Multi-agent Deception in Attack-Defense Stochastic Game

Xueting Li

CMU-RI-TR-20-59

Submitted in partial fulfillment of the requirements
for the degree of Master of Science.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

December 2020

Keywords: Attack-defense Game, Multiple Agent Systems, Defense Strategy, Game Theory, Zero-sum Game, Games of incomplete information, Deception

Abstract

This paper studies a sequential adversarial incomplete information game, the attack-defense game, with multiple defenders against one attacker. The attacker has limited information on game configurations and makes guesses of the correct configuration based on observations of defenders' actions. Challenges for multi-agent incomplete information games include scalability in terms of agents' joint state and action space, and high dimensionality due to sequential actions. We tackle this problem by introducing deceptive actions for the defenders to mislead the attacker's belief of correct game configuration. We propose a k-step deception strategy for the defender team that forward simulates the attacker and defenders' actions within k steps and computes the locally optimal action. We present results based on comparisons of different parameters in our deceptive strategy. Experiments show that our approach outperforms Bayesian Nash Equilibrium strategy, a strategy commonly used for adversarial incomplete information games, with higher expected rewards and less computation time.

Acknowledgments

First, I would like to thank my advisor, Professor Katia Sycara, for her patience, insightful suggestions, and guidance for my research. Her vision in the field of multi-agent systems has deeply influenced me. She has taught me how to conduct solid research and how to think critically. I could not have become the researcher I am today without her and I could not have imagined having a better advisor and mentor for my Master study.

I would like to thank Sha Yi for her help throughout my research project. I really enjoyed working with her. She always provides valuable advises and helps me to find more efficient ways to conduct my research.

I also want to thank Professor Changliu Liu for taking the time to be my committee member and for the useful feedback.

Last, I want to thank my family and my friends for your support and encouragement throughout my life and study.

Contents

List of Figures

Chapter 1

Introduction

In applications with adversarial opponents, collaboration between agents can improve the performance of the system, for example, pursuit-evasion games [17] [18], and attack-defense games [2] [15]. In such adversarial scenarios, a game-theoretic formulation provides a framework to model and reason about the benefits and trade-off between players.

Limitations and challenges arise when the game-theoretic framework is applied to infinite horizon games with incomplete information. First, state space grows exponentially with increasing number of agents in the system. Second, with game-theoretic model, Nash Equilibrium only provides solutions to adversarial games with complete information. For incomplete information games, additional belief needs to be incorporated for the unknown game configurations. In the incomplete information setting, Bayesian Nash equilibrium is introduced to solve such game by incorporating players' beliefs. However, the state-belief space then becomes extremely high-dimensional, which is computationally expensive. Third, due to the sequential nature of infinite horizon games, it is intractable to compute the optimal sequence of strategies for the players. In such incomplete information games where there is a information gap between the players, one can use deception to manipulate the belief of opponents[9]. Deception strategies help the players to gain higher rewards in long time horizon, while sacrificing reward gains in short time horizon.

In this paper, we propose a deceptive game-theoretic framework for the infinite horizon, incomplete information games with multiple players. We consider an attack-defense game where there are multiple defenders, a single attacker, and multiple target assets. The true configuration of the target assets i.e.real target values are known to the defender, but only partially known to the attacker. The attacker aims to maximize the total value of targets attacked by guessing the game configuration while avoiding getting caught by any defender. We present techniques that reduce the computational cost of solving the game compared with a standard Bayesian Nash Equilibrium method for incomplete information games. In particular, defenders and the attacker pre-compute a library of Nash strategies, based on the commonly known information, that is exploited during execution. Additionally, we employ the Monte Carlo Tree Search that samples defenders and attacker actions to further reduce the computation cost. We compare the performance and computation cost of our deceptive defense strategy with and without sampling versus a standard one-step BNE strategy.

The paper makes the following contributions: First, we enrich current attack-defense game formulations by considering (a) how defender moves can deceive the attacker and (b) by con-

sidering an infinite-time-horizon game. Second, we provide a novel framework to enable online planning for a multi-defender team in a game with incomplete information.

The paper is organized as follows. Section 2 reviews some previous research on the use of deception in artificial intelligence system and past studies on attack-defense games. In section 3, we first present the formal definition of the attack-defense game we aim to study. We then review definitions of Nash strategy and Nash Equilibrium for computation of the strategy library before execution. We will also introduce Bayesian Nash Equilibrium in section 3.2 as our baseline for comparison. Monte Carlo Tree Search will be introduced in section 3.5 for improving scalability and computation efficiency. Section 4 shows experimental results of the defender team performance and computation cost when the defender team follows BNE or our deceptive planning against an attacker who follows BNE or the strategy described in section 3.3.

Chapter 2

Related Work

Deception is studied in an incomplete information game setting where one player has private information of the world configuration. It is mostly studied in the field of Cybersecurity games. In [14], [7] and [13], the authors study one-shot game-theoretical use of honeypot in cybersecurity. [3] introduces a game-theoretic model for optimal deployment of honeypots into the network for a finite-horizon multi-stage game. [6] studies an infinite-horizon deception strategy, where the user trying to deceive the cyber-attackers by manipulating their beliefs of whether the attackers has been detected or not.

