Deep Reinforcement Learning for Green Security Games with Real-Time Information

Yufei Wang[1], Zheyuan Ryan Shi[2], Lantao Yu[3], Yi Wu[4], Rohit Singh[5], Lucas Joppa[6], Fei Fang[2]
[1]Peking University, [2]Carnegie Mellon University, [3]Stanford University
[4]University of California, Berkeley, [5]World Wild Fund for Nature, [6]Microsoft Research

Abstract

Green Security Games (GSGs) have been proposed and applied to optimize patrols conducted by law enforcement agencies in green security domains such as combating poaching, illegal logging and overfishing. However, real-time information such as footprints and agents' subsequent actions upon receiving the information, e.g., rangers following the footprints to chase the poacher, have been neglected in previous work. To fill the gap, we first propose a new game model GSG-I which augments GSGs with sequential movement and the vital element of real-time information. Second, we design a novel deep reinforcement learning-based algorithm, DeDOL, to compute a patrolling strategy that adapts to the real-time information against a best-responding attacker. DeDOL is built upon the double oracle framework and the policy-space response oracle, solving a restricted game and iteratively adding best response strategies to it through training deep Q-networks. Exploring the game structure, DeDOL uses domain-specific heuristic strategies as initial strategies and constructs several local modes for efficient and parallelized training. To our knowledge, this is the first attempt to use Deep Q-Learning for security games.

Introduction

Security games (Tambe 2011) have been used for addressing complex resource allocation and patrolling problems in security and sustainability domains, with successful applications in critical infrastructure protection, security inspection and traffic enforcement (Basilico, Gatti, and Amigoni 2009; Durkota et al. 2015; Yin, An, and Jain 2014; Rosenfeld and Kraus 2017). In particular, Green Security Games (GSG) have been proposed to model the strategic interaction between law enforcement agencies (referred to as defenders) and their opponents (referred to as attackers) in green security domains such as combating poaching, illegal logging and overfishing. Mathematical programming based algorithms are designed to compute the optimal defender strategy, which prescribes strategically randomized patrol routes for the defender (Fang, Stone, and Tambe 2015; Fang et al. 2016; Xu et al. 2017).

Despite the efforts, a key element, real-time information, which exists widely in practice in green security domains, has been neglected in previous game models, not to mention the agents' subsequent actions upon receiving the information. For example, rangers can observe traces left by the poacher (e.g., footprints, tree marks) or learn of poacher's location in real time from camera traps and conservation drones. A well-trained ranger would make use of the real-time information to adjust her patrol route. Indeed, stories have been reported that rangers arrested the poachers after finding blood stains on the ground nearby (Maasailand Preservation Trust 2011). Similarly, a poacher may also observe the ranger's action in real time and adjust his attack plan, and the rangers should be aware of such risk. Thus, the prescribed patrol plans in previous work have limited applicability in practice as they are not adaptive to observations during the patrol.

Our paper aims at filling the gap. First, we propose a new game model GSG-I which augments GSGs with the vital element of real-time information and allows players to adjust their movements based on the received real-time information. These features lead to significant complexity, inevitably resulting in a large extensive-form game (EFG) with imperfect information.

Second, we design a novel deep reinforcement learning (DRL)-based algorithm, DeDOL (Deep-Q Network based Double Oracle enhanced with Local modes), to compute a patrolling strategy that adapts to the real-time information for zero-sum GSG-I. DeDOL is among the first few attempts to leverage advances in DRL for security games (Kamra et al. 2018; Trejo, Clempner, and Poznyak 2016) and is the first to use deep Q-learning for complex extensive-form security games. DeDOL builds upon the classic double oracle framework (DO) (McMahan, Gordon, and Blum 2003; Bosansky et al. 2013) which solves zero-sum games using incremental strategy generation, and a meta-method named policy-space response oracle (PSRO) (Lanctot et al. 2017) which augments DO with RL to handle a long time horizon in multi-agent interaction. Tailored towards GSG-I, DeDOL uses a deep Q-network (DQN) to compactly represent a pure strategy, integrates several recent advances in deep RL to find an approximate best response, which is a key step in the DO framework. Further, DeDOL uses domain-specific heuristic strategies, including a parameterized random walk strategy and a random sweeping strategy as initial strategies to warm up the strategy generation process. In addition, ex-

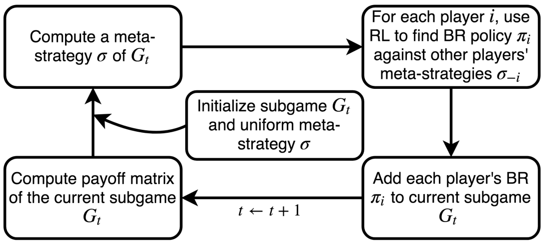


Figure 1: The PSRO framework

ploring the game structure of GSG-I, DeDOL uses several local modes, each corresponding to a specific entry point of the attacker, to reduce the complexity of the game environment for efficient and parallelized training.

Finally, we provide extensive experimental results to demonstrate the effectiveness of our algorithm in GSG-I. We show that the DQN representation in DeDOL is able to approximate the best response given a fixed opponent. In small problems, we show that DeDOL achieves comparable performance as existing approaches for EFGs such as counterfactual regret (CFR) minimization. In large games where CFR becomes intractable, DeDOL can find much better defender strategies than other baseline strategies.

