theorem 10 of [29], products of non-expansion operators are also non-expansion operators. Therefore, also max max, max min, and min max are non-expansion operators, as are linear combinations of those compounds. Similarly, $\sum_a \pi^\epsilon(s,a)Q(s,a)$ for fixed ϵ can be written as a linear combination of summary operators, which by Theorems 8 and 9 of Szepesvari and Littman [29] is a non-expansion. Therefore, the $\bigotimes^\kappa$ operator used in both multi-agent Q(κ) and Expected SARSA(κ) is a non-expansion. Thus, by Lemma 5.3, Q(κ) and Expected SARSA(κ) converge to $Q^\star_\kappa$ for fixed κ, and in the case of Expected SARSA(κ), for fixed ϵ. ■

6 Experiments and Results

In this section we evaluate temporal difference methods with the proposed operator κ; off-policy type of learning Q(κ) and on-policy type of learning Expected SARSA(κ). We experiment with the classical Cliff walking scenario for the single-agent case and the multi-agent Puddle world scenario. Both these domains contain some critical states, a cliff and a puddle respectively, which render very high negative reward for the agent(s) in case of stepping into them. These critical states represent the significant rare events (SREs). We compare our methods with classic temporal difference methods

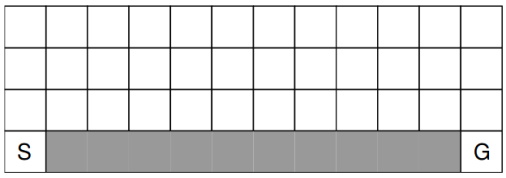

Figure 2: Cliff Walking: The agent needs to get from the start [S] to the goal [G], avoiding the cliff (grey tiles).

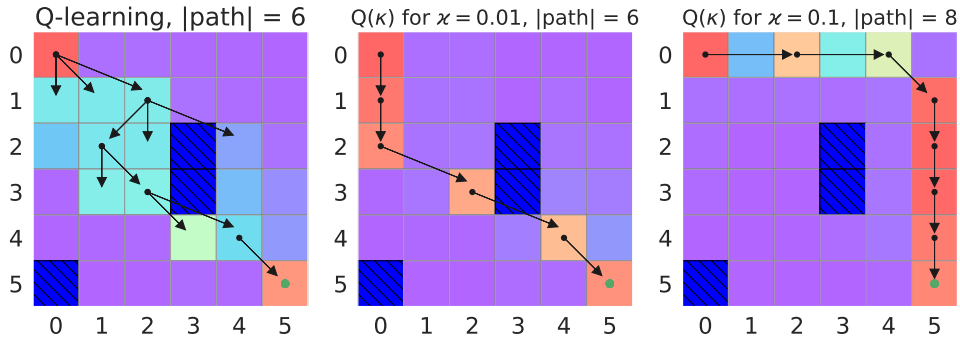

Figure 3: The Puddle world: Q(κ) learns a safer path with increasing $\varkappa$. Puddles are dark blue, the arrows show the optimal actions on the learned path, and the heatmap shows the number of visits to each state (, blue is none).

like SARSA, Q-learning and Expected SARSA. In all the experiments we consider an undiscounted (γ=1), episodic scenario.

Cliff walking: single-agent The Cliff walking experiment as shown in Figure 2 is a classical scenario proposed in Sutton and Barto [27] and used in many other papers ever since (e.g., [31]). The agent needs to get from the start state [S] to the goal state [G], while avoiding stepping into the cliff, otherwise rendering a reward of -100 and sending him back to the start. For every move which does not lead into the cliff the agent receives a reward of -1.

