Meta Strategy Guided Deep Reinforcement Learning in Green Security Games

Tianyu Gu

April 2020

1 Abstract

While multi-agent reinforcement learning algorithms have attracted many research interests, very few algorithms in the field were deployed in real-world scenarios due to their uninterpretablilty and sample inefficiency in the training process. On the other hand, tools based on Defender-Attacker Stackelberg games have been used extensively in the security field such as the ARMOR system in the LAX airport and the PROTECT system by the United States Coast Guard. In this paper, we propose a new meta-strategy guided multi-agent deep reinforcement learning algorithm to incorporate some of the power of security games.

2 Introduction

Stackelberg security game is a game model in which the leader first commits to a strategy and the follower then observes the leader's policy before committing to his own strategy. Stackelberg security game models have been successfully deployed in many real-world settings to protect important infrastructure. [5] helps schedule patrolling and monitoring for the Los Angeles International Airport (LAX). [1] is used by the United States Coast Guard (USCG) to combat terrorism. Outside of guarding physical infrastructure, game theoretic models have also been used to protect virtual targets. [6] developed a browser extension to combat social engineering attacks.

Despite Stackelberg security game models' success , deep reinforcement learning algorithms have attracted an increasing amount of interests because these algorithms do not require domain knowledge and are applicable in more than one scenarios. [3] developed an actor-critic, model-free algorithm based on the deterministic policy gradient. [2] builds upon [3] to maximize the agent's policy's entropy. [4] extends the line of work to incorporate coordination and competition of multiple agents.

While deep reinforcement learning algorithms have attracted much research interests, very few real-world applications have been built based upon these

algorithms because of their uninterpretablility and sample inefficiency during the training process. In this work, we propose an algorithm to leverage the success of the Stackelberg game theoretic models and to increase the interpretability and sample efficiency of the deep reinforcement learning based algorithms. We develop the algorithm in the patrolling setting with defender, attacker, and uav which could be controlled by the defender.

The algorithm follows two steps. The first step is to solve an abstracted simpler game with only attacker and defender. We solve this game using Mixed Integer Linear Programming. We then use the result as a high level strategy to guide the reinforcement learning algorithm based on [4, 2].

3 Game setting

The game takes place in a continuous space with discrete time steps. There are two players in the game, the defender and the attacker. The defender controls two resources, the patroller and the UAV. All agents move simultaneously with continuous displacement. Unlike patroller who is able to catch the poacher and end the game, the UAV, when encountered the poacher, cannot deter the poacher but can inform patroller the real time location of the poacher. The attacker only controls the poacher. There are N targets in the game that are susceptible to be attacked by the attacker. The attack is completed when the attacker moves close enough to one of the targets. The game ends when the attacker is caught or an attack is completed. The attacker can have different types $l \in L$ sampled from a prior distribution Q^L. If the attacker attacks target n without being caught, the attacker gains utility $U_a^{(u)(l)}(n) > 0$, the defender gains utility $U_d^u(n) < 0$. If the attacker is caught by the patroller, the attacker gains utility $U_a^{(c)(l)} < 0$, and the defender gains utility $U_d^c > 0$.

4 Abstracted game

4.1 Abstract game setting

In the abstract game, we only consider the poacher and the patroller. The continuous space is abstracted as an undirected graph $G = (E, V)$. Both agents move discretely along the edge and the game is zero sum. The defender takes Markovian patrolling strategy. The Attacker is able to observe the defender's strategy, but not able to observe defender's exact location on the graph. Assume when the attacker is attacking, the defender has already patrolled long enough so that the probability that the defender is in each node follows the stationary distribution.

4.2 Single type attacker Non Linear Program formulation

We first consider the case where there is only a single type of attacker. The linear program formulation to solve for the optimal Markovian defender strategy

is listed below. v represents the maximum expected payoff for the attacker. π_{ij} represent the probability of the defender moving to node j given he is currently in node i. A_{ij} is a constant which equals to 1 if there exists an edge between node i and node j and 0 otherwise. μ_i is the stationary probability that defender is in node i.

$$\min_{\mu,\pi,v} v \tag{1}$$

$$\text{s.t. } \pi_{ij} \geq 0 \qquad \forall i,j \tag{2}$$

$$\sum_j \pi_{ij} = 1 \qquad \forall i \tag{3}$$

$$\pi_{ij} \leq A_{ij} \qquad \forall i,j \tag{4}$$

$$\mu_i = \sum_j \pi_{ji}\mu_j \qquad \forall i \tag{5}$$

$$\sum_i \mu_i = 1 \qquad \forall i \tag{6}$$

$$\mu_i \in [0,1] \qquad \forall i \tag{7}$$

$$v \geq (1-\mu_j)U_a^u(j) + \mu_j U_a^c(j) \qquad \forall j \tag{8}$$

Since the abstracted game is zero sum, minimizing the expected attacker payoff in equation (1) is the same as minimizing defender payoff. equation (2) and (3) ensures that the transition probability π_{ij} is a valid probability distribution. Equation (4) ensures that the defender can only travel along the edges. Equation (5), (6) and (7) are the stationary equations to compute the defender's stationary probability μ_i. Equation (8) ensures that the attacker is maximizing her expected utility.

