# Learning robust policies when losing control

Richard Klima
University of Liverpool
United Kingdom

Daan Bloembergen
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands

Michael Kaisers
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands

Karl Tuyls
University of Liverpool
United Kingdom

## ABSTRACT

Many real-world applications require control strategies that provide robustness against a model of potential temporary *external control*, such as failures of the designed controller or malicious attacks. In this article we assume a Markovian control model as a stepping stone towards extending Q-learning akin to the options framework, but addressing the risk of losing control involuntarily to possibly malicious 'options'. The resulting reinforcement learning algorithm maximises expected return, and is model-free with respect to domain dynamics, but model-based with respect to control transitions. Our model allows to exploit parallel off-policy updates to efficiently learn from experience. Results demonstrate that effective safe strategies can be learned from mistakes, possibly even before attacks occur. Our algorithm compares favourably to on-policies SARSA and Expected SARSA and off-policy Q-learning in a multi-agent benchmark, can be trained using forward domain models, and is compatible with many state of the art extensions, such as deep learning, Retrace($\lambda$), or Q($\sigma$). We thus pave the way to learn robust strategies in critical multi-agent domains, such as smart grids, where graceful degradation is a prerequisite.

## KEYWORDS

Reinforcement Learning; Safety; Robust Learning

## 1 MOTIVATION AND RELATED WORK

Our research is mainly driven by the need for safety in critical systems, which can be put at risk by one or more system components failing or being compromised in an attack. Innovations in critical systems may yield vulnerabilities to such attacks: e.g., in smart grids communication channels are both needed and potential targets to compromise distributed intelligent energy management strategies [19], demanding new approaches for resilience and cybersecurity. Improving resilience and security in smart grids is a key point of attention for Dutch funding in this area [17].

Consider a distributed control problem, for which each controller executes an individual policy on separate hardware. Any of such controllers may be attacked, potentially affecting or changing the local policy. These attacks or failures can be rare but can have a profound impact on the whole system if the system is not prepared for them. The main question is then, how to obtain effective and robust policies in such systems. The reasons for a controller executing a different policy than desired include: an attack by an adversary (e.g. hacker), a natural disaster (e.g. earthquake) or a mechanical defect (e.g. node malfunction).

Our work touches upon multiple fields including robust control/learning, security games and safe reinforcement learning. 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 [20]. While our assumption of multiple controllers could be likened to distinct plant models, we here seek to maximise in expectation by assuming probabilities of alternate controllers taking over, reducing the model back to one system. In addition, our work assume 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 [14], later work transferred the objective of maximising tolerable disturbances from control theory to reinforcement learning [9]. 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. We thus cover not only minimising adversaries but also random failures or any other policy encoding other adversaries' agendas. In relation to the taxonomy of **safe reinforcement learning** [3] our method falls in between *Worst-Case Criterion under Parameter Uncertainty* and *Risk-Sensitive RL Based on the Weighted Sum of Return and Risk*, depending on the chosen alternate controller objectives.

The domain of **security games** has expanded in recent years with many real-world applications, examples include the ARMOR system for airport security [11] or the PROTECT system for scheduling Coast Guard [13]. The main approach to the security games in the related literature is computing exact solutions and deriving strong theoretical guarantees, mostly using equilibria concepts such as Nash equilibria and Stackelberg equilibria [6], with an extension to the Stackelberg multi-agent setting [8]. This direction of research in security games has been very important to theoretically underpin the field, however, it seems often difficult to deploy exact theoretical solution methods in real-world settings due to strict model assumptions or severe simplifications. Since we base our approach on learning from interactions with the environment we do not need to know the model of the system, which helps to overcome some of the weaknesses of the theoretical approaches.

There has been substantially less work done on using **reinforcement learning** for security games compared to merely game-theoretic approaches. The whole idea of using learning is primarily based on modelling the system as a Markov Decision Process (MDP) with states, actions and rewards. In case of considering multiple agents, MDPs extend to Stochastic Games. Some approaches have used reinforcement learning for patrolling problems [12] or for the illegal rhino poaching problem using Stochastic games [4]. Security games are often modelled as Stackelberg Security games [6], which capture the asymmetry in agents' information about the game. The attacker is assumed to observe the defender's past actions and thus can reason about his strategy, to which the attacker in turn best responds. Related work suggests a framework for asymmetric (Stackelberg) multi-agent reinforcement learning where the roles of defender (leader) and attacker (follower) are fixed [5]. Our work adopts the information asymmetry assumption, providing the control transition model for the *leader*, and allowing leader-strategy-informed best response strategies by attackers. However, we arrive at a more general safe learning approach where the attacks might be very rare and not necessarily adversarial (e.g. random attack). Adversarial attack models may draw on the multi-agent reinforcement learning algorithm Minimax-Q [7] 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.