Puddle world: multi-agent The puddle world environment is a grid world with puddles which need to be avoided by the joint-learning agents. The two agents jointly control the movement of a single robot in this puddle world, with each controlling either direction $\langle \text{up}, \text{down} \rangle$ or $\langle \text{left}, \text{right} \rangle$. Agent 1 can take the actions {*stay, move down, move up*} and agent 2 can choose {*stay, move left, move right, move right by 2*}, thus their action spaces are different, further complicating the learning process compared to the single-agent scenario. The joint action is the combination of the two selected actions. We assume a reward of -1 for every move and -100 for stepping into a puddle (returning to the start node). The agents have to move together from the start node at the top left corner to the goal at the bottom right corner. Figure 3 shows the policy learned by our proposed algorithm Q(κ) for the two joint-learning agents. Note how a safer path (longer, avoiding the puddles) is learned with increasing parameter $\varkappa$. For $\varkappa = 0$ our algorithm degenerates to Q-learning (left panel).

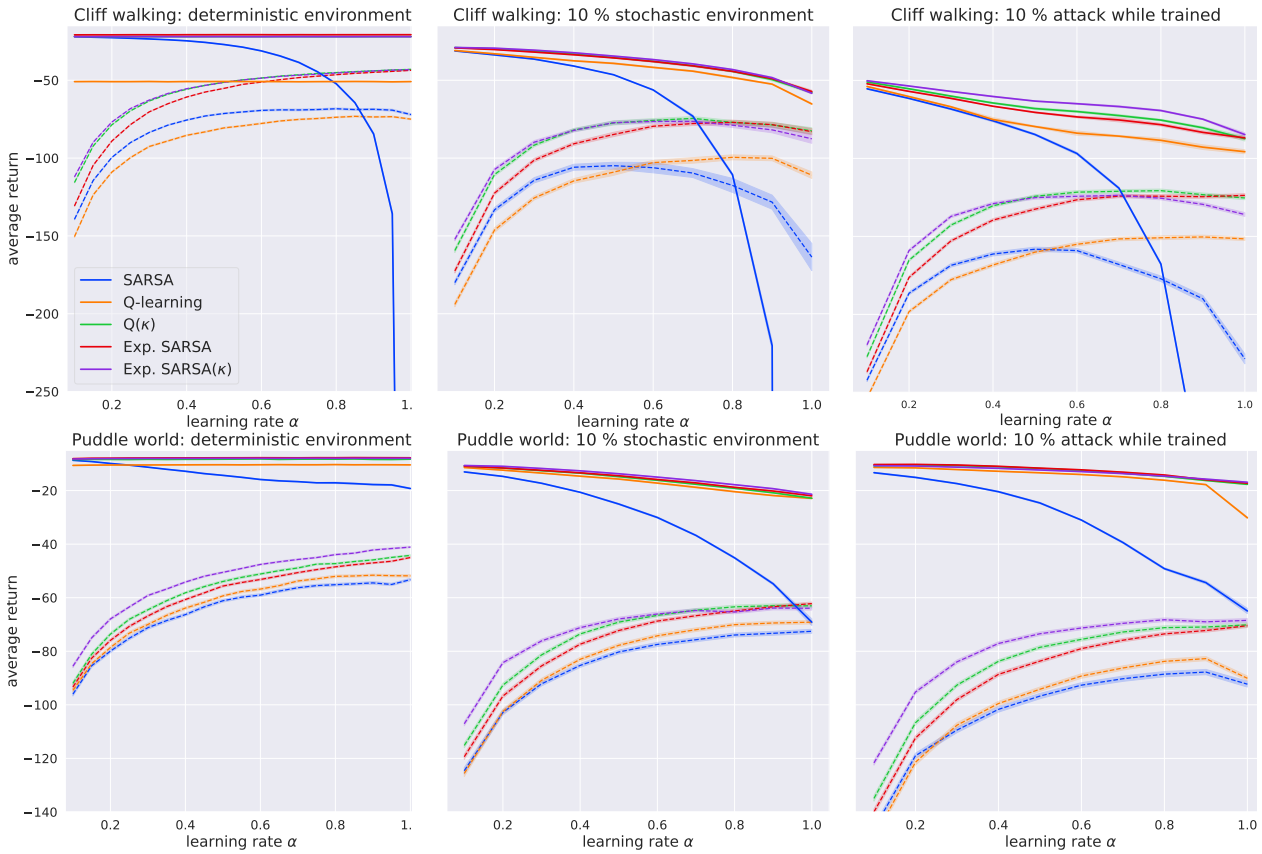

Figure 4: Cliff walking (single-agent) in first row and Puddle world (multi-agent) in second row. Deterministic environment (first column), 10 % stochastic environment (second column) and 10 % attack while training (third column). ϵ-greedy policy with fixed $\epsilon = 0.1$. Early performance - dashed lines (100 episodes), converged performance - solid lines (100, 000 episodes).

