Online Learning in Stackelberg Security Games

Guanda Chen

August 2022

Abstract

This paper investigates the problem of finding the optimal strategy for the defender in Stakelberg security games. By solving multiple linear programming problems, we could calculate the optimal strategy and it suffers from long computation time when dimension is high. To address this issue, we propose a solution, which involves solving mixed-integer linear programming problems to calculate the optimal strategy, significantly reducing the computation time. We conduct multiple numerical experiments and verify the accuracy of the algorithms. In addition, we apply the online combinatorial optimization algorithm "Follow the Leader" to repeated security games and dynamically adjust the leader's strategy by learning from the follower's behavior to achieve optimization goals. We also apply an improved algorithm called "Follow the Perturbed Leader" to solve the problem of failing to converge to the optimal strategy in extreme cases. We conduct theoretical analysis and experimental verification of the algorithm, derive the curve of its regret bound, and prove its effectiveness through multiple numerical experiments. Finally, we design a special example to demonstrate that the improved algorithm can converge to the optimal strategy even when the "Follow the Leader" algorithm fails to converge and prove that its regret is sublinear with respect to time. In conclusion, both the proposed solutions and the improved algorithm can effectively calculate the optimal strategy for the defender and are validated through multiple numerical experiments. These methods have practical value and can provide scientific guidance for decision-making in the security field.

Keywords: Stakelberg security game, Linear programming, Mixed-integer linear programming, Follow the leader, Follow the Perturbed leader

Contents

1 Introduction

In the Satkelberg Game, there exists a leader and a follower, where the leader will make a strategy and the follower will make the best response according to the leader's strategy. In the Stakelberg Security Game, the leader is often a security agency, also known as the defender in this paper, who can make decisions in a strategy space and then allocate resources using a mixed strategy to protect potential targets that may be attacked based on the information collected. After observing the defender's mixed strategy, the follower, referred to as the attacker in the Stakelberg Security Game (SSG), calculates the expected payoff of attacking each target based on the defender's mixed strategy and utility matrix and then chooses the optimal target to attack. Based on the mixed strategy, the potential targets that may be attacked are allocated resources by the defender to protect them, and there is a probability that the target is protected and successfully prevent the attacker's attack, which will affect the payoff of both the attacker and the defender.

1.1 Model and Optimization Ojbective

In our SSG model, the defender has full knowledge of the attacker's payoff matrix, which allows the defender to calculate the optimal protection strategy for each potential attacker. However, in real-world SSGs, the defender faces uncertainty about the attacker's type, which necessitates the incorporation of randomness into the model. To address this challenge, we propose a novel design in which a sequence of potential attackers is provided, and in each round of the SSG, one attacker type in the sequence confronts the defender. Since the attacker type changes from round to round, it is difficult to make optimal decisions without knowing the future attacker types. In response, we leverage the concept of online learning to develop an algorithm that utilizes the information from each round of SSG to compute an optimal protection strategy. We also calculate the hindsight best strategy after each round of attack. The defender then applies both strategies and calculates the difference in payoff, which we refer to as regret. If we can prove that the regret is sublinear in time, we can demonstrate the convergence of our online learning algorithm to the optimal strategy.

1.2 Overveiw and Results

We used two different methods to calculate the defender's optimal mixed strategy. One method is to solve multiple linear programming problems, but it has disadvantages of high complexity and long computation

time in high dimensions. Therefore, we use another method to solve mixed integer linear programming to efficiently calculate the defender's optimal mixed strategy. We also conduct multiple numerical experiments to verify the accuracy of the calculation results.

We apply a novel approach to address the challenge of handling different types of attackers in Stakelberg Security Games. We apply the "Follow the Leader" algorithm to compute an "optimal strategy" in each round and it could converge to the optimal strategy in most cases. However, this method has a disadvantage: the regret obtained by this algorithm increases linearly with time in extreme cases, so it does not satisfy sublinearity and cannot obtain the optimal strategy. Therefore, we proposed another algorithm, Follow the Perturbed Leader, which solves this problem. We designed several extreme examples to test its convergence and proved the sublinearity of the regret obtained by this algorithm through theoretical derivation.