In brief, we present a novel model-free approach to learn safe and robust policies in systems that are prone to be attacked, or in which some parts of the system may fail. We derive two algorithms that are safe with respect to temporary loss of control, and find them to be empirically robust with respect to estimation errors of control loss probabilities. The proposed family of algorithms opens new ways to effective behaviour in such systems, and can help mitigate some of the security threats using a priori information about the potential nature of the attack or failure.

## 2 GENERAL MODEL

We use the standard Stochastic game formulation defined by a tuple $(n, S, A_1 \dots A_n, R_1 \dots R_n, T)$, where $n$ is the number of agents, $S$ is the state space, $A_i$ is the action space of agent $i$, $R_i(s, a) \rightarrow r_i$ is the reward function of agent $i$ for given state $s$ and action $a$ and $T(s, a) \rightarrow s'$ is the transition function. We also define a joint action space $A$ as $A_1 \cup \dots \cup A_n \in A$ and a joint reward $R$ as $R_1 \cup \dots \cup R_n \in R$. Our agents are assumed to be cooperative with a possibility of external attack or external noise in action selection. In general the $n$ agents can be cooperative, adversarial or mixed. Our method is particularly designed for the case where there is a set of cooperating agents, where some of them can sometimes turn to be non-cooperating (possibly adversarial).

We introduce a new algorithm where we consider several control domains, defining whether the cooperating agents are in control of their actions or someone else is in control and can alter the agents' actions taken. During the interaction with the system there are assumed to be transitions of such control. In the proposed algorithm we assume we know the model of the control transition, which is defined by the control transition function $C$. This function defines who is in control of the system, where we also allow for a partial control. In case of multiple agents, the function $C$ can for example define if and when an agent (and which one) is compromised i.e. not in control of his actions. The control function is defined for a very general case and can express various scenarios of malfunction of an agent or of a malicious attack. In our reinforcement learning method we use function $C$ to define the value function $V(s)$, i.e. value function is a function of the control function as $V(s, C)$.

We start introducing our algorithm from a simple case, where we assume only two entities having control, either the optimal agent itself with policy $\pi$ or an external entity with policy $\mu$, then $C$ determines whether we move between $c_\pi$ and $c_\mu$ at any given moment. We define the control transition function $C$ to be consistent with the transition function $T$ as $C(s, a, \sigma) \rightarrow \sigma$, where $\sigma \in M$ describes who (and how) is in control out of $|M|$ different entities. The model assumes Markovian property, thus $p(\sigma') = p(\sigma'|\sigma)$, $\forall \sigma$. The objective of the external policy $\mu$ is defined in a general way and thus can for example be: minimising value (malicious attack), randomising (e.g. random errors). Based on an assumption about the nature of $\mu$ we want to learn simultaneously a model of the environment and a model of $\mu$ without necessarily observing actual attacks. This means learning a safe policy $\pi$ right from the start.

The simple case is when we need to learn only one Q-value function $Q^\pi$. This is in the case when the not-in-control policy can be defined in terms of our own Q-value function e.g. attacker is minimising our Q-value function. Such an assumption can be explained by the Stackelberg property, where the attacker might be able to observe our past actions and thus know our Q-value function. The Q-value function update based on standard Q-learning is then defined:

$$Q^\pi(s, a) \leftarrow Q^\pi(s, a) + \alpha \left[ r(s, a) + \gamma V^{\tilde{\pi}}(s', C) - Q^\pi(s, a) \right] \quad (1)$$

where the value function in state $s$, given the control transition function $C$, is defined as

$$V^{\tilde{\pi}}(s, C) = \sum_{\sigma \in M} p(\sigma) V^\pi(s, \sigma)$$

where the composite policy $\tilde{\pi} \cdot = \sum_{\sigma \in M} p(\sigma)\sigma(s, a)$, which can be seen as the actual executed policy by the system, including any potential malfunctions or/and attacks. Note that we can learn $Q^\pi$ without actually observing any attack or malfunction.