4.3 Multiple type attacker Non linear program formulating

We can then generalize the above formulation to account for multiple attacker types. q^l represents the probability of attacker of type l enters the game. Equation (10) ensures that attacker of different type is maximizing their own expected utility respectively.

$$\min_{\mu,\pi,v} \sum_l q^l v^l \tag{9}$$

$$\text{s.t. (2) (3) (4) (5) (6) (7)}$$

$$v^l \geq (1-\mu_j)U_a^{(u)(l)}(j) + \mu_j U_a^{(c)(l)}(j) \qquad \forall j,l \tag{10}$$

4.4 Converting to Mixed Integer Linear Programming

To remove the non-convex constraints in the above formulations, we apply a technique used in [7] to discretize variables and converting the program to a Mixed Integer Linear Program. The solution will be approximate solutions to the original formulation. Let variables $p_l \in [0,1]$ be the discrete levels with $p_0 = 0$ and $p_L = 1$, define $d_{ijl} \in \{0,1\}$ to be binary variables such that d_{ijl} indicates a particular discrete probability choice p_l Let $\pi_{ij} = \sum_l p_l d_{ijl}$ Replace π_{ij} in the original formulation and define $w_{ijl} = d_{ijl}\mu_i$. Then adding the following constraints completes the MILP.

$$\mu_i - Z(1 - d_{ijl}) \leq w_{ijl} \leq \mu_i + z(1 - d_{ijl}) \qquad \forall i, j \qquad (11)$$

$$-Z d_{ijl} \leq w_{ijl} \leq Z d_{ijl} \qquad \forall i, j \qquad (12)$$

4.5 Complete MILP formulation

We are now ready to present the complete mixed integer linear program for the abstract game.

4.5.1 MILP formulation - single attacker type

Below is the complete formulation for the single type attacker case.

$$\min_{\mu,\pi,v} v \qquad (13)$$

$$\text{s.t. } d_{ijl} \in \{0,1\} \qquad \forall i, j, l \qquad (14)$$

$$\sum_j \sum_l p_l d_{ijl} = 1 \qquad \forall i \qquad (15)$$

$$\sum_l d_{ijl} = 1 \qquad \forall i, j \qquad (16)$$

$$\sum_l p_l d_{ijl} \leq A_{ij} \qquad \forall i, j \qquad (17)$$

$$\mu_j = \sum_i \sum_l p_l w_{ijl} \qquad \forall j \qquad (18)$$

$$\sum_i \mu_i = 1 \qquad \forall i \qquad (19)$$

$$\mu_i \in [0,1] \qquad \forall i \qquad (20)$$

$$\mu_i - Z(1 - d_{ijl}) \leq w_{ijl} \leq \mu_i + z(1 - d_{ijl}) \qquad \forall i, j \qquad (21)$$

$$-Z d_{ijl} \leq w_{ijl} \leq Z d_{ijl} \qquad \forall i, j \qquad (22)$$

$$v \geq (1 - \mu_j) U_a^u(j) + \mu_j U_a^c(j) \qquad \forall j \qquad (23)$$

4.5.2 MILP formulation - multiple attacker type

Below is the complete formulation for the multiple type attacker case.

$$\min_{\mu,\pi,v} \sum_l q^l v^l \tag{24}$$

$$\text{s.t. } d_{ijl} \in \{0,1\} \qquad \forall i,j,l \tag{25}$$

$$\sum_j \sum_l p_l d_{ijl} = 1 \qquad \forall i \tag{26}$$

$$\sum_l d_{ijl} = 1 \qquad \forall i,j \tag{27}$$

$$\sum_l p_l d_{ijl} \leq A_{ij} \qquad \forall i,j \tag{28}$$

$$\mu_j = \sum_i \sum_l p_l w_{ijl} \qquad \forall j \tag{29}$$

$$\sum_i \mu_i = 1 \qquad \forall i \tag{30}$$

$$\mu_i \in [0,1] \qquad \forall i \tag{31}$$

$$\mu_i - Z(1-d_{ijl}) \leq w_{ijl} \leq \mu_i + z(1-d_{ijl}) \qquad \forall i,j \tag{32}$$

$$-Zd_{ijl} \leq w_{ijl} \leq Zd_{ijl} \qquad \forall i,j \tag{33}$$

$$v^l \geq (1-\mu_j)U_a^{(u)(l)}(j) + \mu_j U_a^{(c)(l)}(j) \qquad \forall j,l \tag{34}$$