2 Preliminaries

In the Stakelberg security game, there are two players: the defender and the attacker. The attacker can observe the defender's defense strategy and make decisions that maximize their expected payoff. We are interested in a scenario where the Stakelberg security game is repeated, and the attacker will attack a set of potential targets $[L] \triangleq \{1, \ldots, L\}$. The attacker types are in a set $[K] \triangleq \{1, \ldots, K\}$, and the type of attacker in each round is determined by a given sequence. In each round of the game, the attacker will attack the target that can yield the highest expected payoff based on the available information, and the payoff will depend on whether the target is defended or not, denoted by c_j^i and u_j^i, where c_j^i, u_j^i is the payoff of attacker j when target i covered, not covered by the defender. In the security game, defenders and attackers know full information of their payoffs, when it comes to tie-breaking, attackers will attack the target with the defender's best-expected payoff.

This scenario's Stakelberg security game is a dynamic game that considers the impact of each round's result on the next round. In this scenario, the attacker needs to devise a strategy to maximize their expected payoff, while the defender needs to devise a strategy to protect the targets as much as possible.

2.1 Notations

- Time horizon is T. The time index of each round is $t \in \mathbb{T}$.
- Attacker:

— Set of attacker types $[K]$. Distribution of attacker type could be represented by $probability\ vector$ $\mathbf{q} = (q_1, \ldots, q_k, \ldots, q_K)$. q_k is the probability that the attacker type is k.

— Set of possible chosen path $\mathcal{C} = \{\mathbf{c}_\tau \in \{0,1\}^K | \ \|\mathbf{c}_\tau\|_1 = 1\}$,
where $\mathbf{c}_\tau \in \{(1,0,\ldots,0),(0,1,\ldots,0),\ldots,(0,\ldots,1)\}$. $\mathbf{c}_\tau$ means chosen path (pure strategy of attacker type) at round τ.

— Define $c_j^i \in [-1,0]$, $u_j^i \in [0,1]$ as payoff of attacker i if target j is chosen to be attacked. c_j^i, u_j^i is payoff of attacker i under the condition that target i is covered, not covered by the defender.

— Expected utility function U: $[K] \times [L] \times \mathcal{P} \to \mathbb{R}$. $U(j,i,\mathbf{p}) = c_j^i p_i + u_j^i(1-p_i)$.

— Attacker best response function b: $[K] \times \mathcal{P} \to [L]$. We have mentioned in above that: attackers will attack the target with the best expected payoff. In the Stackelberg repeated security game, there will be an attacker j at round t, having observed the defender's strategy $\mathbf{p}$ and attack the target with the best U. The target to be attacked at round t could be represented as: $b(j,\mathbf{p}) = \arg\max_{i\in[L]} U(j,i,\mathbf{p})$. There may exist tie-breaking when we compute the best response of the attacker. That is, there may exist more than one target to attack that has the best-expected payoff, the attacker will choose the target that has the best defender expected payoff to attack.

◦ Defender:

— Set of targets $[L]$.

— Coverage probability vector $\mathbf{p} = (p_1, ..., p_L)$, p_i is the probability target i is covered by the defender, $p_i \geq 0$, $\sum_{i=1}^L p_i = 1$,$\|\mathbf{p}\|_1 = 1$

— Define strategy space $\mathcal{P}$ as the continuous space of all mixed strategies determined by $p_i \geq 0$, $\sum_{i=1}^L p_i = 1$, namely, $\mathcal{P} = \{\mathbf{p} \in R^L | \ p_i \geq 0, \ \sum_{i=1}^L p_i = 1\}$.

— Define c_d^i and u_d^i as defender's payoff values when target i (covered and not covered) is attacked. $c_d^i \in [-1,0]$, $u_d^i \in [0,1]$.

— Function id: $\{0,1\}^K \to \mathcal{K}$. $id(\mathbf{c}_\tau)$ means attacker type in game of round τ.

— Expected utility vector $\mathbf{f}$: $\mathcal{P} \to \mathbb{R}^K$. $\mathbf{f}(\mathbf{p}) = (f_1(\mathbf{p}), f_2(\mathbf{p}), \ldots, f_k(\mathbf{p}))$, where $f_j(\mathbf{p})$ is the expected utility of defender under the constraint that attacker type is j and $\mathbf{p}$ is given.