We now move to a more advanced model, where we still assume only two entities having control but now the not-in-control policy cannot be defined in terms of our Q-value function $Q^\pi$ and instead is based on a different Q-value function $Q^\mu$. We approach this by learning separate Q-functions for both our target policy $\pi$ and the not-in-control (e.g. attacker) policy $\mu$, and using a mix of both Q-functions based on the control transition model (e.g. probability of attack) to evaluate the next state. The Q-update for our safe policy $\pi$ is then defined as:

$$Q^\pi(s, a) \leftarrow Q^\pi(s, a) + \alpha \left[ r(s, a) + \gamma V^{\tilde{\pi}}(s', C) - Q^\pi(s, a) \right] \quad (2)$$

where the value function in state $s$, given the control transition function $C$, is defined as

$$V^{\tilde{\pi}}(s, C) = \sum_{\sigma \in M} p(\sigma) V^\sigma(s, \sigma)$$

where the value function of the composite policy $\tilde{\pi}$ is a weighted sum[1] of several value functions learned using different Q-value functions. Technically, we want to find our robust policy $\pi$ by learning the Q-function $Q^\pi$ while assuming that in the next state $s'$ *any*

[1]In general this can be any operator, not just *sum* (i.e. linear combination), which would allow for more complex scenarios.

policy $\sigma \in M$ can be active with some probability $p(\sigma|s')$, where $\sum_{\sigma \in M} p(\sigma|s') = 1$. This probability is defined by the control transition function $C$. In our example $M = \{\pi, \mu\}$. Thus, at the same time of learning $Q^{\pi}$ we want to learn Q-functions for all policies $\sigma$ we consider. We can learn the value function using the Q-update function of the not-in-control policy $\mu$ from the same experience stream $< s, a, r, s' >$ generated by policy $\pi$ by using importance sampling (or other methods like in Retrace($\lambda$) algorithm [10]) denoted by $c_s$ as:

$$Q^{\mu}(s,a) \leftarrow Q^{\mu}(s,a) + \alpha c_s \left[ r(s,a) + \gamma V^{\tilde{\pi}}(s', C) - Q^{\mu}(s,a) \right] \quad (3)$$

where $c_s$ is for example importance sampling i.e. $c_s = \frac{\mu(a|s)}{\pi(a|s)}$. This gives us our own *estimation* of the Q-functions for all policies $\sigma$, which in our setting would be just our own policy and that of the attacker.

## 3 TWO-AGENT ADVERSARIAL CASE

We now demonstrate the proposed algorithm on a special case of a distributed cooperative two-agent adversarial scenario, where we assume the attack to be intentional and intelligent. This mainly concerns the control transition function $C$ and thus the definition of the value function, where we assume the attack to minimise the possible return (i.e. minimising Q-function). This assumption comes from the concept of Stackelberg attackers who can fully observe the agent past behaviour and therefore know his strategy vector which he can best-respond to (minimising agent's utility). An important aspect is the sparsity of such attacks with the aim not to be too cautious when not needing to be. We assume two cooperating agents in the model with the attacker minimising over the agents and over their action spaces.

The risk of attack is assumed to be known and is expressed by the probability of attack per state, which is defined by the control transition function $C$. These attacks are expected to be fairly rare. In our algorithm we use a parameter $\kappa$ which expresses the risk of attack and allows the algorithm to learn an accordingly safe strategy $\pi$. Thus, we consider several policies $\sigma \in M$, where

$$M = \{\max_{A_1} \max_{A_2} Q^{\pi}, \max_{A_1} \min_{A_2} Q^{\pi}, \min_{A_1} \max_{A_2} Q^{\pi}, \min_{A_1} \min_{A_2} Q^{\pi}\}$$

with $p(\sigma) = (1-\kappa, \frac{\kappa}{2}, \frac{\kappa}{2}, 0)$, representing the situation where only one agent can be attacked at a time (with probability $\frac{\kappa}{2}$). We now present two new algorithms based on standard off-policy and on-policy algorithms.

*Q(κ): Q-learning based.* We present an off-policy type of learning, based on standard Q-learning. We assume that each agent's Q-function is representative of all agents' Q-functions (which makes sense in a cooperative domain with a single reward function). Then in the two agent case we can compute the optimal actions for both players as

$$\langle a_1^{\star}, a_2^{\star} \rangle = \operatorname*{arg\,max}_{a_1 \in A_1, a_2 \in A_2} Q^{\pi}(s, \langle a_1, a_2 \rangle) \quad (4)$$

where we define the value function as

$$\begin{aligned} V^{\tilde{\pi}}(s, C) = (1-\kappa) & Q^{\pi}(s, \langle a_1^{\star}, a_2^{\star} \rangle) \\ & + \frac{\kappa}{2} \min_{a_1 \in A_1} Q^{\pi}(s, \langle a_1, a_2^{\star} \rangle) \\ & + \frac{\kappa}{2} \min_{a_2 \in A_2} Q^{\pi}(s, \langle a_1^{\star}, a_2 \rangle) \end{aligned} \quad (5)$$

Note that for $\kappa = 0$ this algorithm degenerates into standard Q-learning.