In multi-agent games, both deceptions by attacker and defender have been studies. In [9], the authors consider a situation where an adversary apply deception by camouflage the targets to confuse a sensor team from performing its tasks. They employ a game-theoretic model to analyze the expected strategy of the adversary and find the best response. In attack-defense games, deception is often studied in signal games[20] where the defender deceives the attacker by sending signals of defender type or resource allocation game[19] where deceptive resources are allocated to deceive the attacker. In [12] and [11], attacker deception is considered in a Stackelberg game for calculating the optimal strategy for both defenders and the attacker.

Games where a fraction of players hold incomplete information are considered as asymmetric games. Asymmetric games have been widely studied in the literature. In [8], the author provides an efficient linear programming formulation of asymmetric two-player zero-sum stochastic games with finite horizon. In [4], the author describes an efficient algorithm for solving the value of two-person zero-sum repeated games of incomplete information within arbitrary accuracy. To the best of our knowledge, the current literature does not provide an efficient algorithm to address the computational intractability of the belief state space in asymmetric two-player zero-sum stochastic games with infinite horizon.

Chapter 3

Technical Approach

We first formally define our problem in section 3.1. In section 3.2, we review details of Nash Equilibrium of complete information games for later computation of the strategy library, and the standard Bayesian Nash Equilibrium methods of incomplete information games for comparison in chapter 4. We will introduce our assumptions and methods of attacker belief update in section 3.3, and our k-step deception planning method in section 3.4. Monte Carlo game tree sampling will be discussed in section 3.5 for improving computational efficiency.

3.1 Problem Formulation

We model the game as a two-player zero-sum game with incomplete-information and infinite time horizon. We discretize time and space and represent the environment as a grid world. Both the attacker and defenders have information of the target locations and can always observe the adversary's movement. The game terminates when (i)the attacker is caught by any defender, or (ii) all valuable targets have been attacked. The attacker is considered caught if it is on the same cell as any defender and a target is attacked if the attacker is on the same cell as the target. Throughout the game, the attacker will constantly try to guess the true target configuration. Meanwhile, the defender team will move in a way to deceive the attacker about the true target configuration and mislead the attacker to go to targets of lower value.

We model the game as a Markov Decision Processes (MDP) $< S, A, T, \Phi, \mathbf{R}, \mathbf{b}, \gamma >$. S is the joint state space of defenders, attacker, and targets $S = S_d \times S_a \times S_w$. $S_d = S_0 \times \cdots \times S_n$ where S_i is the state of i^{th} defender. S_i and S_a are the positions of corresponding agent in the grid world. S_w is a binary vector where $S_{w_j} = 1$ if j^{th} target has not been attacked, 0 otherwise. A is the joint action space of the defender team and attacker. $A = A_d \times A_a$ where $A_d = A_1 \times A_2 \times \cdots \times A_n$. A_i is the action space of i^{th} defender and A_a denotes the action space of the attacker. We assume four-connected grid so the action space of each agent is $A_i, A_a = \{N, E, W, S, Z\}$, which corresponds to move to North, East, West, South, and remain in the current cell. T is the deterministic transition matrix that $T(s'|s, a) = 1$ if the following state of taking action a at state s is s', 0 otherwise.

In our game, there is a discrete set of target rewards given different target configurations Φ. Each $\phi_l \in \Phi$ represents a reward set of the l^{th} target configuration. Each ϕ_l consists of

the positions of targets and rewards for attacking different targets. The positions of targets are accessible for both players. However, only the defender team knows the rewards. The real target configuration is the one with the true rewards of targets. The reward set of the real target configuration ϕ_{l^*} is included in this set, i.e. $\phi_{l^*} \in \Phi$. We assume both teams have common knowledge on all possible reward setup. Given the number of targets, there is a finite set of possible target configuration that is known by both players.

The goal of the defenders is to catch the attacker and minimize the total value of targets being attacked. The goal of the attacker is to maximize the value of targets attacked and avoid getting caught by any defender.

$\mathbf{R}$ is the reward function set. For each ϕ_l, there is a distinct reward function $R_l : S \times A \to \mathbb{R}$ corresponding to that target configuration. $R_l(s, a)$ is the reward of defenders' team when joint action a has been taken in the joint state s:

$$R_l(s'|s,a) = \begin{cases} -g^j_{l,w} & \text{if } j^{th} \text{ target is attacked at state } s' \\ g_{l,c} & \text{if attacker is caught by any defender at state } s' \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$\mathbf{g_{l,w}}$ is a vector consists of the values of different targets where $g^j_{l,w}$ is the value of j^{th} target in l^{th} target configuration. $g_{l,c}$ is the value of catching the attacker. Since it is a zero-sum game, the reward for the attacker is thus $-R_l$. The reward function of the real target configuration is denoted as $R_{l^*} \in \mathbf{R}$. The expected future reward is discounted over an infinite time horizon where γ is the discount factor. Both players aim to maximize their expected future reward. The defender team's objective is to maximize the expected discounted reward:

$$\max_{a^0_d \cdots a^\infty_d} \sum_{t=0}^{\infty} \gamma^t R_{l^*}(s_t, a_{d,t} \times a_{a,t}) \tag{3.2}$$

Not knowing the true target configuration, the attacker maintains a belief vector $\mathbf{b} = [b_0, \cdots, b_l, \cdots]$ where $b_l = prob(\phi_l = \phi_{l^*})$ denotes the probability the attacker thinks ϕ_l is the true target configuration. The attacker updates $\mathbf{b}$ based on the observed moves of the defender team, which will be introduced in section 3.3. The defender team can reconstruct the attacker's belief because all information on the attacker side is available to the defender team. This is commonly used to keep track of adversary belief when using deception [6].