— Expected utility function f: $\mathbb{T} \times \mathcal{P} \to \mathbb{R}$. $f(\mathrm{id}(\mathbf{c}_\tau), \mathbf{p})$ is the expected utility of defender under the constraint that attacker type is $\mathrm{id}(\mathbf{c}_\tau)$ at round τ and $\mathbf{p}$ is given. Moreover $f(\mathrm{id}(\mathbf{c}_\tau), \mathbf{p}) = \langle \mathbf{c}_\tau, \mathbf{f}(\mathbf{p}) \rangle$.

— Defender expected utility in a game at round t: $\mathbf{f}_{\mathrm{id}(\mathbf{c}_t)}(\mathbf{p})$: $\mathcal{P} \to \mathbb{R}^K$, where $\mathbf{c}_t$ means path of attacker type at round t, $\mathrm{id}(\mathbf{c}_t)$ means attacker type at round t. In a Stackelberg security game at round t, once the defender's strategy vector $\mathbf{p}$ is given, the attacker type at round t could be determined by the sequence of attacker types. By numeric comparison of expected utility function U, the target to be attacked $b(j, \mathbf{p})$ could also be determined. Hence, the expected utility function of the defender and attacker's expected utility could be represented by t and $\mathbf{p}$. Defender expected utility function f: $f_t(\mathbf{p}) = c_d^{b(j(t),\mathbf{p})} p_{b(j(t),\mathbf{p})} + u_d^{b(j(t),\mathbf{p})}(1 - p_{b(j(t),\mathbf{p})})$, where $j(t)$ is the attacker type at round t, $c_d^{b(j(t),\mathbf{p})}$, $u_d^{b(j(t),\mathbf{p})}$ are defender's payoff values when target $b(j(t), \mathbf{p})$ (covered and not covered) is attacked.

— The expected utility of the defender could be determined when the target to be attacked by attackers is given. However, by using function $b(j, \mathbf{p})$, the expected utility could be determined when attacker type j and defender strategy $\mathbf{p}$ is given. The best response of attacker j could be determined by c_j^i, u_j^i and $\mathbf{p}$, then the defender expected utility could be computed by $b(j, \mathbf{p})$. $f(j, \mathbf{p}) = c_d^{b(j,\mathbf{p})} p_{b(j,\mathbf{p})} + u_d^{b(j,\mathbf{p})}(1 - p_{b(j,\mathbf{p})})$.

3 Problem formulation

In this section, we will discuss the problems that exist in repeated SSG and try to propose methods to solve them. Based on the problem formulation proposed by [1], we proposed an algorithm that applies the "Follow the Perturbed Leader" method to compute optimal strategy. Consider an attacker sequence from $[K]$, the attacker will attack targets according to the attacker sequence. In repeated SSG, the attacker's sequence is unknown to the defender, the defender can utilize information from previous rounds to formulate strategies for the current round. Therefore, at every time step t we can try to conduct online combinatorial optimization to compute a mixed strategy, denoted as $\mathbf{p}_t$. We can compute the expected payoff of defender for the current round based on $\mathbf{p}_t$, and the expected payoff for the past T rounds can be represented as:

$$\sum_{t=1}^{T} f(\mathrm{id}(\mathbf{c}_t), \mathbf{p}) = \sum_{t=1}^{T} \langle \mathbf{c}_t, \mathbf{f}(\mathbf{p}) \rangle.$$

To quantitatively evaluate the performance of $\mathbf{p}_t$, we consider another scenario: in each round, the defender not only knows the information from all previous rounds but also knows the information for the current round.

Based on this information, the defender can compute the optimal strategy and it is denoted by:

$$\mathbf{p}^* = \arg\max_{\mathbf{p}} \sum_{t=1}^{T} f(\mathrm{id}(\mathbf{c}_t), \mathbf{p}).$$

. We can apply the proposed optimal strategy $\mathbf{p}^*$ in each time step of repeated SSG, allowing us to compute the expected payoff for the past T rounds under $\mathbf{p}^*$. The difference between these two payoffs could help us to evaluate the performance of the algorithm. The difference is also denoted as regret, i.e.,

$$\sum_{t=1}^{T} f(\mathrm{id}(\mathbf{c}_t), \mathbf{p}^*) - \sum_{t=1}^{T} f(\mathrm{id}(\mathbf{c}_t), \mathbf{p^t}).$$

In this paper, we applied the "Follow the Perturbed Leader" algorithm to update $\mathbf{p}_t$ at each round and we get the regret bound:

$$\sum_{t=1}^{T} f(\mathrm{id}(\mathbf{c}_t), \mathbf{p}^*) - \sum_{t=1}^{T} f(\mathrm{id}(\mathbf{c}_t), \mathbf{p^t}) \leq 2\sqrt{2T}.$$

Since the average regret at each round will go to zero as T goes to infinity, our algorithm is a non-regret algorithm.

