Deep Reinforcement Learning for Green Security Game with Online Information

Lantao Yu[1]*, Yi Wu[2], Rohit Singh[3], Lucas Joppa[4], Fei Fang[5]
[1]Shanghai Jiao Tong University, [2]University of California, Berkeley, [3]World Wild Fund for Nature
[4] Microsoft Research, [5] Carnegie Mellon University

Abstract

Motivated by the urgent need in green security domains such as protecting endangered wildlife from poaching and preventing illegal logging, researchers have proposed game theoretic models to optimize patrols conducted by law enforcement agencies. Despite the efforts, online information and online interactions (e.g., patrollers chasing the poachers by following their footprints) have been neglected in previous game models and solutions. Our research aims at providing a more practical solution for the complex real-world green security problems by empowering security games with deep reinforcement learning. Specifically, we propose a novel game model which incorporates the vital element of online information and provide a discussion of possible solutions as well as promising future research directions based on game theory and deep reinforcement learning.

Introduction and Research Problem

Game theory has become a well-established paradigm for addressing complex resource allocation and patrolling problems in security and sustainability domains. Models and algorithms have been proposed and studied extensively in the past decade, forming the area of "security game" (Tambe 2011). More recently, machine learning based models have been used to predict adversarial behaviors in green security domains such as wildlife poaching, and game-theoretic solutions built upon the learned behavioral models have been proposed (Xu et al. 2017; Gholami et al. 2017; Kar et al. 2017).

Despite the efforts, a key element, online information, has been neglected in previous game models. For example, a well-trained ranger should be able to use the online information revealed by the traces left by the poacher (e.g., footprints, tree marks) to make flexible patrolling decisions rather than stick to the premeditated patrol routes. Thus there is no doubt that online information received during the interactions between the players plays an important role in the decision-making process and how to incorporate such online information into the solutions remains to be disclosed.

However, incorporating online information into green security games leads to significant complexity, inevitably resulting in games with sequential moves and imperfect information. This makes traditional mathematical programming-based approaches for computing the equilibrium of the game intractable. On the other hand, reinforcement learning (RL) (Sutton and Barto 1998) algorithms are designed to exploit online information. RL employs a goal-oriented learning scheme where the agent learns to maximize its long-term cumulative reward by sequentially interacting with the environment. Recently, by employing the modeling power of deep learning, reinforcement learning has been successfully used on a wide variety of tasks, including playing the games of Atari (Mnih et al. 2015) and Go (Silver et al. 2016), robotic manipulation (Gu et al. 2016) and sequential data generation (Yu et al. 2017). Furthermore, researchers have generalized single-agent RL methods to the multi-agent systems where multiple agents coexist and interact with each other (Busoniu, Babuska, and De Schutter 2008).

Thus in order to provide a more practical solution for the complex real-world security problems, in this paper we propose a novel game-theoretic model, which incorporates the vital online information that has been commonly neglected by literature and provide a discussion of potential algorithms that combine deep reinforcement learning and game theory to approximately compute equilibrium strategies in a complicated spatio-temporal setting with online interactions. In this paper, we illustrate our model and algorithm in the domain of protecting wildlife from poaching but note that the proposed solutions can be applied to other green security domains such as protecting the forest from illegal logging and protecting fisheries from overfishing.

Related Work

Stackelberg Security Games (SSGs) have been studied extensively and successfully deployed in various security domains (Tambe 2011; Pita et al. 2008; An et al. 2011; Fang, Jiang, and Tambe 2013; Fang et al. 2016). Most of the work in SSGs consider implicitly or explicitly a *normal-form game* represented by payoff matrices, where rows correspond to pure strategies of one player, columns correspond to pure strategies of the other player, and payoff values in the matrices represent the utilities for each joint action taken by the players. Each player tries to maximize their own expected utility by finding a pure or a mixed strategy, i.e., a probability distribution over the pure strategy set. However,

*The work was done while L. Yu interned at CMU.