3.2 Nash Equilibrium and Bayesian Nash Equilibrium

Before execution, we pre-compute a strategy library based on Nash Equilibrium of all possible configurations for both the defender team and the attacker. During execution, we utilize the Nash values for deceptive planning in section 3.4. Nash equilibrium is used to model the interaction between multiple agents when all players have complete information of the game and aim to maximize their own expected payoffs. The extension of Nash equilibrium for incomplete information game is Bayesian Nash equilibrium[5] where players incorporate the belief of unknown information. We will show by comparison in section 4 that our k-step deception method outperforms the standard Bayesian Nash equilibrium.

By definition, a Nash equilibrium occurs when no player can do better by unilaterally changing its strategy[10]. Given the full target configuration, i.e. both locations and values, the optimal strategy for each player is to follow the Nash strategy of the complete information game. A Nash strategy can be either a pure strategy (deterministic) or a mixed strategy(stochastic). We consider mixed strategy here because in a single-stage zero-sum game with finite state-action space, a mixed strategy always exits whereas a pure strategy might not[10]. At each time step, each player selects an action based on the probability distribution of the computed mixed strategy. A mixed strategy for the defender team is denoted as $x : S \times A_d \rightarrow [0, 1]$. Similarly, a mixed strategy for the attacker is $y : S \times A_a \rightarrow [0, 1]$. It has been proven in [16] that in a two-player zero-sum finite-state space infinite-horizon stochastic game, there exists a unique Nash value for the complete information game and a unique mixed Nash strategy can be induced from this Nash value. This uniqueness ensures that in a given state of a known target configuration, the attacker knows the exact defender strategies. Thus, based on a specific target configuration, both the attacker and defender team can compute the strategy of one another.

Consider $V(x, y, s)$ as the defender's discounted expected reward obtained by the defenders' and attacker's strategy pair (x, y) starting at state s. The discounted expected reward for the attacker is thus $-V(x, y, s) : S \rightarrow \mathbb{R}$. We denote the Nash strategy of the defender team and attacker to be x^* and y^* correspondingly. To differentiate Nash strategies among different target configurations, we use x_l^*, y_l^* to denote the Nash strategies of l_{th} target configuration. The Nash value $V(x_l^*, y_l^*, s)$ is then the defenders' value obtained from players that follow Nash strategy x_l^* and y_l^* at state s in target configuration ϕ_l. For each $s \in S$, the Nash value satisfies:

$$V(x_l^*, y_l^*, s) \geq V(x, y_l^*, s), \forall x \in X \tag{3.3}$$

$$V(x_l^*, y_l^*, s) \leq V(x_l^*, y, s), \forall y \in Y \tag{3.4}$$

In a zero-sum game, the solution to find the Nash value at single state can be formulated as a min-max problem:

$$V(x_l^*, y_l^*, s) = \min_y \max_x \sum_{a \in A} \left[x(a_d|s) y(a_a|s) \sum_{s' \in S} T(s'|s, a) \left(R_l(s, a) + \gamma V(x^*, y^*, s') \right) \right]$$

where $a = a_d \times a_a$, $x(a_d|s)$ and $y(a_a|s)$ are the probability of the defender team taking action a_d, and attacker taking action a_a at state s correspondingly. With the above update equation, we can obtain the Nash values at each game state with different target configurations.

To enable online planning for both players in section 3.3 and 3.5, we further define a Nash policy Set Π^* which contains Nash policies corresponding to different target configurations ϕ_l. More specifically, for each $\phi_l \in \Phi$, there is a $\pi_l^* = (x_l^*, y_l^*) \in \Pi^*$ where π_l^* is the Nash strategy pair of the defender team and attacker given the target configuration ϕ_{l^*}. Let $\pi_{l^*}^* = (x_{l^*}^*, y_{l^*}^*), \pi_{l^*}^* \in \Pi^*$ denote the policy corresponding to the true target configuration ϕ^*. For simplicity, let $Nash_{l^*}(s) = V(x_{l^*}^*, y_{l^*}^*, s), s = s_d \times s_a \times s_w$ denote the Nash value of a state s when all players know the true target configuration and play best strategies.

An extension of Nash equilibrium to incomplete information game is a Bayesian Nash equilibrium (BNE). BNE is not part of our approach, but will serve as a baseline of comparison with our k-step deception method in section 4. Extending from Equation (3.3) and (3.4) to the BNE

formulation, the uninformed player, in our case the attacker, computes BNE based on its belief of the target configuration:

$$BNE(x_{l^*}, y_{l^*}, s, \mathbf{b}) =$$
$$\min_y \max_x \sum_{a \in A} \left[x(a_d|s) y(a_a|s) \sum_{s' \in S} T(s'|s,a) \sum_{R_l \in \mathbf{R}} \mathbf{b}_l \left(R_l(s,a) + \gamma BNE(x^*, y^*, s') \right) \right]$$

Importantly, the action spaces, the reward functions, possible target configurations, and the belief of the uninformed player are assumed to be common knowledge. That is to say, the informed player(defender team) can compute what the attacker strategy is based on attacker's belief. With that in mind, the optimal strategy of the defender team can then be simply calculated as an optimization function based on the true reward function and attacker's strategy.