4 Optimal strategy in a Stackelberg security game

When it comes to high dimension SSG problem, time cost of solving MLP is pretty high (we have to consider n^k target-attacker corresponding relationships). By the DOBSS algorithm proposed in [4], the optimization problem could be written as a mixed integer linear programming problem. In the [4], the SSG optimization problem could be written as a Mixed-Integer Quadratic Program, it could be represented as follows:

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{q},a}{\text{minimize}} \quad & \sum_{l\in[K]} \sum_{i\in[L]} \sum_{j\in[L]} p^l R^l_{ij} x_i q^l_j \\
\text{subject to} \quad & \sum_{i\in[L]} x_i = 1 \\
& \sum_{j\in[L]} q^l_j = 1 \\
& 0 \leq (a^l - \sum_{i\in[L]} C^l_{ij} x_i) \leq (1 - q^l_j) M \\
& x_i \in [0,1] \\
& q^l_j \in \{0,1\} \\
& a \in \mathcal{R},
\end{aligned} \tag{1}$$

where $\mathbf{x}$ is the mixed strategy of defender, R_{ij}^l is the payoff of defender if target j is attacked by attacker l and defender protects target i. The probability of the attacker's type being l is $p^l = P_{\alpha_l}$. If $q_j^l = 1$, the target attacked by attacker j is i. However, to solve this problem by MIQP, the matrix must be positive definite. The matrix does not always satisfy such conditions. Through the change of variables $z_{ij}^l = x_i q_j^l$, we could transform MIQP problem into a MILP problem:

$$
\begin{aligned}
\underset{\mathbf{x},\mathbf{q},a}{\text{minimize}} \quad & \sum_{l\in[K]}\sum_{i\in[L]}\sum_{j\in[L]} p^l R_{ij}^l z_{ij}^l \\
\text{subject to} \quad & \sum_{i\in[L]}\sum_{j\in[L]} z_{ij}^l = 1 \\
& \sum_{j\in[L]} z_{ij}^l \le 1 \\
& q_j^l \le \sum_{i\in[L]} z_{ij}^l \le 1 \\
& \sum_{j\in[L]} q_j^l = 1 \\
& 0 \le (a^l - \sum_{i\in[L]} C_{ij}^l (\sum_{h\in[L]} z_{ih}^l)) \le (1 - q_j^l)M \\
& z_{ij}^l \in [0,1] \\
& q_j^l \in \{0,1\} \\
& a \in \mathcal{R}.
\end{aligned}
\tag{2}
$$

Utility matrix R and C could be represented as follows: $R_{ii} = c_d^i$, $R_{ij} = u_d^j$ $(i \neq j)$, $C_{ii}^l = c_l^i$, $C_{ij}^l = u_l^j$ $(i \neq j)$.

5 Follow the Perturbed Leader

In this section, we will give a detailed introduction of our algorithms according to follow the perturbed leader[2]. Firstly, we initialize $\mathbf{c}_0(t)$ at round t. Then we compute the best strategy $\mathbf{p}_t$ at time step t by solving mixed integer programming. After T rounds of games, optimal strategy $\mathbf{p}_T$ could be computed and the optimal strategy in hindsight $\mathbf{p}_t^*$ could also be computed by solving MILP. The cumulative regret at round T could be determined. Applying FTPL, our procedure is as follows:

1. At the beginning of each round, conduct initialization of c_0. At round t of game, we initialize $\mathbf{c}_0(t) =$

$(c_{0,1}(t), \ldots, c_{0,L}(t))$, where $c_{0,l}(t) \sim \mathcal{U}[0, 2/\epsilon(t)]$ and $\epsilon(t) = \sqrt{2/t}$.