one player (the attacker) may choose his strategy after observing the other player's (the defender's) strategy and try to best respond to it. When the game is zero-sum, the commonly used solution concepts including Nash Equilibrium, Minimax, Maximin, and Stackelberg Equilibrium coincide, and one can use linear programming (Adler 2013), fictitious play (Brown 1951) and regret minimization (Blum and Monsour 2007) to find the solution. To provide a scalable solution for such games and represent the optimal strategy with a small support set (i.e., only a small number of pure strategies are chosen with non-zero probability), (McMahan, Gordon, and Blum 2003) proposed column/constraint generation techniques, also known as *double oracle algorithm*. This algorithm repeatedly computes the equilibrium strategies of the restricted subgame that only involves a subset of all possible pure strategies for each player, computes a pure strategy that is the best response to the equilibrium strategy of the opponent found previously, and add the new strategies to the restricted game. Double Oracle algorithm is guaranteed to converge to a Nash equilibrium (NE) in games with finite actions. However, in the worst case, the algorithm has to enumerate all pure strategies and solves the original game, such as in Rock-Paper-Scissors. In extensive-form games such as the ones we propose in this paper, the number of pure strategies can be huge when converted to a normal-form game and optimal solution may need a large support set that can hardly be represented efficiently. As a natural generalization to the double oracle algorithm, (Lanctot et al. 2017) proposed policy-space response oracles (PSRO) algorithm, where the pure strategy or action of each player is generalized to a parametrized policy, and any reinforcement learning algorithm, such as deep Q-Learning (Mnih et al. 2015), can be used to solve the best response policy in each iteration. To address the challenge that it can take a very long time for an RL algorithm to find a good response policy as much of the basic behavior has to be relearned from scratch, Lanctot et al. proposed to use deep cognitive hierarchies (DCH) to approximate PSRO using a parallel and asynchronized learning approach with a fixed number of iterations being considered, and thus trades away solution quality for efficiency. This approach loses the convergence guarantee of Double Oracle algorithm but provides empirically good solutions in two example games. In this paper, we use a similar framework of PSRO, with concrete implementation for our proposed security game model.

Spatio-Temporal Security Game with Online Information

Game Model

As shown in Figure 1, the basic environment of the proposed game is based on a grid world. Each grid represents a unique geographic area with terrain features $\phi_{i,j}$ such as animal density, elevation, slope, etc. For simplicity, in our game setting, we choose the most representative feature, the animal density, as $\phi_{i,j}$. There are two players (which can be easily scaled to multiple players for each role) in this game, i.e., the patroller and the poacher. At the beginning of each round, the poacher chooses one out of the four corners as the entry point, and the patroller always starts from the center grid of the world, i.e., the patrol post. The poacher sets off with a limited number of destructive tools, i.e., snares, and tries to use them to catch as many animals as possible, which will be embodied in the reward structure. At each time step t, the patroller takes an action selected from its action space $\mathcal{A}^{pa}$: {*up, down, right, left, stand still*}; simultaneously, the poacher takes an action a_t^{po} selected from its action space $\mathcal{A}^{po}$: {*up, down, right, left, stand still*} $\times$ {*place snare, not place snare*}. Note that since any player can choose to stand still at any time step, we do not make any assumptions about the order of entering the world for both players.

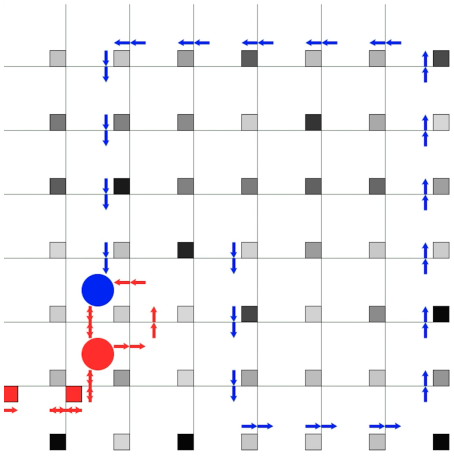


Figure 1: Illustration of the combat poaching security game with online information. The red dot and blue dot represent the poacher and patroller respectively. The red arrows and blue arrows represent their corresponding footprints. The color darkness of the square on the lower right corner of each cell indicates the animal density. The red squares on the upper left corner of some cells represent the destructive tools placed by the poachers. Note that each player only has a local observation, which means they can only observe the game state for their current grid, rather than the game state for the whole world.