6.1 Performance

We replicate the experiment of Van Seijen et al. [31] on the cliff walking domain, which compares our κ methods with Q-learning, SARSA and Expected SARSA. In line with previous results we show (i) an early performance, which is average return over first 100 training episodes and (ii) converged performance, which is average return over 100, 000 episodes. Figure 4 shows this methodology on 3 different settings; (i) a deterministic environment, where each action chosen by the policy is executed with certainty, (ii) an environment with 10% stochasticity, where there is a random action taken with probability 10% instead of the chosen action and (iii) an environment with 10% probability of attack, where there is an adversarial action taken with probability 10% instead of the chosen action, as discussed before we define an attack as an action that minimizes the Q-value function in the given state. The 10% stochasticity in the environment can be seen as a random failure of the system. Furthermore, we also show the same set of experiments for the Puddle world in the second row of that figure. The early performance experiments (100 episodes) are averaged over 300 trials and the converged performance (10, 000 episodes) experiments are averaged over 10 trials. We also

show the 95% confidence intervals. Similarly to Van Seijen et al. [31] we fix the exploration rate to $\epsilon = 0.1$. For the κ methods we fix $\varkappa = 0.1$, further below we also experiment with different settings of the parameter $\varkappa$. Note that the y-axis, showing the average return, is the same in each row for a better comparison. The x-axis shows different learning rates α. We can see how the average return decreases with more complex scenarios, from deterministic, over to stochastic, to one with attacks. We can observe how the κ methods are superior to the other baselines in the early performance experiments, they are especially good in the attack case, which is the scenario the κ methods are the most suitable for. In the converged performance experiments the κ methods beat Q-learning and SARSA and are similar or better compared to Expected SARSA.

6.2 Different Levels of Probability of Attack

In this section we investigate how the methods behave under different levels of attack, defined by the probability of attack per state. We consider an attack on trained (converged) methods, thus we first train each method for $100,000$ episodes (in deterministic environment) and then we test it on $50,000$ trials with given probability of attack per state. We average the results over 10 trials and provide 95% confidence intervals. Note, that this is a different methodology of testing the methods against an adversarial attack compared to the experiments in Figure 4, where we considered attacks while training. This experiment shows the strength of the κ methods for different levels of attacks. We assume the probability of attack to be known here and thus we set the parameter $\varkappa$ to be equal to that probability, which is the meaning of the parameter $\varkappa$ as described before. In other words, parameter $\varkappa$ prescribes how much safely we want to act. We consider very rare attacks (0.001 probability of attack in each state) to more frequent attacks (0.2 probability of attack in each state) as shown in Figure 5. For better visualisation we use logarithmic axes. We train all the methods with learning rate $\alpha = 0.1$ and fixed exploration rate $\epsilon = 0.1$. We also experimented with different learning rates and conclude that all the methods, except SARSA, are quite stable for different learning rates, which can also be seen in the deterministic experiments in Figure 4. However, SARSA is very unstable for different learning rates (also see Figure 4), learns different paths for different learning rates and does not converge fast enough or not at all. This can be partly explained by higher variance of SARSA [31]. This instability of SARSA is demonstrated in our experiments by the wide confidence intervals in Figure 5. We test the different levels of probability of attack on the Cliff walking experiment in the left panel of Figure 5, where we can see that the κ methods compare favourably to the other baselines, however in some parts they give similar performance as Expected SARSA or SARSA. The Cliff walking experiment has a limited expressiveness for testing the methods due to a limited number of possible safe paths with low costs (see Figure 2), which is the reason for the κ methods to show only similar performance compared to the baselines and thus not being able to show the full potential of the κ methods. However, the Puddle world is more expressive, because there are several possible paths differing in level of safety and cost. The bigger solution space of the Puddle world is also induced by the 2 cooperating agents, each having their own action space. Therefore, on the right panel of Figure 5 we show the Puddle world experiment for different levels of probability of attack. Here, we can clearly see the κ methods outperform the baselines, especially Q(κ) is superior over the whole range of considered probabilities of attack. Note that Q(κ) learns a safer path even for very rare attacks (0.001 probability of attack), which also shows in Figure 3, where for 0.01 probability of attack Q(κ) learns a safer path with the same cost compared to Q-learning, which learns a path of the same distance but much closer to the puddles.