2. Conduct online learning at each round. In each round t, we know randomly generated $\mathbf{c}_0(t)$, and $c_1, \ldots, c_{t-1}$ in last $t-1$ rounds. We will "predict" $\mathbf{p}_t$ in round t according to the known information by solving problem below:

$$\textbf{maximize} \sum_{i=1}^{t-1} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}) \rangle + \langle \mathbf{c}_0(t), \mathbf{f}(\mathbf{p}) \rangle.$$

We have $\mathbf{p}_t = \arg\max_{\mathbf{p} \in \mathbb{R}^L} (\sum_{i=1}^{t-1} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}) \rangle + \langle \mathbf{c}_0(t), \mathbf{f}(\mathbf{p}) \rangle)$.

3. Determine best strategy at round t. Having known the sequence of attacker types in last t rounds, the best strategy in round t could be determined. Define $\mathbf{p}_t^*$ as the best strategy subject last t rounds of games, $\mathbf{p}_t^* = \arg\max_{\mathbf{p} \in \mathbb{R}^L} \sum_{i=1}^{t} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}) \rangle$.

4. Compute regret at round t. As the best strategy and "predicted" best strategy are determined, the regret at round t could be written as follows: $\text{Regret}(t) = \sum_{i=1}^{t} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}_t^*) \rangle - \sum_{i=1}^{t} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}_i) \rangle$. $\mathbf{p}_t^*$and $\mathbf{p}_t$ can be computed by solving multiple linear programs or a single mixed-integer linear program.

Algorithm 1 Follow the Perturbed Leader

Input: T, R, C

1: $c_{0,i}(T) \leftarrow \mathcal{U}[0, 2/\epsilon(T)]$

2: $\mathbf{c}_0(T) \leftarrow (c_{0,1}(T), \ldots, c_{0,L}(T))$

3: $\mathbf{p}_t \leftarrow \arg\max \sum_{i=1}^{t-1} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}) \rangle + \langle \mathbf{c}_0(t), \mathbf{f}(\mathbf{p}) \rangle$ ▷ Mixed Integer Linear Program

4: $\text{Regret}(T) \leftarrow \sum_{i=1}^{T} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}_T^*) \rangle - \sum_{i=1}^{T} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}_i) \rangle$

Output: $\mathbf{p}_T$

6 Regret bound

An theorem from [3] could help us to derive the regret bound:

$\textbf{Hanna}(\delta)$[3]: On each period t,

1. Choose $\mathbf{c}_0(t)$ uniformly at random from the cube $[0, \sqrt{t}/\delta]^L$
2. Compute $\mathbf{p}_t^*$ and $\mathbf{p_t}$ by solving MILP.

Denote $\mathrm{Regret}(t, \delta)$ as regret of $\mathbf{Hanna}(\delta)$ at time round t.

Lemma 1. $\mathrm{Regret}(t, \delta) \leq 2\delta RA\sqrt{T} + \frac{D\sqrt{T}}{\delta}$, *where* $D \geq |\mathbf{p} - \mathbf{p}'|, R \geq f(\alpha, \mathbf{p})$, $A \geq |\mathbb{I}_{c_t=l}|$.

In the Stackelberg security game we defined, $D \geq 1, R \geq 1, A \geq 1$

Theorem 1. *Regret at round* T *is less than* $2\sqrt{2T}$, *denoted as:*

$$Regret(T) \leq 2\sqrt{2T}$$

7 Numerical Experiments

Solving Mixed Integer Linear Program is a more effective way to compute $\mathbf{p}_t^*$. In this section, we shall refer to Follow the Perturbed Leader algorithm 0 and perform a verification of sub-linear property in Theorem 1. At the end of this section, we will give a comparison between Follow the Perturbed Leader and Follow the Regularized leader.

7.1 Sublinear Property

In this section, we will give a verification of sub-linear property of regret bound. Under sub-linear property, the average regret goes to zero as T goes to infinity. We will "predict" optimal defender strategy in each time step and determine the offline best strategy at each round. Then, we will compute the regret of each strategy. The cumulative regret difference of p_t^* and p_t will be determined by $\mathrm{Regret}(t) = \sum_{i=1}^{t} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}_t^*) \rangle - \sum_{i=1}^{t} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}_i) \rangle$. We will record regret difference at each time step $\mathbf{t}$ and compare them with theoretic regret bound.