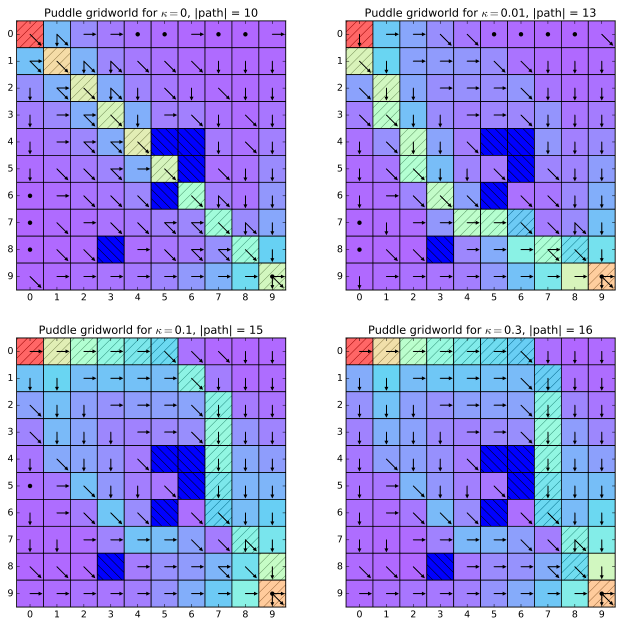
*Expected SARSA(κ): On-policy-mix learning.* We propose an on-policy version based on Expected-SARSA [18]. We define the value function $V^{\tilde{\pi}}(s, C)$ for the Expected SARSA($\kappa$), with the attacker reacting (in expectation) to mixed policy

$$\begin{aligned} V^{\tilde{\pi}}(s, C) = (1-\kappa) & \mathbb{E}_{a_1^{\star}, a_2^{\star} \sim \pi} \left[ Q^{\pi}(s, \langle a_1^{\star}, a_2^{\star} \rangle) \right] \\ & + \frac{\kappa}{2} \min_{a_1 \in A_1} \mathbb{E}_{a_2^{\star} \sim \pi} \left[ Q^{\pi}(s, \langle a_1, a_2^{\star} \rangle) \right] \\ & + \frac{\kappa}{2} \min_{a_2 \in A_2} \mathbb{E}_{a_1^{\star} \sim \pi} \left[ Q^{\pi}(s, \langle a_1^{\star}, a_2 \rangle) \right] \end{aligned} \quad (6)$$

## 4 EXPERIMENTS

We use a version of *Cliff Walking* described by Sutton and Barto [15] for experimental evaluation of our proposed algorithms Q($\kappa$) and Expected SARSA($\kappa$). Our environment is a grid world with puddles, which need to be avoided by the agent, motivated by the independent control transitions in distributed systems. Our algorithm is primarily designed for multi-agent environments, thus in our experimental evaluation we assume two cooperative agents who move in the map according to their joint action. Each agent has two possible actions: stay or move; we distinguish the agents by the direction of their move. The first agent can move horizontally (from left to right) and the second agent can move vertically (from top to bottom). Therefore, the joint action consists of 4 possible actions: stay, move right, move down or move diagonally. The agents start at the left top corner and the game (episode) ends when the agents reach the right bottom corner or step into a puddle. Each move renders negative reward -1 or -1000 for stepping into a puddle. We show the puddle grid world in Figure 1, where the puddles are dark blue.

Firstly, we demonstrate the behaviour of Q($\kappa$) in our grid world in Figure 1. We show learned paths for different levels of parameter $\kappa$. Note that for $\kappa = 0$ our algorithm Q($\kappa$) degenerates into standard Q-learning. We can see that Q-learning learns the shortest possible path, which is, however, very risky due to the close proximity to the puddles. In case of a small perturbation in the strategy (e.g. attack or failure) the system can fall into a puddle, gaining very high negative reward (we use the term reward also for negative numbers, representing *cost*). One can observe how the learned path becomes safer for increasing parameter of $\kappa$. Note that the agents can stay or move only towards the right bottom corner i.e. down or right direction, meaning no returning allowed, which is the reason why even for high values of parameter $\kappa$ the agents do not mind being close to the puddles from the right side of the puddle but trying to avoid the puddle from the left side. The arrows show maximal Q-values in each state, thus one can observe the learned behaviour in each state. The heat-map shows the number of visits to each state.