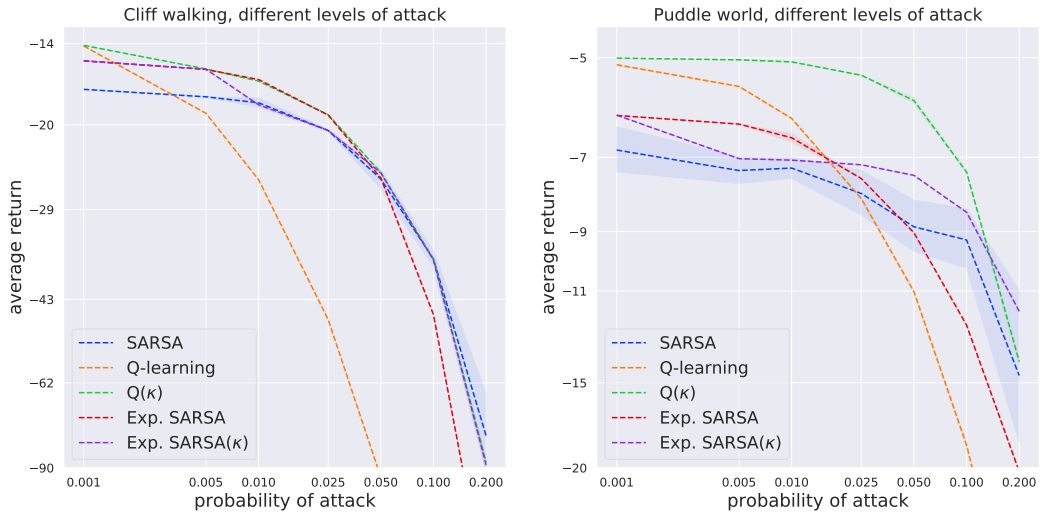

Figure 5: Varying probability of attack: Cliff walking (left), Puddle world (right), trained 100k, test 50k, $\alpha = 0.1$, $\epsilon = 0.1$.

6.3 Robustness Analysis

Here we test the robustness of the proposed algorithms with the operator κ by no longer assuming the probability of attack to be known and thus it is not possible to correctly set the $\varkappa$ parameter for Q(κ) and Expected SARSA(κ). Note that in our previous experiments we set the parameter $\varkappa$ to be equal to the actual probability of attack. In Figure 6 we show the performance of our algorithms for a range of actual attack probabilities (y-axis) while learning using a fixed parameter $\varkappa = 0.1$. Note that we no longer use logarithmic scales as in Figure 5 and do not consider very rare attacks, only probabilities of attack around 0.1. One can see that even for the cases where $\varkappa$ is not equal to the actual probability of attack the proposed κ algorithms still outperform the baselines in most cases, especially for Q-learning and Expected SARSA. In the Cliff walking experiment (the left panel in Figure 6) we get similar performance of the κ methods as SARSA, however SARSA is quite unstable as discussed before and as one can see by the width of the confidence interval. In the Puddle world experiment in the right panel of Figure 6 one can see the superior performance of κ methods, where they beat all the baselines even for fixed parameter $\varkappa$. Thus, we experimentally confirmed that even when we do not know the probability of attack accurately we can learn a better strategy by using the κ methods.