Preliminaries and Related Work

Stackelberg Security Games (SSG) and Green Security Games GSGs are a special class of SSG (Tambe 2011; Pita et al. 2008; Fang, Jiang, and Tambe 2013). In a GSG (Fang, Stone, and Tambe 2015; Basak et al. 2016), a defender and an attacker interact in an area discretized into a grid of targets. The defender strategically allocates a limited number of patrol resources to patrol routes. The attacker chooses a target to attack. Each target is associated with reward and penalty for the defender and the attacker, representing the payoff to them depending on whether the attack on the target is successful. Existing literature on GSGs and SSGs widely employs the solution concept of Strong Stackelberg Equilibrium (SSE), where the defender commits to a strategy that maximizes her expected utility assuming the attacker observes her strategy and best responds to it.

When the game is zero-sum, common solution concepts such as Nash equilibrium (NE), SSE, Minimax, and Maximin, coincide, and a DO framework (McMahan, Gordon, and Blum 2003) is commonly applied to solve the game efficiently when the action space is large. DO is an iterative algorithm where in each iteration an NE is computed for a restricted game, in which each player only has a subset of pure strategies. Each player then adds a best response strategy against the opponent's current NE strategy to the restricted game. DO terminates when each player's best response strategy is already included in the restricted game. DO is guaranteed to converge to an NE of the original two-player zero-sum game.

Most of the literature on SSG have neglected real-time information, with only a few exceptions (Zhang et al. 2014) that are not designed for green security domains.

Extensive-form Games (EFG) EFGs capture the sequential interaction between the players, and often presents more computational challenges than normal-form games (Letchford and Conitzer 2010). Advanced algorithms for solving large-scale two-player zero-sum EFGs with imperfect information use counterfactual regret (CFR) minimization (Zinkevich et al. 2008), first-order methods (Kroer, Farina, and Sandholm 2017), abstraction (Brown and Sandholm 2017b; Čermák, Bošansky, and Lisy 2017), or mathematical programming-based approach enhanced with the DO framework (Bosansky et al. 2013). Despite the huge success in solving poker games whose game tree is wide but shallow (Brown and Sandholm 2017a; Moravčík et al. 2017; Bowling et al. 2015), these approaches are not applicable to GSG-I, as its game tree is prohibitively deep in contrast to poker games. For example, CFR requires traversing the full game tree in each iteration and will run out of memory on the large instances of GSG-I.

Deep RL and Multi-Agent RL Deep RL has recently been widely used in complex sequential decision-making, in both single agent and multi-agent settings (Oh et al. 2015; Leibo et al. 2017; Foerster et al. 2016). They have led to successful applications in Atari games (Mnih et al. 2015), Go (Silver et al. 2016), and continuous action control (Mnih et al. 2016). An RL problem is usually formulated as a Markov Decision Process (MDP), comprising the state space S, action space A, transition probability P, reward function r, and the discounting factor γ. Q-learning (Watkins and Dayan 1992) is a popular value-based RL methods for discrete action space. The Q-value of a state-action pair (s, a) under policy π is defined as $Q^{\pi}(s_t, a_t) = \mathbb{E}_{s,a\sim\pi}[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})|s_t, a_t]$. DQN (Mnih et al. 2015) uses a deep neural network Q^{θ} to learn the optimal Q value $Q^*(s, a) = max_{\pi} Q^{\pi}(s, a)$, by storing transitions $\{s, a, r, s'\}$ in an off-line replay buffer and minimizing the following loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[(Q^{\theta}(s, a) - (r + \gamma \max_{a'} Q^{\tilde{\theta}}(s', a'))^2] \quad (1)$$

where $Q^{\tilde{\theta}}$ is the target Q network whose parameters $\tilde{\theta}$ are periodically copied from θ to stabilize training. Besides Q-learning, Policy Gradient (Sutton et al. 2000) is another kind of popular RL method. It employs a parametric stochastic policy π_{θ}, and updates θ by gradient ascent according to the following theorem:

$$\nabla_{\theta} E_{\pi_{\theta}}[r] = E_{s,a\sim\pi_{\theta}}[\nabla_{\theta} log\pi_{\theta}(a|s) \cdot Q^{\pi_{\theta}}(s, a)] \quad (2)$$

A recent progress in multi-agent RL is the PSRO method (Lanctot et al. 2017) (illustrated in Figure 1) which generalizes DO by extending the pure strategies in the restricted game to parametrized policies and using deep RL to compute an approximate best response. PSRO provides a tractable approach for multi-player games with a long time horizon. However, since training in deep RL is time-consuming, it can only run a very limited number of iterations for large games, far less than needed for the convergence of DO. Thus, it may fail to find good strategies. We propose several enhancements to PSRO to mitigate this concern, and provide a concrete implementation for GSG-I.