Puddle gridworld for $\kappa = 0$, |path| = 10
Puddle gridworld for $\kappa = 0.01$, |path| = 13
Puddle gridworld for $\kappa = 0.1$, |path| = 15
Puddle gridworld for $\kappa = 0.3$, |path| = 16

**Figure 1: Puddle gridworld: Two agents need to get from the top left corner to the bottom right corner. Agent A (B) has two possible actions; *stay* or *move right (down)*. The resulting move is a joint action: stay, move right, move down, or move diagonal. There is a reward of $-1$ per time step, and stepping into a puddle (blue squares) results in a reward of $-1000$. The learned path is shown by diagonal stripes (//). The heat-map shows the number of visits per state, and the arrows show the optimal joint action in each state. The results are for Q($\kappa$) with different values of $\kappa$; note that for $\kappa = 0$ the algorithm reduces to standard Q-learning. One can see how the learned path becomes safer with increasing $\kappa$.**

In order to evaluate the proposed algorithm Q($\kappa$) and Expected SARSA($\kappa$) we show reward evolution for varying intensity of attacks. These two algorithms are described in Section 3. An attack means changing one of the two agent's strategy in a malicious way. We assume that only one agent can be attacked at a time. In our experimental setup with two agents, attacking one of them means that the joint action is changed only partially from one of the agents. As described before there are four possible joint actions: stay, move right, move down or move diagonally. In case of attack these joint actions change in following manner; (i) action *stay* to either *move right* or *move down*, (ii) action *move right* to either *stay* or *move diagonally*, (iii) action *move down* to either *stay* or *move diagonally* and (iv) action *move diagonally* to either *move right* or *move down*.

We compare the algorithms against off-policy and on-policy baselines: Q-learning, SARSA and Expected SARSA. We fix the exploration parameter for all the algorithms to $\epsilon = 0.1$. We train every method for 50000 episodes, then test it for 50000 episodes with attacks and average over 100 trial runs; we also show 95% confidence intervals. We assume that we know the probability of attack for the learning phase (i.e. knowing the control transition function $C$), thus we set the parameter $\kappa$ equal to the probability of attack. We show this in Figure 2, where the left figure depicts probabilities of attack in range $(0.001, 0.2)$ and the right figure in range $(0.1, 0.9)$. The curves are plotted with a 95% confidence intervals. We can see that Q($\kappa$) performs the best in both figures, Expected SARSA($\kappa$) also beats all the baselines. The bottom figures in Figure 2 show how many times the algorithm stepped into a puddle - rendering a reward -1000. We can see that for very low probabilities of attack (i.e. 0.001) the algorithm Q($\kappa$) steps into a puddle relatively more often than some of the baselines, however the performance in terms of reward is better (top figures), which means that the algorithm does not take unnecessarily safe paths in case of very rare attacks. The dotted/dashed lines, crosses 'x' and pluses '+' in respective colours show the algorithms' performance in case of no attacks i.e. the rewards of learned path, thus we can see how *safe* a path an algorithm learns.

To test our proposed algorithms we perform a robustness analysis, where we assume we might not know the probabilities of attack and thus are not able to correctly set the $\kappa$ parameter. We show a reward evolution for fixed parameter $\kappa$ in Figure 3, $\kappa = 0.01$ in the left figure and $\kappa = 0.1$ in the right figure. One can see that even for the cases where parameter $\kappa$ is not equal to probability of attack i.e. assuming wrong intensity of attack, algorithm Q($\kappa$) performs the best (up to probability of attack 0.05 for $\kappa = 0.01$ and for probability of attack equal to 0.1 or 0.2 for $\kappa = 0.1$). Algorithm Expected SARSA($\kappa$) also outperforms the baselines algorithms in those cases. This robustness evaluation confirms strong properties of our proposed algorithms. We conclude that the proposed algorithms are robust to the cases of inaccurate estimate of probability of attack (within some bounds).