As shown in Figure 1, each player will leave traces (i.e., footprints) as he moves through an area. These traces can be used by the player to get a sense of his opponent's track and strategy, which also play an important role in the real world. It is nontrivial for human to explicitly program the optimal action rule based on the observed traces since in some scenarios simply following the footprints (for the patroller) or escaping from the footprints (for the poacher) is not the best strategy. For example, a poacher may not be deterred by the patroller's footprint if he knows the animal density in that direction is very high and the patroller may have already left. Furthermore, it is also unclear what is the best strategy when multiple footprints exist in the same grid.

In this game, each player has only *local observation*, which means they can only observe the game state in the current grid rather the full game state. In the real world, both the poacher and patroller typically have a limited view due to the dense vegetation and complex terrain. We assume the

players have unlimited memory and can keep a record of the observations since the beginning of each game.

To accurately reflect the goal of the poacher and patroller, we design the reward structure as follows. Suppose a snare has been placed in a grid with coordinate (i, j), at each time step the environment will determine whether or not an animal is caught by that snare based on a function $P(\phi_{i,j})$, which takes the animal density of that area as input and output the probability of an animal being caught. After catching an animal, the snare will be removed from the system. Since the poacher tries to catch as many animals as possible by strategically placing the snares, and in the meantime avoid encountering the patroller, he will receive a positive reward r^{po}_{animal} when a certain snare catches an animal and a negative reward (i.e., penalty) r^{po}_{catch} when encountering the patroller. Since the patroller tries to finding the snares before they cause any damage and catch the poacher to stop them from placing more snares, he will receive a positive reward r^{pa}_{catch} when finding a poacher and a negative reward r^{pa}_{animal} when a snare catches an animal. For generality, here we do not require the game to be a zero-sum game ($r^{po}_{catch} + r^{pa}_{catch} = 0, r^{po}_{animal} + r^{pa}_{animal} = 0$). The game ends when the patroller successfully find the poacher and all the snares or the maximum time step has been reached.

A player's pure strategy or policy in this game (we use policy and strategy interchangeably in this paper) is a deterministic mapping from his observation and action history to his action space. His final payoff is the cumulative reward minus cumulative penalty. A player can employ a mixed policy, which is a probability distribution over the pure strategies.

Most work in security game literature assumes that the attacker can observe the defender's mixed policy before choosing his own policy. However, sometimes the attackers in green security domains may not be intelligent enough to adapt to the defender's strategy. In this paper, we focus on two extreme cases and discuss the solution approaches for finding the optimal defender strategy under these settings. The first case is when the attacker is reactive but not adaptive, i.e., the attacker is not able to adapt to the defender's strategy, but instead chooses a fixed heuristic policy only based on the features of the environment and the online information. In the chosen policy, the attacker may still react to online information, e.g., the poacher may be deterred by the footprints of the patroller, but such reaction does not take into account the defender's mixed strategy, from which additional information can be derived, e.g., the probability that the defender will return to an area he has been to before. In this case, we aim to find the optimal defender strategy against a known reactive attacker. The second case is the attacker is highly adaptive and best responds to the defender's strategy. In this case, the solution concept we use is Strong Stackelberg Equilibrium (SSE), and we aim to find the defender's strategy in SSE, which is also the defender's optimal strategy assuming a best-responding attacker.

Defender Policy Representation

Before discussing the computation of equilibrium strategies for the players, we first consider how to efficiently represent a player's pure or mixed strategies. Given the complexity of the game, finding the optimal policy is challenging. In fact, the memory needed for just representing a defender's pure strategy in the form of a detailed mapping from full observation and action history to action grows exponentially in the size of the game. Thus representing a player's mixed strategy in its naive form (a probability distribution over the pure strategies) can be intractable. Therefore, instead of using a naive form, we consider several compact representations of the defender's pure or mixed strategy using the key elements of the history and the environment and look for the defender's optimal strategy in the restricted space.