Other Related Work Patrolling game is an EFG where a patroller moves on a graph and an attacker chooses a node

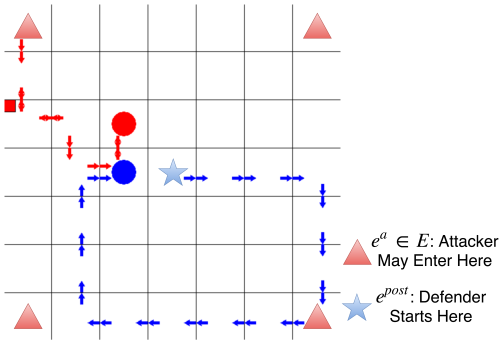


Figure 2: Illustration of GSG-I. The red dot and blue dot represent the attacker and defender respectively. The red arrows and blue arrows represent their corresponding footprints. The red squares on the upper left corner of some cells represent the attack tools placed by the attackers. Each player only observes the opponent's footprint in their current cell.

to "penetrate" (Agmon, Kraus, and Kaminka 2008; Basilico, Gatti, and Amigoni 2009; Horák and Pechoucek 2017). Our game model extends patrolling games by allowing multiple attacks and partial observability of traces of movement in real time for both the defender and attacker.

Green Security Game with Real-Time Information

In this section, we introduce GSG-I, Green Security Game with Real-Time Information. As shown in Figure 2, the basic environment of GSG-I is a grid world, with each cell representing a unique geographic area. The game has two players, the defender and the attacker. At the beginning of the interaction, the attacker randomly chooses an entry point e^a from a set E of entry points, while the defender always starts from the patrol post e^{post}. The attacker starts with a limited number of attack tools and uses them to set attacks at his desired locations. Such a game model is appropriate for a number of green security domains, e.g., rangers patrol in a conservation area to prevent poaching by removing animal snares and arresting poachers. At each time step, the defender picks an action from her action space $\mathcal{A}^d$: {*up, down, right, left, stand still*}. Simultaneously, the attacker picks an action from his action space $\mathcal{A}^a$: {*up, down, right, left, stand still*} $\times$ {*place attack tool, not place attack tool*}.

Suppose an attack tool has been placed in a cell with coordinate (i, j). At each time step, the attack tool successfully launches an attack with probability $P_{i,j}$. After a successful attack, the tool will be removed from the system. The defender tries to remove the attack tools prior to the attack and catch the attacker to stop him from placing more attack tools. She receives a positive reward $r^{tool}_{i,j}$ when she removes an attack tool from cell (i, j), a positive reward r^{catch} on catching the attacker, and a negative reward $p^{attack}_{i,j}$ when an attack tool launches an attack at cell (i, j). The interaction ends either when the defender finds the attacker and all the attack tools, or when a maximum time step T is reached. The defender's final payoff is the cumulative reward in the game. The attacker receives (positive or negative) rewards corresponding to these events as well and in this paper we focus on zero-sum games.

As shown in Figure 2, both players leave footprints as they move around. There can be many other forms of real-time information such as dropped belongings and local witnesses, yet for simplicity we use only footprints in this paper. In our game we assume both players have only *local observations*. They only observe their opponent's footprints in the current cell rather than the full grid, reflecting the fact that they often have a limited view of the environment due to the dense vegetation, complex terrain, or formidable weather. We assume the players have unlimited memory and can keep a record of the observations since the beginning of each interaction.

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

Computing Optimal Patrol Strategy

It is nontrivial to find an optimal patrol strategy. Simple action rules such as following the footprints or escaping from the footprints may not be the best strategy as shown in experiments. We now introduce DeDOL, our algorithm designed for computing the optimal defender's patrol strategy in zero-sum GSG-I. DeDOL builds upon the PSRO framework. Thus we will first introduce a DQN-based oracle for computing an approximate best response, and then introduce DeDOL, which uses the best response oracle as a subroutine.

Approximating Player's Best Response

We first consider an easier scenario where either player is static, i.e. using a fixed and possibly randomized strategy. The player's fixed strategy and the game dynamics of GSG-I then defines an MDP for the other player. We represent the other player's policy by a neural network, and use reinforcement learning to find an empirically best response strategy. In this subsection we assume the defender is the learning player, as the method for the attacker is identical.

Due to the strong spatial patterns of GSG-I, we employ a convolutional neural network (CNN) to represent the defender policy $\pi_d(a^d_t|s^d_t)$, which is a mapping from her state space to her action space. The input to the CNN is the defender's state s^d_t, represented by a 3-D tensor with the same width and height as the grid world and each channel encoding different features. Specifically, the first 8 channels are the binary encodings of the local attacker footprints ({four directions} $\times$ {entering or leaving}); the next 8 channels are similar encodings of the defender's own footprints; the 17th channel is one-of-K encoding that indicates the defender's current location; the 18th channel is the success probability of the attack tools of the grid world; the 19th channel is the normalized time step, which is the same for all cells.

Figure 3 shows the neural network architecture when the game has a 7×7 grid (the network architecture of other grid sizes is detailed in Appendix A[1]). The first hidden layer is a convolutional layer with 16 filters of size 4×4 and strides

[1]All appendices can be found at `https://arxiv.org/abs/1811.02483`

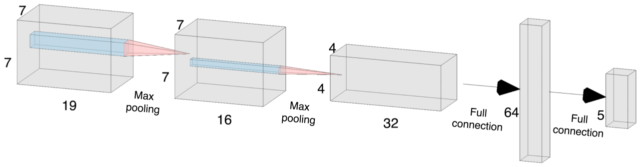


Figure 3: The defender's neural network architecture for the 7×7 grid.

