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

Yufei Wang\textsuperscript{1}, Zheyuan Ryan Shi\textsuperscript{2}, Lantao Yu\textsuperscript{3}, Yi Wu\textsuperscript{4}, Rohit Singh\textsuperscript{5}, Lucas Joppa\textsuperscript{6}, Fei Fang\textsuperscript{2}  
\textsuperscript{1}Peking University, \textsuperscript{2}Carnegie Mellon University, \textsuperscript{3}Stanford University  
\textsuperscript{4}University of California, Berkeley, \textsuperscript{5}World Wild Fund for Nature, \textsuperscript{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-
exploring 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 SSGs are a special class of SSG (Tambe 2011; Games Stackelberg Security Games (SSG) and Green Security Fender strategies than other baseline strategies. CFR becomes intractable, DeDOL can find much better def actional regret (CFR) minimization. In large games where CFR becomes intractable, DeDOL can find much better defender strategies than other baseline strategies.

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 $Q^\pi(s, a) = \mathbb{E}_{s' \sim P(s', r)}[\gamma \sum_{t=0}^{\infty} \gamma^t r(s_{t+1}, a_{t+1}) | s_t, a_t]$. DQN (Mnih et al. 2015) uses a deep neural network $Q^\theta$ to learn the optimal Q value $Q^\pi(s, a) = max_a Q^\pi(s, a)$, by storing transitions $\{s, a, r, s'\}$ in an off-policy replay buffer and minimizing the following loss:

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

where $Q^\theta$ is the target Q network whose parameters $\theta$ are periodically copied from $\theta$ 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 $\pi_\theta$, and updates $\theta$ by gradient ascent according to the following theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s,a \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(a|s) \cdot Q^\pi(s, a)]$$ (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 patrol moves on a graph and an attacker chooses a node.
corresponding to these events as well and in this paper we
game. The attacker receives (positive or negative) rewards
The defender’s final payoff is the cumulative reward in the
time. The interaction ends either when the defender finds the attacker and all the
The red squares on the upper left corner of some cells rep-
are included in the interaction area to prevent poaching by removing animal snares and
To “penetrate” (Agmon, Kraus, and Kaminka 2008; Basilico,
the attacker. The defender randomly chooses an entry point \( e^a \) from a
\[ \{ \text{up, down, right, left, stand still} \} \]
simultaneously, the attacker picks an action from his
\( \{ \text{attack tool, not place attack tool} \} \)
Suppose an attack tool has been placed in a cell with coordi-
attacker to stop him from placing more attack
tools. She receives a positive reward \( r^\text{catch}_{t+1} \) when she removes
an attack tool from cell \((i, j)\), a positive reward \( r^\text{catch} \) on
catching the attacker, and a negative reward \( r^\text{attack}_{t+1} \) 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 veg-
etation, 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 pa-
er 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 ac-
ion rules such as following the footprints or escaping from
the footprints may not be the best strategy as shown in ex-
periments. We now introduce DeDOL, our algorithm de-
gigned for computing the optimal defender’s patrol strategy
in zero-sum GSG-I. DeDOL builds upon the PSRO frame-
work. 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 reinforce-
ment 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
convolutional neural networks (CNN) to represent the de-
"
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:

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

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 \(Q^\pi(s,a)\) in Eq.2 is replaced by \(r + \gamma V^\pi(s) - V^\pi(s')\) 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 \(V^\pi(s)\) 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 \(\frac{1}{(m-1) \times N} \sum_{0 \leq n \leq N} P_{i,j}\). 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 \(\bar{P} \in \mathbb{R}^5\) (\(P_k\) 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

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

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

\[
\eta_a(b_t^a = 1 | s_t^a) = \frac{\exp(P_{m,n} / \tau)}{\sum_i \sum_j \exp(P_{i,j} / \tau)}
\]

where \(\tau\) is a temperature parameter.
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.

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 $G_t$, $\text{Nash}(G_t)$, 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 $\text{Nash}(G_t)$ and $\text{Unif}(G_t)$, the uniform random strategy where the player chooses each strategy in $G_t$ with equal probability. That is, with probability $1 - \alpha$ he plays $\text{Nash}(G_t)$, and with probability $\alpha$ he plays $\text{Unif}(G_t)$.

As a result, the trained DQN is an (approximate) best response against an attacker entering from any other unexplored strategy 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 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 an animal.
optimizer (Kingma and Ba 2015) with a learning rate of 7.

On the grid world, we train both DQNs and actor-critic using Adam.

We compare against a parameterized heuristic poacher 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 random sweeping strategy, parameterized random walk.

We start by comparing the performance of several methods against a parameterized heuristic random walk poacher.

The ability to approximate a best response strategy against a

Figure 6: The learning curves of patroller DQNs against parameterized heuristic random walk poachers on $7 \times 7$ 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 $7 \times 7$. 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 $7 \times 7$ grid world, we train both DQNs and actor-critic using Adam optimizer (Kingma and Ba 2015) with a learning rate of 0.0001 for 500000 episodes. More detailed training parameters are provided in Appendix B. Figure 6 shows the learning curves of both DQNs in $7 \times 7$ grid. The actor-critic algorithm does not converge in our experiments.

In the smaller $3 \times 3$ grid 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 $5 \times 5$ grid.

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 $3 \times 3$ 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 $7 \times 7$ grid with random animal density.

The defender’s utility with $\alpha = 0$ 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 $4.5 \times 10^7$ 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
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 $5 \times 5$ and $7 \times 7$ grid. $5 \times 5$ game has 25 time steps, and $7 \times 7$ 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 $5 \times 5$ grid is 3 days, and 5 days on the $7 \times 7$ game. For the local + global version of DeDOL, we allocate 2 days for local mode on $5 \times 5$, and 3 days on $7 \times 7$. We report the performance of DeDOL in Table 2. As aforementioned, with even a $5 \times 5$ grid, there are over $10^{50}$ game states and $10^{39}$ 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 poacher1.

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.

### 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.

---

1Here, 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.

---

**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.

**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.