3.3 Attacker Strategy and Belief Update

Since both players (defender team and the attacker) assume their adversary would play optimally and the attacker does not know whether the defender team might take deceptive action to manipulate its belief, the attacker updates its belief about each target configuration based on its observation of defenders' last actions. We assume the attacker's initial belief of the target configuration is uniformly distributed as $b_l = \frac{1}{|\Phi|}, \forall l$. After actions (a_d, a_a) have been executed at state s, the attacker updates its belief based on defenders' action a_d:

$$b'_l = prob(\phi_l|a_d, s) = prob(x^*_l|a_d, s) = \alpha \cdot b_l \cdot x^*_l(a_d|s) \tag{3.5}$$

$prob(\phi_l|a_d, s) = prob(x_l|a_d, s)$ because for a given target configuration ϕ_l, the defenders will play optimally, namely the defender team's Nash strategy x^*_l. α is a normalizing factor such that $||\mathbf{b}|| = 1$. We assume the attacker plays a strategy based on its updated belief $\mathbf{b}$ and corresponding Nash strategy of each ϕ_l (which we later referred to as the belief-based Nash Strategy):

$$y(a_a|\mathbf{b}, s) = \sum_{\pi_l \in \Pi} b_l \cdot y^*_l(a_a|s) \tag{3.6}$$

where $s = s_d \times s_a \times s_w$. $y^*_l(a_a|s)$ is the probability of executing a_a at state s with strategy y^*_l. $y(a_a|\mathbf{b}, s)$ is the attacker strategy during execution, which is not only based on the joint state s but also based on the attacker's belief of the target configuration. Since the defender team knows 1) the target configuration set, 2) all actions are fully observable, 3) the attacker will update its belief based on defenders' action, the defender team can perfectly reconstruct the belief of the attacker. We will utilize this feature during our k-step deception method for the defender team in the following section.

3.4 Deceptive Planning

An overview of our method is shown in figure 3.1. Before execution, we pre-compute the Nash strategy library based on all possible target configurations. During execution, at each time step,

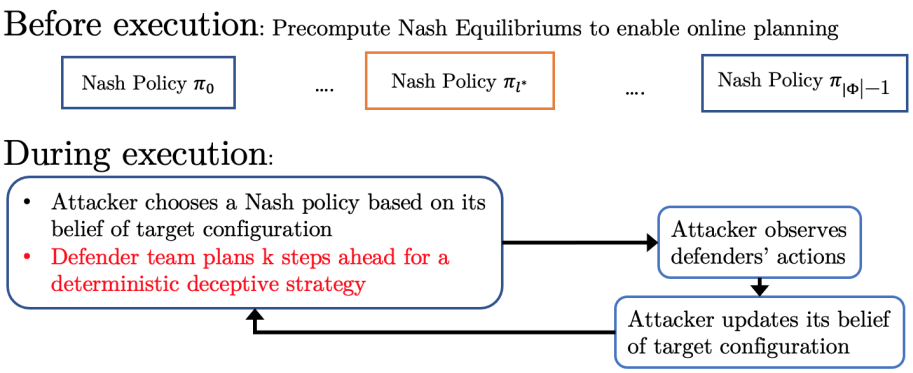


Figure 3.1: Flow diagram of our method. Before execution, we pre-compute the Nash equilibrium policies. During execution, the attacker tries to play Nash strategy based on its belief. The defender team tries to deceive the attacker of the target configuration.

the attacker takes an action based on its belief of the true target configuration. Since the defender team can perfectly reconstruct the attacker's belief, the defenders will forward simulate all actions k steps ahead and plays the deceptive action that gives the estimated highest value. After both teams take actions simultaneously, the attacker updates its belief of the true target configuration based on its observation of the defenders' actions.

Algorithm 1 shows the detailed computation of the k-step deceptive strategy. k is a user-defined constant that determines how many steps the user would like to simulate forward. This algorithm first builds a tree of depth k to explore all deterministic moves the defender can take in k time steps and all the attacker beliefs $\mathbf{b}$ induced by those actions. The attacker's actions are simulated in a stochastic way based on their belief of the target configurations and the corresponding Nash strategies. We estimate the value of the leaf states by their Nash value in the true target configuration as if both players were to play a Nash equilibrium in complete information games from that state forward. Then the algorithm evaluates each k-depth action sequence by back-propagating the Nash values of the leaf states. The first action in the action sequence that has the highest expected value is then executed by the defender team.