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 $relu$ non-linear transformation and a max-pooling layer. The output layer is a fully-connected layer which transforms the hidden representation of the state to the final policy: each output dimension represents the Q-value of each action, and the neural network corresponds to a pure defender strategy where she takes the action with the highest Q-value.

We use Deep Q-learning (Mnih et al. 2015) to approximate the best response with the above neural network. Due to the highly dynamic environment of GSG-I, the training of the vanilla version of DQN proved difficult, especially when the other player uses a randomized strategy. Therefore, we employ the double DQN methods (Van Hasselt, Guez, and Silver 2016) to improve the stability of training, and the loss we minimize changes to:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[(Q^{\theta}(s,a) - (r + \gamma Q^{\tilde{\theta}}(s', \arg\max_{a'} Q^{\theta}(s',a')))^2]$$

Furthermore, we incorporate the dueling network architecture (Wang et al. 2016) upon double DQN for more efficient learning. We also implement the actor-critic algorithm (Konda and Tsitsiklis 2000) as an alternative to DQN, where $Q^{\pi_\theta}(s,a)$ in Eq.2 is replaced by $r + \gamma V^{\pi_\theta}(s) - V^{\pi_\theta}(s')$ to lower the variance. The neural network in Figure 3 then corresponds to a stochastic policy where each dimension of the output layer represents the probability of choosing that action. We implement another CNN to approximate the state-value $V^{\pi_\theta}(s)$ by changing the last output layer to be a scalar. At last, we apply gradient clipping (Pascanu, Mikolov, and Bengio 2013) to the training to deal with the gradient exploding issue.

The reader might notice that this neural network-based representation does not capture all defender strategies. However, the strong expressiveness makes it a memory-efficient alternative. Furthermore, we show later that we lose little by using this compact representation.

The DeDOL Algorithm

Having deep RL as the players' best response oracle lays the groundwork for finding the optimal defender strategy in GSG-I. In zero-sum GSG-I, the SSE strategy is also the NE strategy. The PSRO framework (Lanctot et al. 2017) (Figure 1) can be applied to compute the NE strategy. A naive implementation of PSRO in GSG-I is as follows: we use a randomly initialized DQN as the initial strategy for each player. At each iteration, we first get the payoff matrix for the current strategies by simulation and compute the NE for the current game matrix. Then we fix the NE strategy for one player and calculate the best response strategy of another player with DQN, as detailed above. We add these best response strategies of each player to the current game if they are better than the existing strategies, and repeat the procedure until no better responses can be found for either player.

However, as we will show in the experiment section, this naive implementation (referred to as Vanilla-PSRO throughout) does not perform well in practice due to the following limitations: 1) randomly initialized DQNs are rarely meaningful policies, and it takes several iterations for Vanilla-PSRO to evolve a reasonable set of strategies out of them. This problem is especially prominent as the training of DQN takes a nontrivial amount of time. 2) The computed best response in Vanilla-PSRO tends to overfit to the specific NE strategy that it was trained against, and may not be robust against other opponent strategies in the complex GSG-I. 3) In GSG-I, the attacker could enter the grid world through multiple entry points. Using a single best response defender DQN to deal with all these possibilities makes the training rather difficult and the learned strategies sub-optimal.

Therefore, we propose DeDOL, which enhances Vanilla-PSRO by introducing three key elements as discussed below.

Initial Strategies for DO The problem with naive initial strategies is that the best response against a highly exploitable strategy could still be highly exploitable itself. Thus, adding such a best response strategy to the strategy profile helps little. To alleviate this problem, we propose two lightweight yet effective domain-specific heuristic strategies as the initial strategies of DO.

For the attacker, we use a parameterized random walk policy. Suppose the current coordinate of the attacker is (m,n) and the maximum coordinate on the map is (M,N). We can define the average success probability for the *up* direction as $\frac{1}{(m-1)\cdot N} \sum_{0 \le i < m, 0 \le j \le N} P_{i,j}$. Recall that $P_{i,j}$ is the probability that an attack tool launches an attack successfully at cell (i,j). Similarly, we can define the average success probability for all the other directions. For simplicity, we use an integer $k \in \{1, \ldots, 5\}$ to denote one of the five directions (the fifth "direction" is for the action "stand still"). This way, we can get an average success probability vector $\bar{P} \in \mathbb{R}^{+5}$ ($\bar{P}_5$ is the success probability of the current grid). 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 parameterized heuristic policy for the attacker's movement as

$$\pi_a(a^a_t = k | s^a_t) = \frac{\exp(w_p \cdot \bar{P}_k + w_i \cdot I_k + w_o \cdot O_k)}{\sum_z \exp(w_p \cdot \bar{P}_z + w_i \cdot I_z + w_o \cdot O_z)} \quad (3)$$

where w_p, w_i and w_o are parameters for the average success probability, entering and leaving footprints, respectively.

The success probability of the attack tool directly impacts the decision of where to place it. We define the probability of placing an attack tool in cell (m,n) as

$$\eta_a(b^a_t = 1 | s^a_t) = \frac{\exp(P_{m,n}/\tau)}{\sum_i \sum_j \exp(P_{i,j}/\tau)} \quad (4)$$

where τ is a temperature parameter.