In this experiment, we choose randomized utility matrix of defender and attackers as in (2). The utility matrices of defenders and payoff matrices of attackers are as follows:

Table 1: Utility Matrix of defender

Protected Target	Attack 1	Attack 2	Attack 3	Attack 4	Attack 5	Attack 6
Target 1	0.32	-0.84	-0.4	-0.13	-0.36	-0.44
Target 2	-0.87	0.56	-0.4	-0.13	-0.36	-0.44
Target 3	-0.87	-0.84	0.01	-0.13	-0.36	-0.44
Target 4	-0.87	-0.84	-0.4	0.11	-0.36	-0.44
Target 5	-0.87	-0.84	-0.4	-0.13	0	-0.44
Target 6	-0.87	-0.84	-0.4	-0.13	-0.36	0.22

Table 2: Payoff Matrix of attacker (target is not covered by defender)

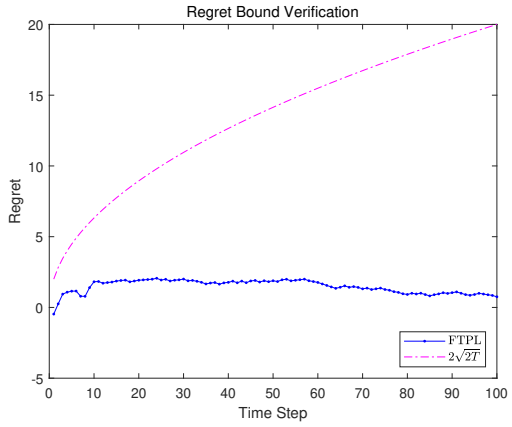
Attacked Target	Attacker 1	Attacker 2	Attacker 3	Attacker 4	Attacker 5
Target 1	0.94	0.58	0.48	0.22	0.91
Target 2	0.99	0.83	0.56	0.20	0.35
Target 3	0.01	0.54	0.98	0.45	0.17
Target 4	0.94	0.38	0.85	0.55	0.78
Target 5	0.60	0.24	0.43	0.31	0.61
Target 6	0.78	0.81	0.26	0.48	0.51

Table 3: Payoff Matrix of attacker (target is covered by defender)

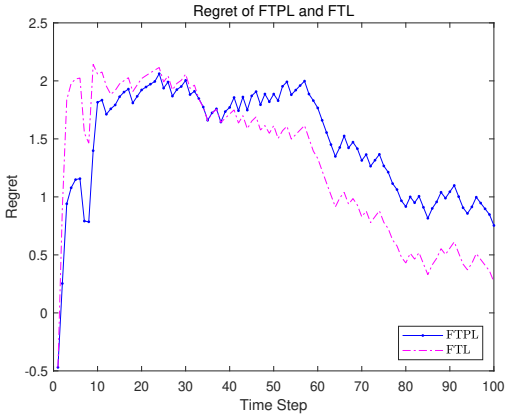
Attacked Target	Attacker 1	Attacker 2	Attacker 3	Attacker 4	Attacker 5
Target 1	-0.1	0.44	-0.43	0.05	-0.25
Target 2	-0.68	0.55	0.21	-0.10	-0.17
Target 3	0	0.44	-0.68	0.19	-0.10
Target 4	-0.92	-0.06	0.43	0.45	0.11
Target 5	-0.60	0.14	0.35	0.29	0.42
Target 6	0.69	-0.76	-0.05	-0.39	0.7

In the experiment, we use last $t-1$ rounds' information (attacker sequence) and randomized $\mathbf{c}_0(t)$ to "predict" optimal strategy at round t. Then we use first t rounds' information to determine the best offline

strategy. At each time step t, the expected utility under FTPL strategy p_t and best offline strategy p_t^* could be determined. The difference of the accumulated expected utility is $\mathrm{Regret}(t)$. According to $\mathrm{Regret}(t)$, we could give a verification of sub-linear property in Theorem 1. The graph above shows us that the regret



(a)



(b)

of FTPL at round t increases slower than theoretical upper bound. From sub-linear property, our algorithms are no-regret algorithms.

7.2 FTPL FTL

In the last numerical experiment, we will compare "Follow the Perturbed Leader" with another online learning method, "Follow the leader"[5]. Its algorithm is as follows:

Algorithm 2 Follow the Leader

Input: T, R, C

1: $\mathbf{p}_T \leftarrow \arg\max \sum_{i=1}^{t-1} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}) \rangle$ ▷ Mixed Integer Linear Program

2: $\mathrm{Regret}(T) \leftarrow \sum_{i=1}^{T} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}_T^*) \rangle - \sum_{i=1}^{T} \langle \mathbf{c}_i, \mathbf{f}(\mathbf{p}_i) \rangle$

Output: $\mathbf{p}_T$

From the graph above, when time step t is small, FTPL has better performance. With time going by, FTL has better performance in predicting optimal strategy.