Heuristic Rule-Based Policy The first model family of defender policy we consider is based on heuristic rules. Suppose the current coordinate of the patroller is (m, n) and the maximum coordinate on the map is (M, N), then we can define the average animal density for the *up* direction as $\frac{1}{(m-1)\cdot N}\sum_{0\le i<m, 0\le j\le N}\mathcal{A}_{i,j}$, where $\mathcal{A}_{i,j}$ is the animal density for area with coordinate (i, j). Similarly, we can define the average animal density for all the other directions. For simplicity, we will use an integer $k \in \{1, \dots, 5\}$ to denote one of the five directions (the fifth "direction" is for the action "stand still") and we can get an average animal density vector $A \in \mathbb{R}^{+5}$ (A_5 is the animal density of the current location). Another important factor that should be taken into consideration is the observed footprints. We use vectors $I \in \{0, 1\}^5$ and $O \in \{0, 1\}^5$ to represent the footprints states, where each dimension I_k (or O_k) is a binary variable, indicating whether or not there is an entering (or leaving) footprint from that direction (for the fifth "stand still" direction, $I_5 = O_5 = 0$). Now we can define the heuristic defender policy as

$$\pi_{pa}(a^{pa}_t = k|s^{pa}_t) = \frac{\exp(w_a \cdot A_k + w_i \cdot I_k + w_o \cdot O_k)}{\sum_z \exp(w_a \cdot A_z + w_i \cdot I_z + w_o \cdot O_z)} \quad (1)$$

where w_a, w_i and w_o are parameters for the average animal density, enter footprints and leaving footprints respectively.

Deep Neural Network Based Policy The second class of defender policy is represented by neural networks, and we will use reinforcement learning to find an empirically optimal strategy. From the perspective of reinforcement learning, at each time step t, the state of the poacher s^{po}_t and patroller s^{pa}_t are comprised by their memory of the opponent's observed footprints, current position coordinates, the corresponding terrain information, i.e., animal density and the normalized game time. The policy $\pi_{pa}(a^{pa}_t|s^{pa}_t)$ and $\pi_{po}(a^{po}_t|s^{po}_t)$ for the patroller and poacher respectively will be a mapping from their state space to their action space.

Since the game state has strong spatial patterns, here we employ a convolutional neural network for implementing the learning policy, which takes as input a 3-D tensor, with the same width and height as the grid world and each channel representing different features. Specifically, the first eight channels are binary values indicating the existence of each kind of footprints ({four directions} $\times$ {entering or leaving}); the ninth channel is the animal density; the tenth channel is one-of-K encoding, indicating the agent's current loca-

tion; the eleventh channel is the normalized time step, which is the same for all grids. In our experiments, the defender neural network takes a state representation of size 7×7 as input. The first hidden layer is a convolutional layer with 16 filters of size 4×4 and strides 1×1. The second layer is a convolutional layer with 32 filters of size 2×2 and strides 2×2. Each hidden layer is followed by a $relu$ nonlinear transformation and a max-pooling layer. The output layer is a fully-connected dense layer which transforms the hidden representation of the state to the final policy. If the neural network is an action-value network, then each output dimension represents the Q-value for each action, and the neural network corresponds to a pure strategy for the defender where the defender takes the action with the highest Q-value generated by the network, with the given state as input. If the neural network is a stochastic policy network, then each output dimension represents the probability of an action $\pi_{pa}(a_t^{pa}|s_t^{pa})$, and the neural network corresponds to a mixed strategy for the defender where the probability of taking each pure strategy is the joint probability of taking the chosen action in each time step. While some pure or mixed defender strategies cannot be captured by the neural network based representation, the strong expressiveness makes it a memory-efficient alternative. Furthermore, as detailed in later sections, the NN-based representation combined with RL algorithms leads to defender strategies that perform well empirically. And we expect that the current feedforward neural network implementation of the policy can further be strengthed by incorporating recurrent neural networks, such as Long Short-Term Memory networks (Hausknecht and Stone 2015).

Attacker Policy Representation

Likewise, there are two ways to represent the attacker's policy. Different from the defender's policy, the attacker's policy is two-fold. At each time step, he should decide whether to put a snare in current location and which direction to move. For the movement decision, we again use three parameters w_a, w_i and w_o to control different factors. For the placing snares decision, since a higher animal density means a higher probability of catching an animal, we define the probability of placing a snare in grid (m, n) as $\frac{\exp(A_{m,n}/\tau)}{\sum_i \sum_j \exp(A_{i,j}/\tau)}$, where τ is a temperature parameter.