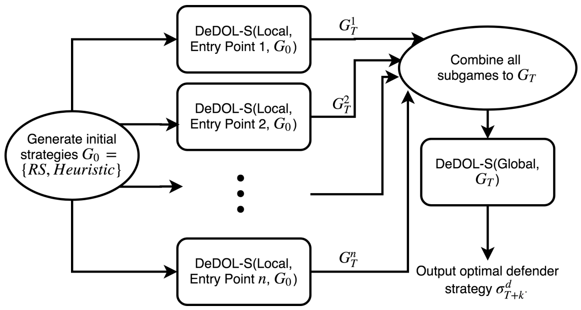


Figure 4: The DeDOL Algorithm

The behavioral model as described above is boundedly rational. Real-world applications often feature bounded rationality due to various constraints. We use this parameterized heuristic policy as the initial attacker strategy in DeDOL with parameters set following advice from domain experts[2].

For the defender's initial strategy, we could use a similar, and even simpler parameterized heuristic random walk policy, as her decision only involves movement. However, here we propose to use another more effective policy, called random sweeping. In the beginning, the defender randomly chooses a direction to move in and heads towards the boundary. She then travels along the boundary until she finds any footprint from the attacker and follows the footprints. If there are multiple footprints at the same cell, she randomly chooses one to follow. This turns out to be a very strong strategy, as to defeat it, the attacker has to confuse the defender using his footprints.

Exploration and Termination The best response against the NE of a subgame G_t, $Nash(G_t)$, may not generalize well against other unexplored strategies in a complex game like GSG-I. In DeDOL, the fixed player instead uses a mixture of $Nash(G_t)$ and $Unif(G_t)$, the uniform random strategy where the player chooses each strategy in G_t with equal probability. That is, with probability $1-\alpha$ he plays $Nash(G_t)$, and with probability α he plays $Unif(G_t)$.

As a result, the trained DQN is an (approximate) best response to the NE strategy mixed with exploration, rather than the NE itself. Therefore we need to check in subroutine VALID (Algorithm 2) whether it is still a better response to the NE strategy than the existing strategies. This method is similar to the better response oracle introduced in (Jain, Conitzer, and Tambe 2013). If neither of the new strategies for the two players is a better response, we discard them and train against the NE strategies without exploration. The parent procedure Algorithm 1 terminates if we again find no better responses. Algorithm 1 may also terminate if it is intended to run a fixed number of iterations or cut short by the user. Upon termination, we pick the defender NE strategy (possibly plus exploration) and the attacker's best response which together give the highest defender's expected utility.

Local Modes We refer to the algorithm introduced so far as DeDOL-S (Algorithm 1). Our main algorithm DeDOL, illustrated in Figure 4, uses DeDOL-S as a subroutine. We

[2]We showed the experts the attacker behaviors with different parameter combinations using our designed GUI, and pick the one they think most reasonable.

Algorithm 1 DeDOL-S

Input: Mode (local/global), attacker entry point (if local), initial subgame G_0, exploration rate α

1: **for** iteration t **do**
2: Run simulations to obtain current game matrix G_t.
3: $Nash(G_t) = (\sigma_t^d, \sigma_t^a), Unif(G_t) = (\rho_t^d, \rho_t^a)$.
4: Train defender DQN f_t^d against $(1-\alpha)\sigma_t^a + \alpha\rho_t^a$.
5: Train attacker DQN f_t^a against $(1-\alpha)\sigma_t^d + \alpha\rho_t^d$.
6: VALID(f_t^d, f_t^a, G_t)
7: **if** TERMINATE condition satisfied **then**
8: $k^* = \arg\max_k\{defEU((1-\alpha)\sigma_k^d + \alpha\rho_k^d, f_k^a)$, and $defEU(\sigma_k^d, \overline{f_k^a})$ if any were ever calculated$\}$
9: **Output:** Defender optimal strategy from the k^*th iteration per above, current subgame G_t

Algorithm 2 VALID

Input: DQNs f_t^d, f_t^a, subgame G_t with NE (σ_t^d, σ_t^a)

1: **if** $\sigma_t^a \cdot G_t(f^a, f_t^d) \geq \sigma_t^a \cdot G_t(f^a, f_k^d), \forall k < t$ **then**
2: Defender best response f_t^d is valid, add to G_t
3: **if** $\sigma_t^d \cdot G_t(f^d, f_t^a) \geq \sigma_t^d \cdot G_t(f^d, f_k^a), \forall k < t$ **then**
4: Attacker best response f_t^a is valid, add to G_t
5: **if** neither of the above is true **then**
6: Fix σ_t^a from G_t, train defender DQN $\overline{f_t^d}$ against it.
7: Fix σ_t^d from G_t, train attacker DQN $\overline{f_t^a}$ against it.
8: Do Lines 1-4 with f_t^d, f_t^a replaced by $\overline{f_t^d}, \overline{f_t^a}$.
9: If neither 'if' is true again, signal TERMINATE

now conclude this section by introducing the key feature of DeDOL: local modes.

Since it is challenging for a single defender DQN to approximate best response against an attacker entering from different cells, we divide the original game (referred to as the global mode) into several local modes. In each mode the attacker has a fixed entry location. In DeDOL, we first run DeDOL-S in each of the local modes in parallel. After a few iterations, we combine the DQNs trained in all local modes to form a new subgame. Then, we use this new subgame as the initial subgame and run DeDOL-S in the global mode for more iterations.