5 Using abstracted strategy to guide training RL agents

After solving the abstract game, for each state, we can fit a continuous distribution $\rho(a|s)$ to the discrete action distribution computed by the MILP. Then we follow the routine similar to Soft Actor Critic. Instead of of adding the entropy term $-\alpha \log \pi(a'|s')$ to the Q estimation, we add the K-L Divergence between the trained policy and the abstract game policy ψ, $\alpha \log \frac{\pi(a'|s')}{\psi(a'|s')}$

5.1 Kullback-Leibler divergence

The Kullback-Leibler divergence (KL-Divergence) is a metrics used to measure similarity of two probability distributions. For distributions P and Q of a continuous random variable, the KL-Divergence is defined to be

$$\begin{aligned} D_{KL}(P\|Q) &= \mathbb{E}_P\left[\frac{P}{Q}\right] \\ &= \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx \end{aligned}$$

In this work we use the KL-Divergence between the trained policy and the abstract policy solved using MILP as a regularization term in the Bellman equation.

5.2 KL-Divergence regularized reinforcement learning

Let ψ denote the abstract policy solved using MILP and α denote the regularization coefficient. Then the RL problem we are trying to solve becomes

$$\pi^* = \arg\max_{\pi} E_{\pi}[\sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha D_{KL}(\pi(\cdot|s_t) \| \psi(\cdot|s_t)))]$$

The Bellman equation for Q^π becomes

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_{s'}\left[\mathbb{E}_{a'}\left[R(s,a,s') + \gamma(Q^\pi(s',a') + \alpha D_{KL}(\pi(\cdot|s') \| \psi(\cdot|s')))\right]\right] \\ &= \mathbb{E}_{s'}\left[\mathbb{E}_{a'}\left[R(s,a,s') + \gamma(Q^\pi(s',a') + \alpha\, \mathbb{E}_{a'}\left[\frac{\pi(a'|s')}{\psi(a'|s')}\right])\right]\right] \\ &= \mathbb{E}_{s'}\left[\mathbb{E}_{a'}\left[R(s,a,s') + \gamma(Q^\pi(s',a') + \alpha\left(\frac{\pi(a'|s')}{\psi(a'|s')}\right))\right]\right] \end{aligned}$$

5.3 Q-function update for defender

Since the above Q function is an expectation from the environment dynamics and the policy, it can be approximated with samples.

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_{s'}\left[\mathbb{E}_{a'}\left[R(s,a,s') + \gamma(Q^\pi(s',a') + \alpha\left(\frac{\pi(a'|s')}{\psi(a'|s')}\right))\right]\right] \\ &\approx r + \gamma\left(Q_\phi^{\text{targ}}(s',\tilde{a}') + \alpha\left(\frac{\pi(\tilde{a}'|s')}{\psi(\tilde{a}'|s')}\right)\right) \text{ s.t. } a' \sim \pi(\cdot|s') \end{aligned}$$

Suppose we have a random batch of transitions, $B = (s, a, r, s', d)$ from the replay buffer. We first compute targets for the Q function

$$y = r + \gamma(Q_\phi^{targ}(s', \tilde{a}') + \alpha \log \frac{\pi_\theta(\tilde{a}'|s')}{\psi(\tilde{a}'|s')})$$

note $\tilde{a}' \sim \pi_\theta(\cdot|s')$, is sampled from the current policy. we then update Q-function by one step of gradient descent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} (Q_\theta(s', \tilde{a}') - y)$$

5.4 Policy update for defender

We update the policy to maximize the expected future return plus the expected future KL-Divergence. This quantity J is

$$E_{a \sim \pi_\theta}[Q(s,a) + \alpha \log \frac{\pi_\theta(\tilde{a}|s)}{\psi(\tilde{a}|s)}]$$

Using the reparameterization trick, this term is also equal to

$$E_{\xi\sim N}[Q(s,\tilde{a}_\theta(s,\xi)) + \alpha \log \frac{\pi_\theta(\tilde{a}_\theta(s,\xi)|s)}{\psi(\tilde{a}_\theta(s,\xi)|s)}]$$

Thus, to maximize this term, we apply gradient ascent using the gradient

$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} [Q(s,\tilde{a}_\theta(s,\xi)) + \alpha \log \frac{\pi_\theta(\tilde{a}_\theta(s,\xi)|s)}{\psi(\tilde{a}_\theta(s,\xi)|s)}]$$

5.5 Complete Algorithm

In this section we present the complete algorithm. Algorithm 1 presents the whole training procedure.