Like the defender's policy, we can use a similar convolutional neural network structure to represent the attacker's strategy. The only difference is that we should change the output layer since the action space combines both the movement decisions and placing snare decisions.

Finding Optimal Defender Strategy Against Reactive Attacker

In this section, we consider the case where the attacker is reactive and find optimal defender strategies in the restricted defender strategy space with the compact representations. We focus on the neural network based policy and propose to use deep reinforcement learning based algorithms to find the empirical optimal defender strategy against a reactive attacker with known parameters[1].

With deep reinforcement learning, the intelligence of the agent can evolve through the interactions between the agent and the environment. Basically, there are two categories of RL methods, i.e. the value based methods and the policy based methods. For value based methods, one of the most effective and efficient methods is Deep Q-Learning (DQN) (Mnih et al. 2015), which uses an off-policy replay buffer to keep all the transitions (s_t, a_t, r_t, s_{t+1}) and update the Q-network as

$$Q(s_t, a_t) \leftarrow r_t + \gamma \max_{a'} Q^{\text{target}}(s_{t+1}, a') \tag{2}$$

To solve the proposed game, which involves a highly dynamic environment, we employ the double DQN methods (Van Hasselt, Guez, and Silver 2016) to improve the stability of training:

$$Q(s_t, a_t) \leftarrow r_t + \gamma Q^{\text{target}}(s_{t+1}, \arg\max_{a'} Q(s_{t+1}, a')) \tag{3}$$

For policy based methods, i.e., the stochastic policy representation, we directly optimize the θ-parametrized policy with the policy gradient theorem (Sutton et al. 2000)

$$\begin{aligned}\nabla_\theta \mathbb{E}_\tau[R] &= \mathbb{E}_\tau[\sum_{t=0}^{T} \nabla_\theta \log \pi(a_t|s_t, \theta) \cdot (Q^\pi(s_t, a_t) - V^\pi(s_t))] \\ &= \mathbb{E}_\tau[\sum_{t=0}^{T} \nabla_\theta \log \pi(a_t|s_t, \theta) \cdot A^\pi(s_t, a_t)] \end{aligned} \tag{4}$$

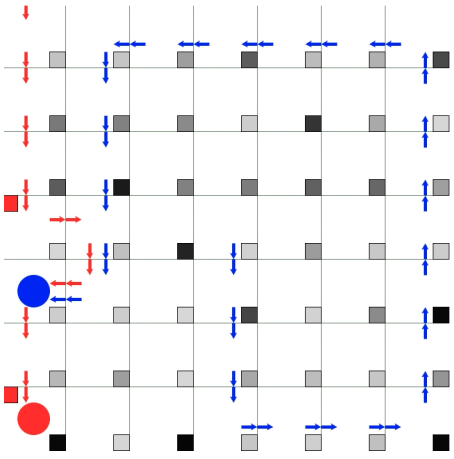
where the advantage function $A^\pi(s_t, a_t)$ can either be approximated by Monte Carlo methods or neural networks. In our experiments, for a lower variance, we use a neural network to approximate the state-value function $V^\pi(s)$, and use $r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t)$ as an unbiased estimation of the advantage function $A^\pi(s_t, a_t)$, i.e. the actor-critic algorithm (Konda and Tsitsiklis 2000).

Experiments and Discussions

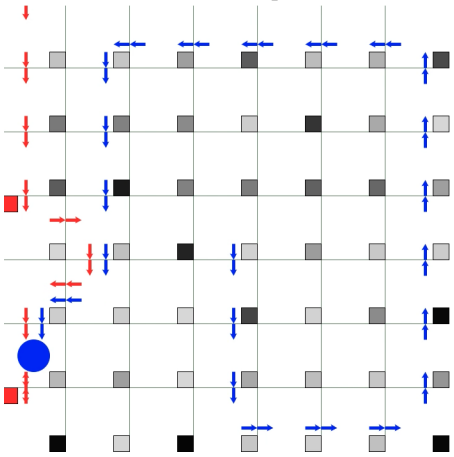
In this section, we will discuss some initial experiment results. First, we want to examine the effectiveness of different defender policies against a reactive heuristic rule-based attacker policy. In the first set of experiments, the patroller is implemented with heuristic rules, policy gradient methods and deep Q-learning and fight against a fixed heuristic poacher, whose parameters w_a, w_i and w_o are set based on suggestions from domain experts.