When the attacker enters from the same location, both players (especially the defender) will face a more stable environment and thus are able to learn better strategies more quickly. More importantly, these strategies serve as good building blocks for the equilibrium meta-strategy in the global mode, thus improving the strategy quality. In the following section, we show that this approach performs better than several other variants.

Experiments

We test DeDOL in GSG-I using a case study on wildlife anti-poaching, where the grid world represents a wildlife conservation area. The attacker corresponds to a poacher carrying attack tools, i.e., snares, to catch animals. The defender corresponds to a patroller moving in the area to stop poaching by removing snares and arresting poachers. Each cell of the grid world has a corresponding animal density, which is proportional to the probability that a snare successfully catches

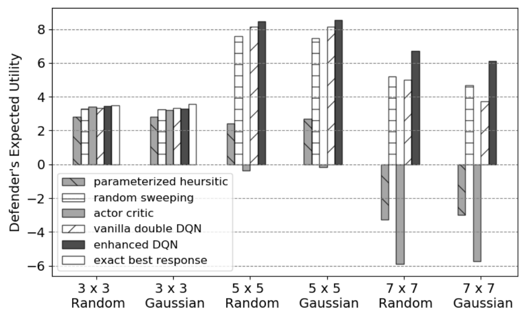


Figure 5: Expected utilities of different patroller strategies against a parameterized random walk poacher.

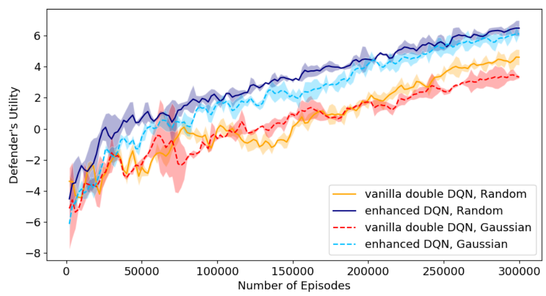


Figure 6: The learning curves of patroller DQNs against parameterized heuristic random walk poachers on 7×7 grids, averaged across four runs.

an animal in that cell. The animal densities are generated either uniformly randomly, or following a mixture Gaussian. The latter reflects that in reality the animal density is often higher along mountain ranges and decreases as we move away. The game environment of different types and sizes are shown in Appendix C. We test DeDOL on three grid worlds of different sizes: $3 \times 3, 5 \times 5$, and 7×7. All experiments are carried out on Microsoft Azure standard NC6 virtual machines, with a 6-core 2.60 GHz Intel Xeon E5-2690 CPU, a Tesla K80 GPU, and a 56G RAM.

Best Response Approximation

The ability to approximate a best response strategy against a fixed opponent is foundational to our algorithm. Therefore, we start by comparing the performance of several methods against a parameterized heuristic poacher with parameters set following the advice from domain experts. We compare the random sweeping strategy, parameterized random walk patroller with parameters set by grid search, the vanilla double DQN, the dueling double DQN + gradient clipping (enhanced DQN), and the actor-critic algorithm. On the 7×7 grid world, we train both DQNs and actor-critic using Adam optimizer (Kingma and Ba 2015) with a learning rate of 0.0001 for 300000 episodes. More detailed training parameters are provided in Appendix B. Figure 6 shows the learning curves of both DQNs in 7×7 grid. The actor-critic algorithm does not converge in our experiments.

In the smaller 3×3 game with 4 time steps, we can compute the exact poacher best response given a patroller strategy (Bosansky et al. 2013) (details in Appendix F). However, this method becomes intractable with just a 5×5 grid

α	0	0.1	0.15	0.25	0.4
3×3 Random	0.60	0.43	**0.73**	0.73	0.44
3×3 Gaussian	0.12	0.39	**0.64**	0.37	0.04

Table 1: Patroller's expected utility with different exploration rate α on the 3×3 grid using DeDOL (global only).

which has 25 time steps and over 10^{20} information sets.

The results of each method are summarized in Figure 5. The enhanced DQN patroller achieves the highest expected utility among all compared strategies in all settings. Compared to the exact solution in the 3×3 game, the enhanced DQN is indeed a very good best response approximation. Figure 7 provides an illustration of the learned enhanced DQN strategy on a 7×7 grid with random animal density. Note that the enhanced DQN patroller cleverly learns to first patrol towards the corner on a path of high animal densities. She then moves along the boundary, and upon finding the poacher's footprints, follows them to catch the poacher. After the poacher is caught, she induces from the observed footprints that the entry point of the poacher should be the bottom right corner. Hence she patrols that area and successfully removes a snare there. A similar visualization for the trained poacher DQN against a random sweeping patroller is shown in Appendix D. We dropped the actor-critic algorithm in subsequent experiments as it performs poorly.