To guarantee the deceptive action is not worse than the original Nash strategy, we define $Decep(s, \mathbf{b}, n)$ to be the expected discounted reward for the defender team, at state s and attacker belief $\mathbf{b}$, to take a deceptive action sequence with length n, i.e. actions within n time steps. Denote $Decep^*(s, \mathbf{b}, n)$ to be the maximum of all $Decep(s, \mathbf{b}, n)$ among all legal action sequences of length n. $Decep^*(s, \mathbf{b}, n)$ is calculated as follows:

$$Decep^*(s, \mathbf{b}, 0) = Nash_{l^*}(s) \tag{3.7}$$

$$Decep^*(s, \mathbf{b}, k+1) = \max_{a_d} T(s, a, s')y(a_a|\mathbf{b}, s)(R^*(s, a) + \gamma Decep^*(s', \mathbf{b}', k)) \tag{3.8}$$

In Theorem 13, we show that $Decep^*(s, \mathbf{b}, n)$ will always be greater or equal to the Nash value of state s. for all $n \in Z^+$,

$$Decep^*(s, \mathbf{b}, n) \geq Nash_{l^*}(s), \forall \mathbf{b} \tag{3.9}$$

The detailed proof is can be accessed at the link here . Based on the theorem above, we optimize defender's local payoff by finding a local optimal k-step pure strategy $(a^1, a^2..., a^k)^*$ with

Algorithm 1: kStepDeception finds the best action and maximum expected value

Input: State s, Attacker Belief $\mathbf{b}$, Step k
Output: Best action and corresponding value

1 $maxValue \Leftarrow Nash^*_{l*}(s)$;
2 $bestAction \Leftarrow x^*_{l*}(s)$;
3 **if** $k > 0$ **then**
4 **for** $a_d \in A_d$ **do**
5 $\mathbf{b}' \Leftarrow updateBelief(\mathbf{b}, a_d)$;
6 $value \Leftarrow \sum_{a_a \in A_a, s' \in S} [p(a_a|\mathbf{b}, s) \cdot T(s'|s, a_d, a_a) \cdot kStepDeception(s', \mathbf{b}', k-1).maxValue]$;
7 **if** $value > maxV$ **then**
8 $maxValue \Leftarrow value$;
9 $bestAction \Leftarrow a_d$
10 **end**
11 **end**
12 **end**
13 **return** *bestAction,maxValue*;

value $Decep^*(s, \mathbf{b}, k)$. From the proof we also know there exits at least one such pure strategy $(a^1, a^2..., a^k)$.

Although the deceptive action returned from the k-step deception algorithm may result in a lower reward in the current state compared with a Nash strategy, by simulating k step forward into the future, our deceptive action gives higher reward in the longer time horizon. With this method, it is guaranteed that the action sequence returned from our k-step deception algorithm will always gives higher, or equal, expected discounted reward compared with Nash strategy. However, explore the full game tree of depth k may still be computationally heavy as the state space increases. We incorporate Monte Carlo sampling methods into our approach in the coming section.

3.5 Game Tree Sampling

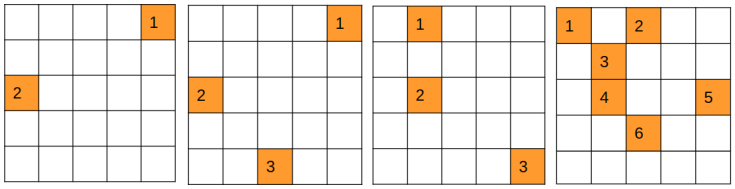
In Algorithm 1, we built a search tree with all valid defender and attacker moves. The tree size (number of states) grows exponentially with the action space of the agents: $|s \in Tree_k| = (|A_d| \cdot |A_a|)^k$, where $Tree_k$ is the search tree with step size k. With multiple defenders, the number of defender actions $|A_d|$ grows exponentially with the number of defenders. In order to improve the scalability of our approach, we adopt the idea of Monte Carlo Tree Search (MCTS)[1]. Inspired by MCTS, we sample m actions at each step instead of exploring all $a_d \in A_d$ (at line 4 in Algorithm 1). The size of the search tree is thus $|s \in Tree_k| = (m \cdot |A_a|)^k$. We compare two different sample criteria: (1): Sample actions based on their probability in the Nash strategy π^* : $p(a_d|s) = x^*_{l*}(a_d|s)$. (2): Sample actions based on uniform distribution: $p(a_d|s) = \frac{1}{|A_d|}$

Apart from the two above mentioned sample criteria for the defender team, to further improve the computation efficiency, we also sample attacker actions during our forward simulation (at line 6 in Algorithm 1). We compare two different sampling criteria: (1): Sample actions with maximum probability in Nash strategy: $a_a = \arg\max y^*_{l*}(a_a|s)$. This decreases the tree size from $(m \cdot |A_a|)^k$ to m^k. (2): Sample the top H percent actions based on the probability of actions from largest to the lowest: $\sum y^*_{l*}(a_a|s) = H$.

Chapter 4

Results

We extensively ran simulations based on the configurations in figure 4.1. In our experiments, we assume there is only one real target and all players are aware of this information. We set $g^j_{l,w} = 5$ when j^{th} target is the real target, 0 otherwise. The catch reward $g_{l,c} = 5$. Unless explicitly stated, $k = 2$ for all deceptive planning. We randomized agents' initial positions and compared the defender payoff and computation cost among different methods in section 3.5.