Further testing of the proposed algorithms include convergence analysis in Figure 4. We trained the proposed algorithms and the baselines on low number of episodes i.e. (in range (1000, 10000)), then test it on 50000 episodes of fixed probability of attack and averaged over 100 trials. The right subfigure in Figure 4 shows the reward evolution for different number of training episodes for fixed probability of attack 0.01 and the left subfigure for probability of attack 0.1. One can see that off-policies (Q-learning and Q($\kappa$)) converge slower compared to the on-policies (SARSA, Expected SARSA and Expected SARSA($\kappa$)). However, the proposed algorithm Q($\kappa$) and Expected SARSA($\kappa$) eventually converge to superior behaviour compared to the baselines as demonstrated above. Note, the instability of SARSA, where the confidence interval is very wide. On the other hand the proposed algorithms are very stable i.e. narrow confidence intervals.

## 5 DISCUSSION

The proposed type of algorithms presents a new way of learning robust policies in presence of rare malfunctions or attacks. Our experimental evaluation shows promising results where our method was able to beat standard Q-learning, SARSA and Expected SARSA algorithms. This is a first experimental evaluation done on a simple gridworld inspired by Cliff walking presented by Sutton and Barto [15]. We showed an example with two agents with a possibility of rare attacks on one of those two agents, causing losing control of executed action. While the extended setting of losing control can be seen as a new Markov game, we argue a) that this class of

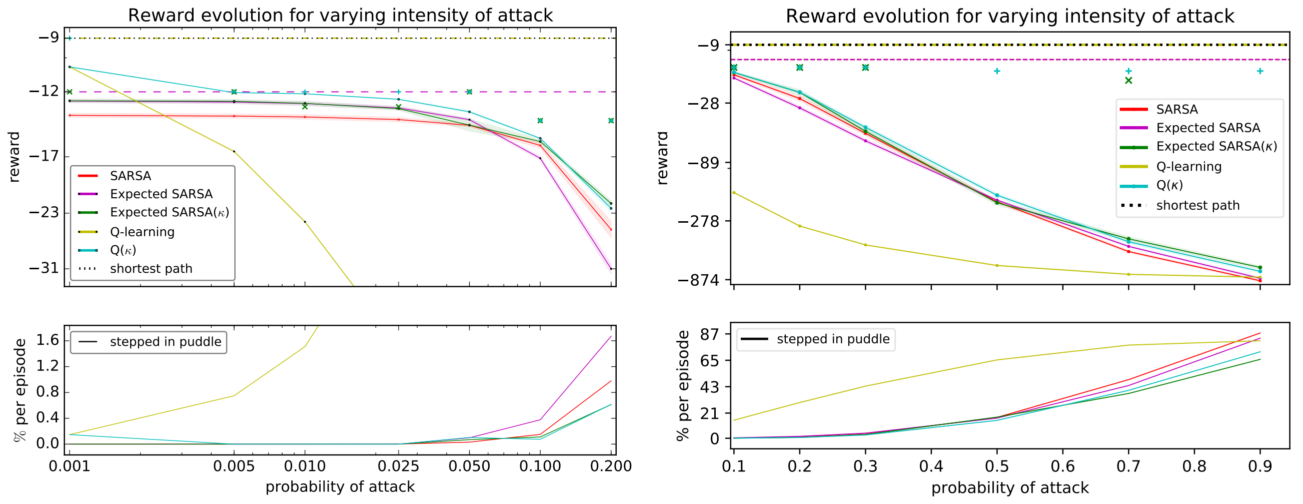


**Figure 2: Reward evolution: Comparing algorithms Q($\kappa$) and Expected SARSA($\kappa$) with baselines (SARSA, Expected SARSA and Q-learning). The dotted/dashed lines and '+' and 'x' markers in respective colours are the rewards of learned path (i.e. no attack). The black dashed line is the reward of the shortest path. Note that we train Q($\kappa$) and Expected SARSA($\kappa$) with $\kappa$ equal to probability of attack. All the shown algorithm have fixed exploration $\epsilon = 0.1$.**