Small Games

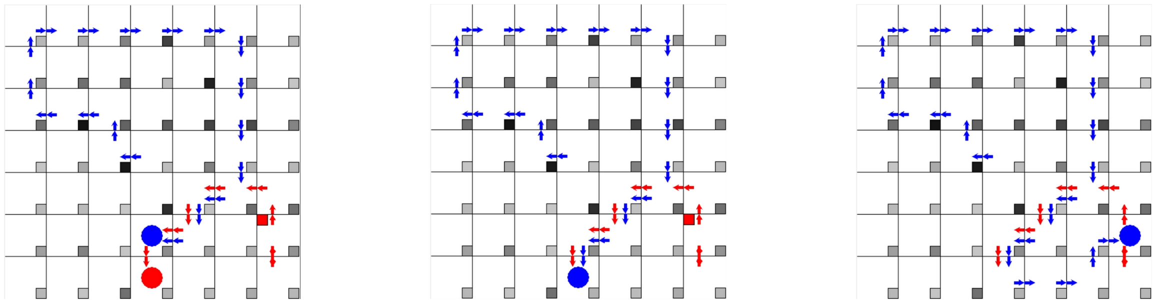
Now we have shown that (enhanced) DQN can approximate a best response well, we move on to test the whole DeDOL algorithm. We first test it on a 3×3 grid. The game has 4 time steps, and the attacker has 3 snares. The full game tree has roughly 4.5×10^7 nodes.

Before going into the main results, we tune the exploration rate α by running 16 iterations in DeDOL-S global mode. Table 1 shows the highest patroller's expected utility against an exact best response poacher with different exploration rate. Since $\alpha = 0.15$ achieves the best result in both map types, we set α to be 0.15 in the following experiments. The defender's utility with $\alpha = 0.15$ is also much higher than with $\alpha = 0$, showing that exploration is helpful.

We now compare the performance of DeDOL with other baselines in GSG-I. To investigate whether DQNs trained in local modes would indeed help in global mode, we implement three versions of the DeDOL algorithm: 1) we run zero iteration in local modes, i.e., run DeDOL-S directly in global mode; 2) we run DeDOL-S in local modes for several iterations, return to the global mode and run for several more iterations (Figure 4); 3) we run DeDOL-S purely in the local modes, and upon termination, return to the global mode, compute an NE strategy and running no more iterations.

We use the counterfactual regret (CFR) minimization (Zinkevich et al. 2008), random sweeping, and Vanilla-PSRO as three baselines. Each learning algorithm runs for a day. In particular, in the local + global version of DeDOL, half a day is used for local modes and the rest for the global mode. With chance sampling, the CFR algorithm traverses roughly 4.3×10^6 nodes in one iteration, and finishes 3500 iterations in a day on our hardware.

The first two rows of Table 2 report the highest patroller's expected utilities, calculated against an exact best response poacher. We note that the highest patroller's utility achieved



(a) At time step 18 (b) At time step 19 (c) At time step 23

Figure 7: The learned patroller DQN strategy against a parameterized heuristic random walk poacher. Here, the darkness of the square in each cell indicates the animal density.

	Random Sweeping	Vanilla PSRO	DeDOL Pure Global Mode	DeDOL Local + Global Mode	DeDOL Pure Local Mode	CFR
3 × 3 Random	-0.04	0.65 (16)	0.73 (16)	**0.85** (10 + 2)	0.71 (20)	**1.01** (3500)
3 × 3 Gaussian	-0.09	0.52 (16)	0.75 (16)	**0.86** (10 + 2)	0.75(20)	**1.05** (3500)
5 × 5 Random	-1.91	-8.98 (4)	-1.63 (4)	-0.42 (4 + 1)	**-0.25** (5)	-
5 × 5 Gaussian	-1.16	-9.09 (4)	-0.43 (4)	**0.60** (4 + 1)	-2.41 (5)	-
7 × 7 Random	-4.06	-10.65 (4)	-2.00 (4)	**-0.54** (3 + 1)	-1.72(5)	-
7 × 7 Gaussian	-4.25	-10.08 (4)	-4.15 (4)	**-2.35** (3 + 1)	-2.62(5)	-

Table 2: The highest patroller's expected utility among all DO / CFR iterations. The numbers in the parentheses show the finished DO / CFR iterations within the given running time. The highest value among all algorithms are in bold. The detail values of the defender expected utility at each iteration of DeDOL are shown in Appendix E.

by DeDOL is slightly lower than that of CFR given the same amount of time. However, DeDOL needs much less memory as it only needs to store several neural networks, while the CFR algorithm has to store the whole huge game tree. The table also shows that all implementation versions of DeDOL outperform Vanilla-PSRO, and have much higher utility than the random sweeping baseline. In addition, the local + global modes version achieves the best result in both map types, which proves its effectiveness. Note that the local mode implementation finishes more iterations because the training of DQNs converges faster in its simpler environment.

Large Games

We also perform tests on large games with 5×5 and 7×7 grid. 5×5 game has 25 time steps, and 7×7 game has 75 time steps. In both games, the attacker has 6 snares.

We still implement 3 versions of DeDOL as detailed in the previous subsection. The running time on 5×5 grid is 3 days, and 5 days on the 7×7 game. For the local + global version of DeDOL, we allocate 2 days for local mode on 5×5, and 3 days on 7×7. We report the performance of DeDOL in Table 2. As aforementioned, with even a 5×5 grid, there are over 10^{50} game states and 10^{20} information sets. Thus, CFR becomes intractable in terms of running time and memory usage, so is computing the exact best response. Therefore, in Table 2 the patroller's expected utilities are calculated against their respective best response DQN poacher[3].