In the experiments, we observe that with a granularity of 0.1 and searching range from -5 to +5, searching optimal parameters for heuristic methods requires lots of simulations and can be very time-consuming. On the contrary, DQN methods can find the optimal strategy very quickly and efficiently, through the interactions between the agent and environment. The expected utility for the heuristic patroller with grid search for the optimal parameters is 2.97 and for the DQN-based patroller is 7.23, from which we can

[1]For heuristic rule-based policy, we simply use grid search to find the parameters of the empirically optimal defender strategy in the restricted space.



(a) At timestep 22



(b) At timestep 23

Figure 2: Screenshots of the game, where the poacher strategy is heuristic rule based and the patroller strategy is DQN based.

tell that when faced with a non-adaptive opponent, DQN methods can effectively find a strategy that performs very well. But when we train a DQN-based poacher against a fixed DQN-based patroller, we observe that the DQN-based patroller can easily be exploited by an adaptive opponent. This is because a DQN policy is a deterministic policy since given a certain state, we always choose the action with the maximum Q-value. In a security game, any pure strategy can be easily exploited by the opponent, which reveals the challenge of the proposed game. Then a natural solution should be that instead of learning an action-value function and taking actions greedily, we can directly optimize a stochastic policy. However, in our experiments, we find that in such a highly dynamic game with imperfect information, it is very difficult to effectively train a stochastic policy with policy gradient methods. One possible reason is that different from the traditional RL problems, in this game the state is only a very limited local observation and it is not informative enough to learn the optimal action distribution. Before seeing any footprints, we require the agent to perform a long sequence of random guesses and as shown in Figure 2, what makes DQN patroller so effective is that DQN-based patroller learns to sweep around the border until he finds any footprints to follow. While for the policy gradient methods, the optimal action distribution for any state without footprints is observed to be close to a random uniform distribution. In other words, PG methods learn an average of all possible sweeping routes, and as a result, it will be not quite effective in such a dynamic imperfect information game.

We further investigate how to find the optimal defender strategy in a Stackelberg setting, i.e., when the attacker is adaptive and best responds to the defender's mixed strategy. When the game is zero-sum (or fixed-sum), the SSE strategy is also the NE strategy for the defender. One possible way of computing NE in such a game is using Double Oracle algorithm (McMahan, Gordon, and Blum 2003; Jain et al. 2011; Jain, Conitzer, and Tambe 2013). Based on the above experiment, we know that deep Q-learning is good at approximating the best response strategy given a fixed opponent, meaning that it can serve as the key building block for the Double Oracle algorithm. Specifically, we can start with a small number of pure strategies for both players. At each iteration, we first get the payoff matrix for current pure strategies with simulations and compute the Nash Equilibrium for the current game matrix. Then we fix the mixed strategy for one player and compute the best response strategy of another player with deep Q-learning. After getting a new best response policy for each player, we evaluate the value of the new payoff matrix and repeat the procedure until convergence. In our proposed game model, the game may not be zero-sum, but the two players' payoffs are negatively correlated. Despite the lack of theoretical guarantee, we adopt the DO framework, start with one pure strategy for each player represented by a randomly initialized DQN, and then use our proposed deep Q-learning approach to approximate the best response strategy for each player. We test the expected utility of the defender's strategy we got from the framework against a best-responding attacker experimentally. Table 1 shows the preliminary results. The second and third column shows the expected utility for the defender and attacker respectively, where the defender uses the NE strategy computed in the restricted game and the attacker uses the best response strategy against the defender's strategy (approximated by a trained DQN policy). The first observation is that the best defender strategy of the first four iterations is found in iteration 3, as measured by the value in the second column, and the solution quality is much higher than the first iteration where the defender is using a random DQN policy. This means using double oracle with deep Q-learning can be a promising way to find a good mixed strategy for the defender for such a complex game setting. Our second observation is that double oracle has not converged within four iterations as expected. In fact, the values fluctuate a lot, leading to the concern that limiting the number of iterations in Double Oracle to be small may be problematic.