Algorithm 1 Training procedure

Initialize policy parameters θ^i, Q-function parameters ϕ_1^i, ϕ_2^i, replay buffer D_i for each agent i
Set target parameters equal to main parameters $\phi^i_{\text{targ},1} \leftarrow \phi^i_1$, $\phi^i_{\text{targ},2} \leftarrow \phi^i_2$
repeat
  for each agent i **do**
    Obtains observation s_i and select action $a_i \sim \pi_\theta(\cdot|s_i)$
  end for
  Execute actions $\{a_1, \cdots, a_n\}$ in the environment
  for each agent i **do**
    Obtains next observation s'_i, reward r_i, and done signal d
    Execute TRAINAGENT_i$(s_i, a_1, \cdots, a_n, r, s'_i, d)$
  end for
  if done signal d is terminal **then**
    Reset environment
  end if
until convergence

Algorithm 2 presents training an agent with centralized Q-functions, double Q trick, and meta strategy.

Algorithm 2 TRAINAGENT_i with meta strategy

1: Input: observation s_i, actions of all agents $a_1, \cdots, a_n$, reward r, next observation s_i', done signal d
2: Store $(s_1, \cdots, s_n, a_1, \cdots, a_n, r, s_i', d)$ in replay buffer D_i
3: **if** it's time to update **then**
4: **for** number of updates **do**
5: Randomly sample $B = (s_1, \cdots, s_n, a_1, \cdots, a_n, r, s_i', d)$ from D
6: Compute targets for the Q-functions:

$$y = r + \lambda(1-d)\left(\min_{k=1,2} Q^i_{\phi^i_{\text{targ},k}}(s_i', \tilde{a_1}', \cdots, \tilde{a_i}', \cdots, \tilde{a_n}') + \alpha \log \frac{\pi_{\theta^i}(\tilde{a_i}'|s_i')}{\psi^i(\tilde{a_i}'|s_i')}\right)$$

$$\text{s.t. } \tilde{a_i}' \sim \pi_{\theta^i}(\cdot|s_i')$$

7: Update Q-functions by one step of gradient descent using

$$\nabla_{\phi^i_k} \frac{1}{|B|} \sum_{(s_i, a_1, \cdots, a_n, r, s_i', d) \in B} \left(Q_{\phi^i_k}(s_i, a_1, \cdots, a_n) - y\right)^2 \text{ for } k = 1, 2$$

8: Update policy by one step of gradient ascent using

$$\nabla_{\theta^i} \frac{1}{|B|} \sum_{(s_i, a_1, \cdots, a_n, r, s_i', d) \in B} \left(\min_{k=1,2} Q_{\phi^i_k}(s, a_1, \cdots, \tilde{a}_{i\theta^i}(s_i), \cdots, a_n) + \alpha \log \frac{\pi_{\theta^i}(\tilde{a}_{i\theta^i}(s_i)|s_i)}{\psi^i(\tilde{a}_{i\theta^i}(s_i)|s_i)}\right)$$

$$\text{s.t. } \tilde{a}_{i\theta^i}(s_i) \sim \pi_{\theta^i}(\cdot|s_i)$$

9: Update target network with

$$\phi_{\text{targ},k} \leftarrow \rho \phi^i_{\text{targ},k} + (1-\rho)\phi^i_k$$

10: **end for**
11: **end if**

6 Experiments

7 Conclusion

References

[1] Bo An et al. "PROTECT – A Deployed Game Theoretic System for Strategic Security Allocation for the United States Coast Guard". In: *AI Magazine* 33.4 (Dec. 2012), p. 96. DOI: 10.1609/aimag.v33i4.2401. URL: https://www.aaai.org/ojs/index.php/aimagazine/article/view/2401.

[2] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *CoRR* abs/1801.01290 (2018). arXiv: 1801.01290. URL: http://arxiv.org/abs/1801.01290.

[3] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning." In: *ICLR*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15.

[4] Ryan Lowe et al. "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments". In: *CoRR* abs/1706.02275 (2017). arXiv: 1706.02275. URL: http://arxiv.org/abs/1706.02275.

[5] James Pita et al. "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*. AAMAS '08. Estoril, Portugal: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 125–132.

[6] Zheyuan Ryan Shi et al. "Towards Thwarting Social Engineering Attacks". In: *CoRR* abs/1901.00586 (2019). arXiv: 1901.00586. URL: http://arxiv.org/abs/1901.00586.

[7] Yevgeniy Vorobeychik, Bo An, and Milind Tambe. "Adversarial Patrolling Games". In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*. AAMAS '12. Valencia, Spain: International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 1307–1308. ISBN: 0981738133.