Similar to the results in small games, all versions of DeDOL significantly outperform the Vanilla-PSRO and the random sweeping baseline. The Vanilla-PSRO performs extremely poor here because it starts with a poor randomly initialized DQN strategy, and the strategies it evolved within the running time is still highly exploitable in the large grids. This validates the effectiveness of using the more reasonable random sweeping/parameterized heuristic strategies as the initial strategies in DeDOL. We also note DeDOL with local mode (either local + global retraining or pure local) achieves the highest defender's expected utility in all settings. This suggests that the strategies obtained in local modes are indeed very effective and serve as good building blocks to improve the strategy quality after returning to global mode.

[3]Here, we train a separate DQN for a longer time than in a DO iteration. We also test against the poacher's heuristic strategy and pick the better one, which is always the DQN in the experiments.

Discussions and Future Directions

We discuss a few questions the reader may have and some future directions. First, policy gradient performs poorly in GSG-I because it learns an average of all possible sweeping routes. Second, training DQNs is time-consuming. Though we have shown promising utility improvements, approximating NE definitely needs more iterations. Third, the global best response of an NE strategy computed in one local mode may actually be in another local mode. To address this, we hope to find a method to automatically restrict the global best response being in the current mode, which we leave for future research. Another future direction is to consider maximum entropy Nash equilibria as the meta-strategy. Finally, DeDOL is proposed for zero-sum GSG-I, but we expect it can be adapted to general-sum GSG-I, especially when the game is close to zero-sum.

Acknowledgement

The Azure computing resources are provided by Microsoft for Research AI for Earth award program.

References

Agmon, N.; Kraus, S.; and Kaminka, G. A. 2008. Multi-robot perimeter patrol in adversarial settings. In *ICRA'08*.

Basak, A.; Fang, F.; Nguyen, T. H.; and Kiekintveld, C. 2016. Combining graph contraction and strategy generation for green security games. In *GameSec'16*.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS'09*.

Bosansky, B.; Kiekintveld, C.; Lisy, V.; Cermak, J.; and Pechoucek, M. 2013. Double-oracle algorithm for computing an exact nash equilibrium in zero-sum extensive-form games. *AAMAS'13*.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit holdem poker is solved. *Science*.

Brown, N., and Sandholm, T. 2017a. Libratus: the superhuman ai for no-limit poker. In *IJCAI'17*.

Brown, N., and Sandholm, T. 2017b. Safe and nested subgame solving for imperfect-information games. In *NIPS'17*.

Čermák, J.; Bošansky, B.; and Lisy, V. 2017. An algorithm for constructing and solving imperfect recall abstractions of large extensive-form games. In *IJCAI'17*.

Durkota, K.; Lisỳ, V.; Bosanskỳ, B.; and Kiekintveld, C. 2015. Optimal network security hardening using attack graph games. In *IJCAI'15*.

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'16*.

Fang, F.; Jiang, A. X.; and Tambe, M. 2013. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *AAMAS'13*.

Fang, F.; Stone, P.; and Tambe, M. 2015. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI'15*.

Foerster, J.; Assael, I. A.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*.

Horák, K., and Pechoucek, M. 2017. Heuristic search value iteration for one-sided partially observable stochastic games.

Jain, M.; Conitzer, V.; and Tambe, M. 2013. Security scheduling for real-world networks. In *AAMAS'13*.

Kamra, N.; Gupta, U.; Fang, F.; Liu, Y.; and Tambe, M. 2018. Policy learning for continuous space security games using neural networks.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. *ICLR'15*.

Konda, V. R., and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *NIPS'00*.

Kroer, C.; Farina, G.; and Sandholm, T. 2017. Smoothing method for approximate extensive-form perfect equilibrium. In *IJCAI'17*.

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Perolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *NIPS'17*.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *AAMAS'17*.

Letchford, J., and Conitzer, V. 2010. Computing optimal strategies to commit to in extensive-form games. In *EC'10*.

Maasailand Preservation Trust. 2011. Poacher arrested with game meat. https://mpttalk.wordpress.com/ 2011/05/11/poacher-arrested-with-game-meat/.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *ICML'03*.

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*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *ICML'16*.

Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*.

Oh, J.; Guo, X.; Lee, H.; Lewis, R. L.; and Singh, S. 2015. Action-conditional video prediction using deep networks in atari games. In *NIPS'15*.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. *ICML'13*.

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 *AAMAS'08*.

Rosenfeld, A., and Kraus, S. 2017. When security games hit traffic: Optimal traffic enforcement under one sided uncertainty. In *IJCAI'17*.

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.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS'00*.

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

Trejo, K. K.; Clempner, J. B.; and Poznyak, A. S. 2016. Adapting strategies to dynamic environments in controllable stackelberg security games. In *CDC'16*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *AAAI'16*.

Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; and De Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. *ICML'16*.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.

Xu, H.; Ford, B.; Fang, F.; Dilkina, B.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; and Nsubaga, M. 2017. Optimal patrol planning for green security games with black-box attackers. In *GameSec'17*.

Yin, Y.; An, B.; and Jain, M. 2014. Game-theoretic resource allocation for protecting large public events. In *AAAI'14*.

Zhang, C.; Jiang, A. X.; Short, M. B.; Brantingham, P. J.; and Tambe, M. 2014. Defending against opportunistic criminals: New game-theoretic frameworks and algorithms. In *GameSec'14*.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *NIPS'08*.