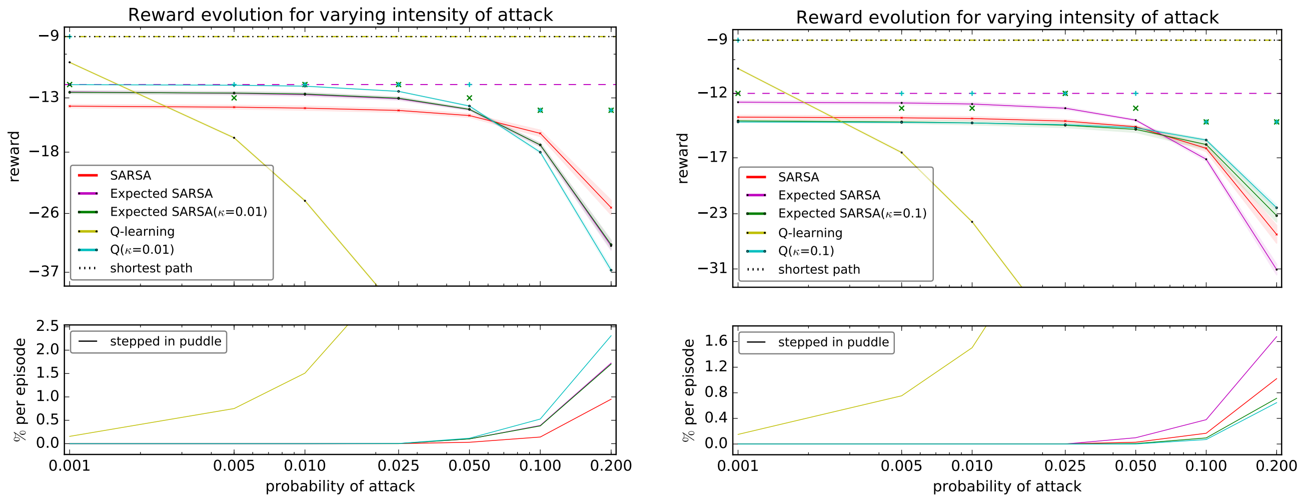
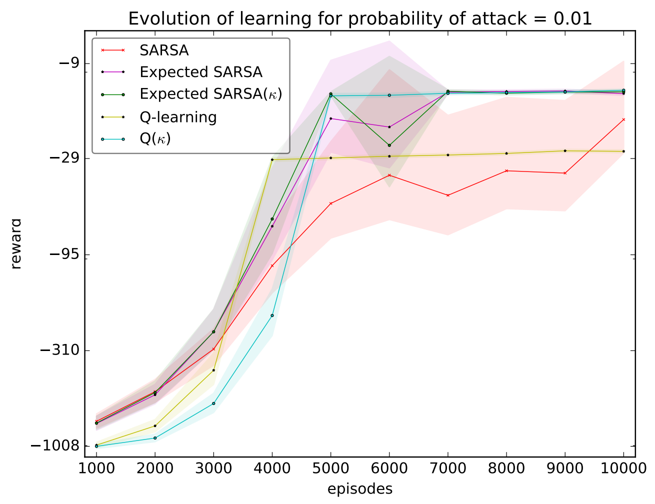


**Figure 3: Robustness of Q($\kappa$) and Expected SARSA($\kappa$): Left figure shows reward evolution for fixed $\kappa = 0.01$ and right figure for fixed $\kappa = 0.1$. We can see that even when parameter $\kappa$ and probability of attack are different the performance is still very good. For $\kappa = 0.01$ we obtain superior performance of Q($\kappa$) (and also Expected SARSA($\kappa$)) for probabilities of attack up to 0.05, for $\kappa = 0.1$ the proposed algorithms are superior for probabilities of attack 0.1 and 0.2.**

games is relevant and useful in applications, and b) our formulation based on a control loss model provides a solution that exploits the identical action space (which model-free approaches applied to the extended game would not).

In future work we will extend the experimental evaluation to new and more complex environments, where we hope to further demonstrate the performance of the proposed methods and analyse their behaviour deeper. We aim to show our algorithms over-performing SARSA by a larger margin. An example of more daring environment is a narrow passage walking where the agents control different directions with different intensity and need to learn to coordinate to walk safely through this passage.

As a next step we want to extend the method to the decentralised multi-agent case, where several agents learn to cooperate while