7.3 Extreme Case

In this section, we will introduce a special case of FTL. FTL has bad performance on it. Based on this case, we will develop its idea and propose an extreme case of stackelberg security game where FTL performs bad. We will discuss an example from the monograph "Online Learning and Online Convex Optimization" by Shai Shalev-Shwartz to support our point. The following example from [6] shows that FTL may have poor performance:

Example 1. (Failure of FTL). Let $S = [-1, 1] \subset \mathbb{R}$ and consider the sequence of linear functions such that $f_t(w) = z_t w$ where

$$z_t = \begin{cases} -0.5 & \text{if } t = 1 \\ 1 & \text{if } t \text{ is even} \\ -1 & \text{if } t > 1 \wedge t \text{ is odd} \end{cases}$$

Then, the predictions of FTL will be to set $w_t = 1$ for t odd and $w_t = -1$ for t even. The cumulative loss of the FTL algorithm will therefore be T while the cumulative loss of the fixed solution $u = 0 \in S$ is 0 . Thus, the regret of FTL is T !

In the example above, when t is even, the best strategy is $w = 1$, and FTL predicts the best strategy is $w = -1$, when t is odd, the best strategy is $w = -1$ and FTL predicts the best strategy is $w = 1$. FTL always returns a bad strategy. The example provides an idea to create a sequence of attacker types where FTL performs bad. We could choose a sequence that could deceive defender under FTL, that is, the best strategy in the last $t - 1$ rounds becomes a bad strategy at round t. Then we could get a linear regret of FTL. From the example above, we design a repeated Stackelberg security game to give a verification of FTL's bad performance in some extreme cases. The payoff matrix is as follows:

Table 4: Utility Matrix of defender

Protected Target	Attack 1	Attack 2
Target 1	0	-1
Target 2	-0.9	0

Table 5: Payoff Matrix of attacker (target is not covered by defender)

Attacked Target	Attacker 1	Attacker 2
Target 1	0	1
Target 2	1	0

Table 6: Payoff Matrix of attacker (target is covered by defender)

Attacked Target	Attacker 1	Attacker 2
Target 1	0	0.5
Target 2	0.5	0

Based on the payoff matrix, we could choose a sequence of attacker type. We choose a repeated attacker type sequence of $1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1$. At each round t, FTL will provide a bad strategy to defender and when t is divisible by 19, the strategy will be not such bad. The regret of FTPL and FTL is as follows: FTL has a bad performance in the example above. Almost every time step, defender under FTL

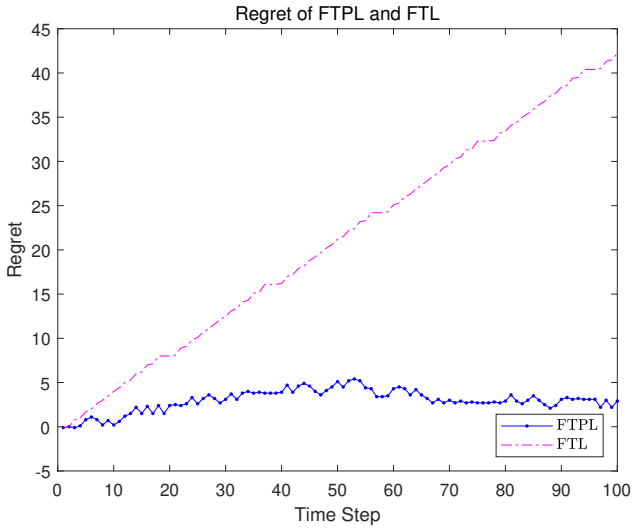


Figure 2

will be deceived by the last $t-1$ attackers and apply a bad strategy at round t. The cumulative regret of FTL is linear, and the cumulative regret of FTPL is sublinear.

References

[1] Maria-Florina Balcan, Avrim Blum, Nika Haghtalab, and Ariel D Procaccia. Commitment without regrets: Online learning in stackelberg security games. In *Proceedings of the sixteenth ACM conference on economics and computation*, pages 61–78, 2015.

[2] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[3] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[4] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 895–902, 2008.

[5] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[6] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.