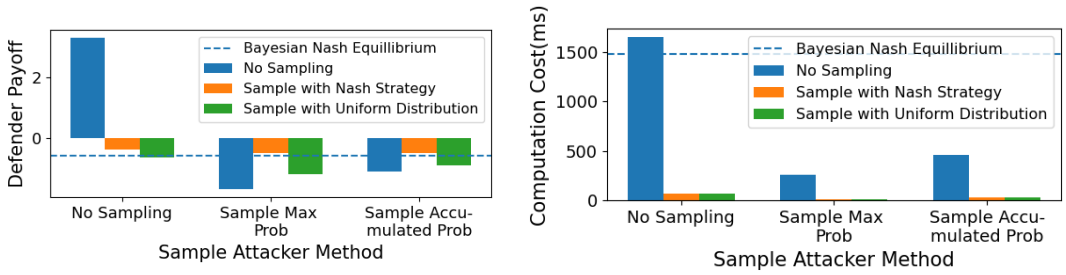


(a) (b) (c) (d)

Figure 4.1: Experiment configurations.

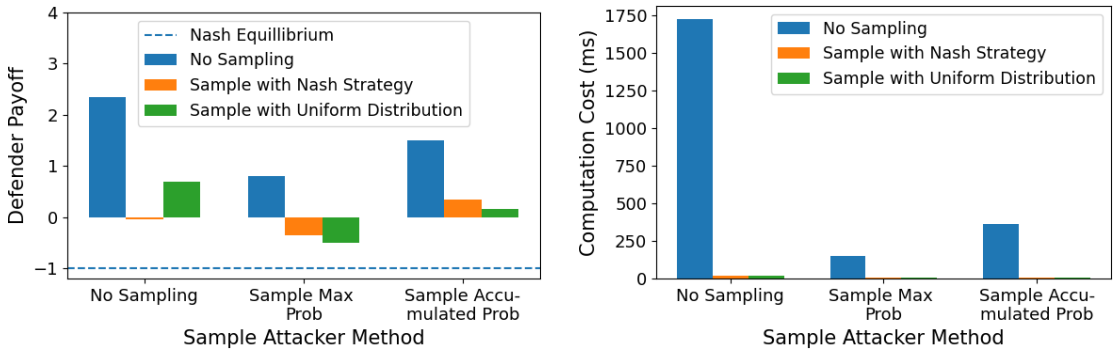
We ran the experiments on an Intel i7 Quad-Core machine with 16 GB memory. The code is written in python and we use scipy.optimize.linprog as the linear programming solver for solving the (Bayesian) Nash equilibrium. We ran six sets of experiments, detailed settings of each set of experiments are as follows: (1) Defender team: three different deceptive strategies or one-step BNE strategy; Attacker: belief-based Nash Strategy; Configuration: 4.1a; (2) Defender team: three different deceptive strategy; Attacker: one-step BNE strategy or belief-based Nash Strategy; Configuration: 4.1a (3) Defender team: deceptive strategy with three different sampling method; Attacker: belief-based Nash Strategy; Configuration: 4.1b and 4.1c (4) Defender team: deceptive strategy with two different k values; Attacker: belief-based Nash Strategy; Configuration: 4.1b and 4.1c (5) Defender team: deceptive strategy with three different sample methods and sample size; Attacker: belief-based Nash Strategy; Configuration: 4.1b and 4.1c (6) Different size of possible target configuration set; Defender team: deceptive strategy without sampling; Attacker: belief-based Nash Strategy; Configuration: 4.1d.

First, we compare the performance and computation cost between a traditional one-step BNE approach and our $k = 2$ deceptive strategy. For the sake of fairness, we also use the pre-computed Nash values as a state estimation function for the calculation of BNE. Since the computation cost of BNE grows exponentially with the increase of actions and target configurations, it is not solvable when there are more than two target configurations. As a result, experiment set 1 is run on configuration 4.1a, resulting in two possible target configurations: (1) $g^1_{l,w} = 5, g^2_{l,w} = 0$, (2) $g^1_{l,w} = 0, g^2_{l,w} = 5$. As shown in figure 4.2b, the computation cost of our 2-step full-planning with



(a) (b)

Figure 4.2: Results of experiment set 1: Different Action Sample Methods for configuration 4.1a: (a) Defender Payoff (b) Computation Cost



(a) (b)

Figure 4.3: Experimental Results with Different Action Sample Methods for configuration 4.1b and 4.1c: (a) Defender Payoff (b) Computation Cost

deception is similar to that of a single-step BNE. However, in figure 4.2a, the deceptive strategy has a significant performance increase with defender payoff of 3.3 while BNE only has a payoff of -0.6. The sampling methods have similar defender payoff compare to BNE.

The second experiment set was run based on configuration 4.1a and the same two possible target configuration as the first experiment set. When we plan without sampling, the average defender payoff decreases slightly from 3.45 to 3.3 when the attacker change strategy from belief-based Nash Strategy to BNE. This implies that our no-sampling approach is robust against a smarter adversary(an attacker who follows BNE) even we do not simulate attacker to follow BNE during planning (high computation cost). However, if we plan with sampling, the average defender payoff decreases dramatically from 1.212 to -0.868. The mismatch between the attacker strategy we assume and real attacker strategy makes the sampling methods weaker.