7 Discussion and Conclusion

We presented a new operator κ for temporal difference learning, which improves robustness of the learning process against potential attacks or perturbations in control. We proved convergence of Q(κ) and Expected SARSA(κ) to (i) the optimal value function $Q^\star$ of the original MDP in the limit where $\varkappa \to 0$; and (ii) the optimal robust value function $Q^\star_\kappa$ of the MDP that is generalized w.r.t. κ for constant parameter $\varkappa$. In the latter case we also proved convergence of a cooperative joint-action learning version of our methods.

Our complementary empirical results demonstrate that the proposed κ-methods indeed provide

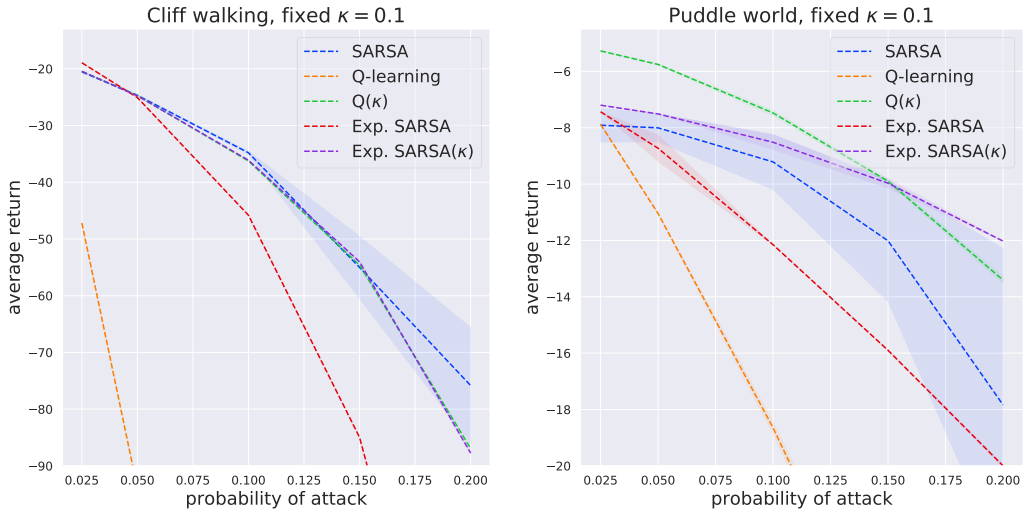

Figure 6: Robustness analysis: Cliff walking (left), Puddle world (right), trained 100k, test 50k, $\alpha = 0.1$, $\epsilon = 0.1$, $\varkappa = 0.1$.

robustness against a chosen scenario of potential attacks and failures in both single- and multi-agent settings. Although our method assumes that a model of such attacks and failures is known to the agent, we further demonstrate that our methods are robust against small model errors. Moreover, we show that even in absence of attacks or failures, our method learns a policy that is robust in general against environment stochasticity, in particular in the early stages of learning.

There are several interesting directions for future work. One possibility would be extending the control space, allowing for more agents being attacked or malfunctioning with different intensity. We defined the probability of control depending on the state, but more parameters could be introduced. Such extensions would narrow the reality gap and would allow for learning more complex policies, where we believe our approach could prove even more competitive. Furthermore, the target of adversarial policies could be learned from experience, where ideas from opponent modelling could be used (e.g., DPIQN [8]).

Our proposed operator κ can be closely linked or even combined with some recent state-of-the-art reinforcement learning methods. Considering a multi-step update, the operator could be combined with Retrace(λ) [17], which would potentially speed up convergence. Another promising extension of our model would be to combine it with Q(σ) [1] to allow for mixed multi-step updates. Note that the parameter σ in this algorithm can also be time- or state-dependent similarly to the potential extensions of the control in our model. This would allow to learn robust policies against more complex controls such as multi-step attacks. Another interesting extension along this line would be to model the control transition similar to the options framework [2, 28], in which case the alternate control policies could be seen as "malicious" options over which the agent has no control, with potentially complex initiation sets and termination conditions. Such extensions would further increase the flexibility of our proposed operator, making it applicable to a wide range of real-world scenarios.