Future Research Directions

Since training a best response DQN policy against the new opponent is often time-consuming, we can only run double oracle algorithm for a few iterations given limited computa-

Table 1: Preliminary results of double oracle algorithm combined with Deep Q-learning. The Expected Utilities (EU) in the table are the simulation results when the defender uses the NE strategy computed in the restricted game, and the attacker uses the best response strategy against the defender's strategy (approximated by a trained DQN policy).

Iteration k	Defender EU	Attacker EU
1	-5.8775	5.9038
2	-4.3308	5.1455
3	-1.8848	2.6240
4	-3.7824	3.9441

tion resources and training time. Even with the DCH framework proposed by Lanctot et al., the limited number of iterations still present as a major bottleneck. This is one major challenge for future research.

Also, we argue that the key online information and timing aspects should not be neglected by the security game community. Integrating deep reinforcement learning with classical game-theoretic approaches can be a promising way of finding solutions to such complex problems. Alternative frameworks need to be proposed to find SSEs instead of NEs efficiently when the game is not near zero-sum.

References

Adler, I. 2013. The equivalence of linear programs and zero-sum games. *International Journal of Game Theory* 42(1):165–177.

An, B.; Pita, J.; Shieh, E.; Tambe, M.; Kiekintveld, C.; and Marecki, J. 2011. Guards and protect: Next generation applications of security games. *ACM SIGecom Exchanges* 10(1):31–34.

Blum, A., and Monsour, Y. 2007. Learning, regret minimization, and equilibria.

Brown, G. W. 1951. Iterative solution of games by fictitious play. *Activity analysis of production and allocation* 13(1):374–376.

Busoniu, L.; Babuska, R.; and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*.

Fang, F.; Nguyen, T. H.; Pickles, R.; Lam, W. Y.; Clements, G. R.; An, B.; Singh, A.; Tambe, M.; and Lemieux, A. 2016. Deploying paws: Field optimization of the protection assistant for wildlife security. In *AAAI*, 3966–3973.

Fang, F.; Jiang, A. X.; and Tambe, M. 2013. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 957–964. International Foundation for Autonomous Agents and Multiagent Systems.

Gholami, S.; Ford, B.; Fang, F.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Nsubaga, M.; and Mabonga, J. 2017. Taking it for a test drive: a hybrid spatio-temporal model for wildlife poaching prediction evaluated through a controlled field test. In *Proceedings of the European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases, ECML PKDD*.

Gu, S.; Holly, E.; Lillicrap, T.; and Levine, S. 2016. Deep reinforcement learning for robotic manipulation. *ArXiv e-prints*.

Hausknecht, M., and Stone, P. 2015. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*.

Jain, M.; Korzhyk, D.; Vaněk, O.; Conitzer, V.; Pěchouček, M.; and Tambe, M. 2011. A double oracle algorithm for zero-sum security games on graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 327–334. International Foundation for Autonomous Agents and Multiagent Systems.

Jain, M.; Conitzer, V.; and Tambe, M. 2013. Security scheduling for real-world networks. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 215–222. International Foundation for Autonomous Agents and Multiagent Systems.

Kar, D.; Ford, B.; Gholami, S.; Fang, F.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Nsubaga, M.; et al. 2017. Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 159–167. International Foundation for Autonomous Agents and Multiagent Systems.

Konda, V. R., and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014.

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Perolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. *arXiv preprint arXiv:1711.00832*.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 536–543.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Pita, J.; Jain, M.; Marecki, J.; Ordóñez, F.; Portway, C.; Tambe, M.; Western, C.; Paruchuri, P.; and Kraus, S. 2008. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, 125–132. International Foundation for Autonomous Agents and Multiagent Systems.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering

the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Tambe, M. 2011. Security and game theory: Algorithms. *Deployed Systems, Lessons Learned*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *AAAI*, 2094–2100.

Xu, H.; Ford, B.; Fang, F.; Dilkina, B.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Nsubaga, M.; et al. 2017. Optimal patrol planning for green security games with black-box attackers. In *International Conference on Decision and Game Theory for Security*, 458–477. Springer.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2852–2858.