In the third experiment set, we compare the performance of different sampling methods. We can see that all deceptive methods have better performance than the Nash Strategy. Full exploration of defender actions show the best performance but also the highest computation cost (in figure 4.3a). Figure 4.3a shows that simulating single action with maximum probability has the worst performance. Simulating the actions that take 75% of the total probability has a similar performance as simulating all attacker actions. Regarding computation cost, figure 4.3b shows

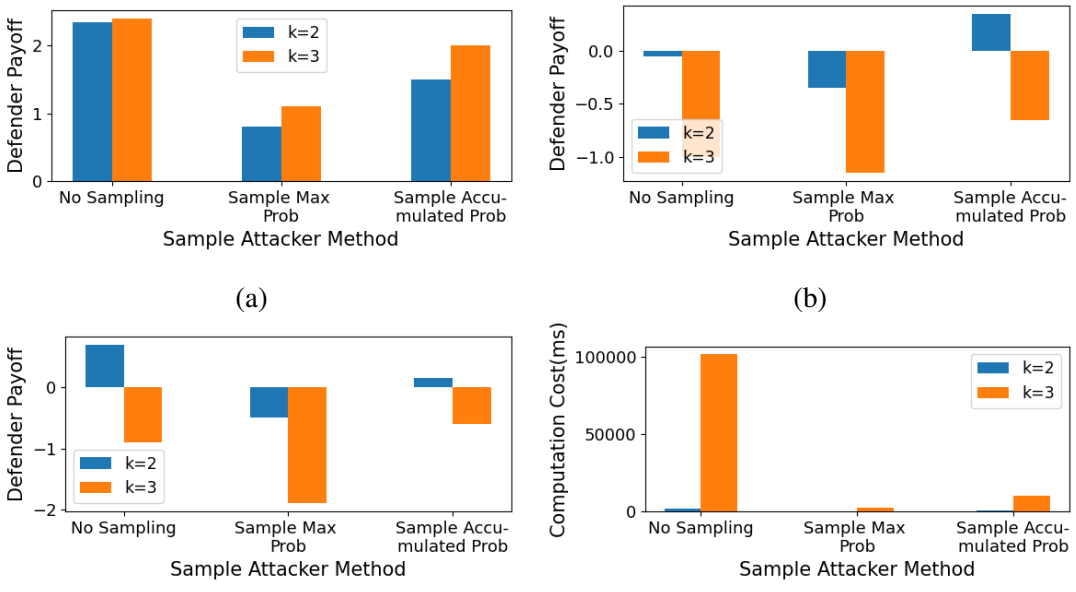


Figure 4.4: Results of experiment set 4: (a)-(c) Defender Payoff with Different Attacker Action Sample Methods When Defender Sample Method is (a)no sampling (b) sample based on Nash strategy (c) sample based on uniform distribution ;(d) Computation Cost with Different Attacker Action Sample Method when all defender actions are considered(no sampling)

that the computation cost of sample the top $H = 75\%$ percent actions is approximately 20% to 40% of the cost of full simulation.

In fourth experiment set, we investigate the influence of the value of k from figure 4.4. Figure 4.4a shows slight performance increase with the increase of k when we explores all legal defender moves. Figure 4.4b and 4.4c shows that sampling methods has significant performance decrease with k value. It is because when the search tree grows with k, the sampling methods lead to less accurate evaluation of the states. Although we have some performance increase in figure 4.4a, figure 4.4d shows that the computation cost of $k = 3$ is approximately 100 times more than that of $k = 2$ but the performance increase is only around 2%.

In fifth experiment, we investigate the influence of different defender action sample methods

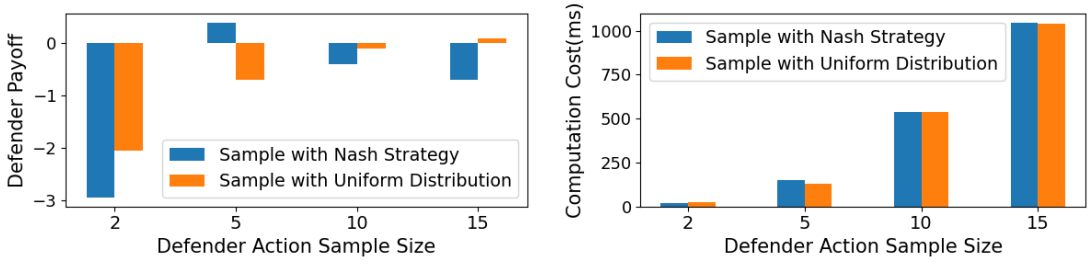
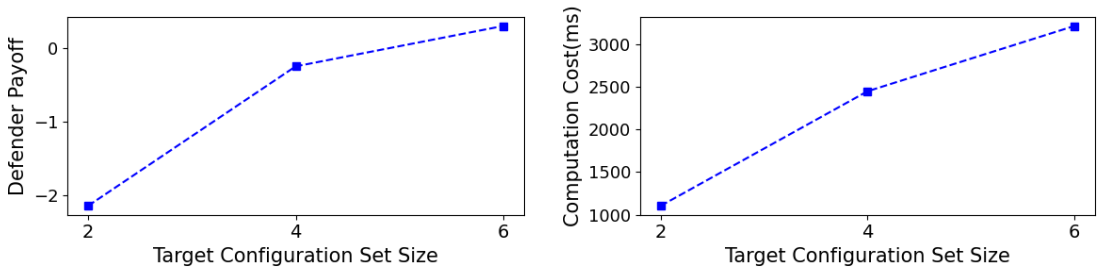


Figure 4.5: Results of experiment set 5: Different Defender Action Sample Method and Different Sample Size for configuration 4.1c: (a) Defender Payoff (b) Computation Cost



(a) (b)

Figure 4.6: Results of experiment set 6: Experimental Results with Different Target Configuration Set Size (a) Defender Payoff (b) Computation Cost

and different sample size. Figure 4.5a shows that for Nash strategy based sampling, sample size 5 has the best performance. For uniform sampling, the performance increase with sample size. However, the computation cost also increases rapidly. Overall, Nash strategy based sampling with sample size 5 has the best performance with a low computation cost.