Acknowledgement

This project has received funding in the framework of the joint programming initiative ERA-Net Smart Energy Systems' focus initiative Smart Grids Plus, with support from the European Union's Horizon 2020 research and innovation programme under grant agreement No 646039.

References

[1] Kristopher De Asis, J. Hernandez-Garcia, G. Holland, and Richard S. Sutton. Multi-step reinforcement learning: A unifying algorithm. In *AAAI Conference on Artificial Intelligence*, 2018.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of Association for the Advancement of Artificial Intelligence Conference (AAAI)*, pages 1726–1734, 2017.

[3] Kamil Andrzej Ciosek and Shimon Whiteson. OFFER: Off-environment reinforcement learning. In *Proceedings of Association for the Advancement of Artificial Intelligence Conference (AAAI)*, 2017.

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

[5] Flaviu Cristian, Bob Dancey, and Jon Dehn. Fault-tolerance in air traffic control systems. *ACM Transactions on Computer Systems (TOCS)*, 14(3):265–286, 1996.

[6] Aryeh Dvoretzky. On stochastic approximation. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 39–55. University of California Press, 1956.

[7] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*, 2017.

[8] Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. A deep policy inference q-network for multi-agent systems. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1388–1396, 2018.

[9] 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.

[10] Richard Klima, Karl Tuyls, and Frans Oliehoek. Markov Security Games: Learning in Spatial Security Problems. *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*, pages 1–8, 2016.

[11] John C Knight. Safety critical systems: challenges and directions. In *Proceedings of the 24th International Conference on Software Engineering*, pages 547–550. ACM, 2002.

[12] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. Nash in Security Games: An Extended Investigation of Interchangeability, Equivalence, and Uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

[13] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. Technical report, Brown University, 1994.

[14] Jing Liu, Yang Xiao, Shuhui Li, Wei Liang, and CL Philip Chen. Cyber security and privacy issues in smart grids. *IEEE Communications Surveys & Tutorials*, 14(4):981–997, 2012.

[15] Jian Lou, Andrew M Smith, and Yevgeniy Vorobeychik. Multidefender security games. *IEEE Intelligent Systems*, 32(1):50–60, 2017.

[16] Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2): 335–359, 2005.

[17] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1054–1062, 2016.

[18] James Pita, Manish Jain, Janusz Marecki, Fernando Ordonez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed ARMOR Protection: The Application of a Game Theoretic Model for Security at the Los Angeles International Airport. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, pages 1805–1812, 2008.

[19] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[20] Sui Ruan, Candra Meirina, Feili Yu, Krishna R Pattipati, and Robert L Popp. Patrolling in a stochastic environment. Technical report, Electrical and Computer Engineering Department, University of Connecticut, Storrs, 2005.

[21] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. PROTECT: A Deployed Game Theoretic System to Protect the Ports of the United States. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1:13–20, 2012.

[22] Martin L Shooman. *Reliability of computer systems and networks: fault tolerance, analysis, and design*. John Wiley & Sons, 2003.

[23] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3): 287–308, 2000.

[24] Satinder P Singh, Andrew G Barto, Roderic Grupen, and Christopher Connolly. Robust reinforcement learning in motion planning. In *Advances in neural information processing systems (NIPS)*, pages 655–662, 1994.

[25] D. R. Smart. *Fixed point theorems*. Cambridge University Press, Cambridge, 1974.

[26] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, and Thore Grapel. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[27] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. The MIT Press, Cambridge, MA, 1998.

[28] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[29] Csaba Szepesvari and Michael L Littman. Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms. Technical report, Brown University, 1997.

[30] John N Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.

[31] Harm van Seijen, Hado van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of Expected Sarsa. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL 2009*, pages 177–184, 2009.

[32] Ye Yan, Yi Qian, Hamid Sharif, and David Tipper. A survey on smart grid communication infrastructures: Motivations, requirements and challenges. *IEEE communications surveys & tutorials*, 15(1):5–20, 2013.

[33] Kemin Zhou and John Comstock Doyle. *Essentials of robust control*, volume 104. Prentice hall, Upper Saddle River, NJ, 1998.