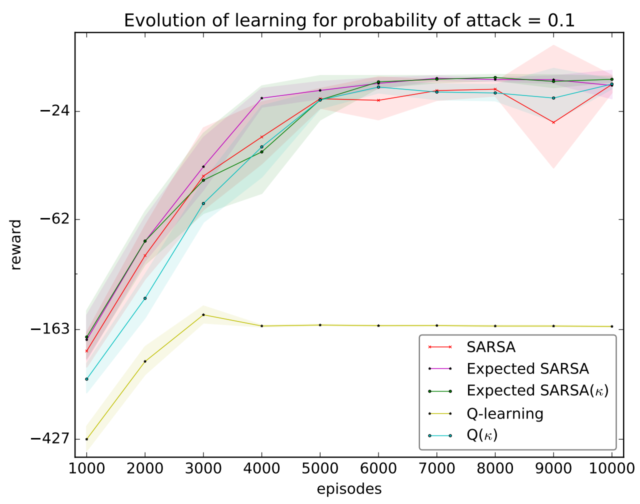


**Figure 4: Convergence: We compare the convergence of our proposed algorithms Q($\kappa$) and Expected SARSA($\kappa$) with several baselines. On x-axis there is the number of episodes the algorithms are trained for. After training the algorithms are tested in 50000 testing episodes with probability of attack 0.01 (left figure) or 0.1 (right figure). One can see that off-policies converge slower than on-policies, which is to be expected. We can also observe the stability of learning by looking at the 95% confidence intervals, e.g. SARSA is very unstable with very wide confidence interval.**

being robust to losing control of actions taken. Such a setting resembles independent Q-learning, which we plan to compare our approach with. Furthermore, we could assume several agents being attacked or malfunctioned with different intensity. This would be expressed by a more complex control transition function with potentially more parameters depending for example on state or time. State-dependent control transition function is motivated by the fact that in practice some states might be critical and would require an extremely safe policy. Continuing in that direction, the control transition function could also be time dependent, where some time steps are more prone to attack or malfunction than others. These extensions would narrow the reality gap and would allow for learning more complex policies, where we believe our approach could bring new insights into learning robust policies in complex environments. For a very large domains we also aim to combine our method with some modern and powerful function approximation techniques such as deep neural networks.

Our proposed method can be closely linked or even combined with some of the state-of-the-art reinforcement learning methods. We already mentioned Retrace($\lambda$) [10], which can be combined with our approach in a multi-step Q-update for robust learning in environments where we might not be always in control of our actions. Promising extension of our model would be to combine it with Q($\sigma$) [1] to allow for mixed multi-step updates. Note that the parameter $\sigma$ in this algorithm can also be time- or state-dependent similarly to the proposed extensions of the control transition function in our model. Combining our model with Q($\sigma$) algorithm shows new ways how to learn robust policies against more complex attacks e.g. multi-step attacks or assuming a strategic attacker deciding on when to attack. Another interesting extension would be to model the control transition function similar to the options framework [2, 16], 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.

## 6 CONCLUSION

In this work we presented a new method based on standard reinforcement learning algorithms, which can learn robust policies in systems with potential attacks or malfunctions. The proposed framework provides robustness against a chosen stochastic control transition model, which describes the probability of attack or malfunction of every part of the studied system. This control transition function is assumed to be a chosen (and thus known) robustness target by the system designer, which is justified in many real-world domains, where expertise is available on the type of expected attack or potential malfunctions that may become critical to system performance. Our preliminary experiments provide promising results, and given the flexibility of the framework and its ability to model attacks with temporal and state-space structure, it lends itself to being applicable to a wide range of real-world scenarios.

## ACKNOWLEDGMENTS

This research was partially funded through the ERA-Net Smart Grids Plus project Grid-Friends, with support from the European Union's Horizon 2020 research and innovation programme.

## REFERENCES

[1] Kristopher De Asis, J. Hernandez-Garcia, G. Holland, and Richard Sutton. 2018. Multi-Step Reinforcement Learning: A Unifying Algorithm. In *AAAI Conference on Artificial Intelligence.*

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The Option-Critic Architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence.* 1726–1734.

[3] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

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

[5] Ville Könönen. 2004. Asymmetric multiagent reinforcement learning. *Web Intelligence and Agent Systems: An international journal* 2, 2 (2004), 105–121.

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

[7] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings*. Elsevier, 157–163.

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

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

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

[11] James Pita, Manish Jain, Janusz Marecki, Fernando Ordonez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. 2008. 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*, Vol. 3. 1805–1812.

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

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

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

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

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

[17] Ministerie van Economische Zaken en Klimaat. 2018. Regeling van de Minister van Economische Zaken en Klimaat van 8 maart 2018, nr. WJZ/18026207, tot wijziging van de Regeling nationale EZ-subsidies en de Regeling openstelling EZK- en LNV-subsidies 2018 in verband met de openstelling van de subsidiemodules inzake Topsector energieprojecten en enkele wijzigingen ervan. *Staatscourant* 14209 (2018).

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

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

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