Last, we tested the scalability of our approach regarding the size of target configuration space. The real target in each simulation is always target 2. The possible real target indexes in each simulation are: when $|\Phi| = 2$: [2,5] ;when $|\Phi| = 4$: [2,4,5,6] ;when $|\Phi| = 6$: [1,2,3,4,5,6]. Figure 4.6b shows the computation cost increase slowly with the size of possible target configurations. Figure 4.6a shows defender team can gain more payoff when there are more possible target configurations.

Chapter 5

Conclusion and Future Research

In this paper, we studied the attack-defense game in which the attacker has incomplete information of the game configuration. We have proposed a k-step deception algorithm in which the defender team can generate a deceptive strategy based on pre-computed Nash strategy library and forward simulation of game tree. Comparing our algorithm to the traditional Bayesian Nash Equilibrium, our algorithm can handle larger action space and larger target configuration space. Results from experiments show that our method is more computationally efficient and receives larger payoff in the game.

From our experiments, we observe that as k increases, the computation cost increases exponentially, but may not necessarily leads to higher reward gain. Choosing the value of k remains an open problem and it is worth investigating if there is a optimal k value given a specific game configuration and user preference of computation limits. Additionally, choosing the proper methods of sampling in the game tree influences the policy performances. Further research can be done with exploring more methods of sampling.

Bibliography

[1] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-carlo tree search: A new framework for game ai. In *AIIDE*, 2008.

[2] Z. Deng and Z. Kong. Multi-agent cooperative pursuit-defense strategy against one single attacker. *IEEE Robotics and Automation Letters*, 5(4):5772–5778, 2020.

[3] K. Durkota, V. Lisỳ, B. Bošanskỳ, and C. Kiekintveld. Optimal network security hardening using attack graph games. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[4] A. Gilpin and T. Sandholm. Solving two-person zero-sum repeated games of incomplete information. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 903–910, 2008.

[5] J. C. Harsanyi. Games with incomplete information played by "bayesian" players, i–iii part i. the basic model. *Management science*, 14(3):159–182, 1967.

[6] K. Horák, Q. Zhu, and B. Bošanskỳ. Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security. In *International Conference on Decision and Game Theory for Security*, pages 273–294. Springer, 2017.

[7] C. Kiekintveld, V. Lisỳ, and R. Píbil. Game-theoretic foundations for the strategic use of honeypots in network security. In *Cyber Warfare*, pages 81–101. Springer, 2015.

[8] L. Li and J. Shamma. Lp formulation of asymmetric zero-sum stochastic games. In *53rd IEEE Conference on Decision and Control*, pages 1930–1935. IEEE, 2014.

[9] V. Lisỳ, R. Zivan, K. Sycara, and M. Pěchouček. Deception in networks of mobile sensing agents. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1031–1038, 2010.

[10] J. Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[11] T. Nguyen and H. Xu. Imitative attacker deception in stackelberg security games. In *IJCAI*, pages 528–534, 2019.

[12] T. H. Nguyen, Y. Wang, A. Sinha, and M. P. Wellman. Deception in finitely repeated security games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2133–2140, 2019.

[13] J. Pawlick, E. Colbert, and Q. Zhu. Modeling and analysis of leaky deception using signaling games with evidence. *IEEE Transactions on Information Forensics and Security*, 14(7):1871–1886, 2018.

[14] R. Píbil, V. Lisỳ, C. Kiekintveld, B. Bošanskỳ, and M. Pěchouček. Game theoretic model of strategic honeypot selection in computer networks. In *International Conference on Decision and Game Theory for Security*, pages 201–220. Springer, 2012.

[15] S. Sengupta and S. Kambhampati. Multi-agent reinforcement learning in bayesian stackelberg markov games for adaptive moving target defense. *arXiv preprint arXiv:2007.10457*, 2020.

[16] L. S. Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[17] D. Sincák. Multi–robot control system for pursuit–evasion problem. *Journal of electrical engineering*, 60(3):143–148, 2009.

[18] R. Vidal, S. Rashid, C. Sharp, O. Shakernia, J. Kim, and S. Sastry. Pursuit-evasion games with unmanned ground and aerial vehicles. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 3, pages 2948–2955. IEEE, 2001.

[19] Y. Yin, B. An, Y. Vorobeychik, and J. Zhuang. Optimal deceptive strategies in security games: A preliminary study. In *Proc. of AAAI*, 2013.

[20] J. Zhuang, V. M. Bier, and O. Alagoz. Modeling secrecy and deception in a multiple-period attacker–defender signaling game. *European Journal of Operational Research*, 203(2):409–418, 2010.