1 Computational Game Theory for Security

Yevgeniy Vorobeychik, Washington University in Saint Louis

1.1 Introduction

Security, or protection from harm, is a central social good, crucial for thriving organizations and communities. In security, harm comes not spontaneously, but as a result of deliberate vicious acts by a malicious party or parties. For example, home security protects residents against unwanted intrusions, security in an airport entails protection against threats of violence by organized terror groups, while cybersecurity in an organization means protection against sabotage by parties who may aim to reduce the organization's competitiveness or harm its reputation. The interaction between the *defender*, who strives to protect a collection of assets, and an *attacker*, who aims to achieve some malicious goal that likely leads to direct or indirect harm to these assets, is thus endemic in security encounters. Game theory, which is a mathematical framework for modeling interactions among strategic (self-interested) agents whose actions impact one another, has emerged as an important framework for modeling such defender-attacker interactions in security.

An important conceptual step in game-theoretic modeling for security is to frame the interaction between the defender and attacker as sequential, with the defender first deploying ("committing" to) its defense strategy, and the attacker then responding to deployed defense in deciding on the attack strategy. A formal representation for this structure of interaction is as a Stackelberg game, in which a leader (in our case, the defender) chooses a strategy first, and the follower (the attacker) chooses thereafter, upon observing the leader's strategy. While alternative modeling frameworks abound, the Stackelberg model has had considerable practical impact [Abbas et al. 2017, Fang et al. 2015, Tambe 2011], and will therefore be the primary focus of this chapter.

We begin this chapter by describing a formal model first of Stackelberg games (Section 1.2), and then their particular variations common in security settings, *Stackelberg security games* (Section 1.3), along with solution techniques for these. Subsequently, we consider a particular class of security problems in which the defender aims to *interdict* a path taken by an attacker through a network (Section 1.4). We broaden this first to consider attackers as decision makers in a dynamic setting, and, finally, study security interactions in dynamic

environments in which the defender commits to a *policy*, formalized as *stochastic Stackelberg games*, with adversarial patrolling problem providing a concrete application of this modeling framework (Section 1.5).

1.2 Stackelberg Games

Let us start with Stackelberg games, generally. Consider two players, a leader (L) and a follower (F). Both L and F have sets of actions, A_L and A_F, respectively, along with their respective utility functions $u_L(a_L, a_F)$ and $u_F(a_L, a_F)$. Throughout, we assume that $A_L \cup A_F$ is finite. Let σ_L be a probability distribution over A_L, with $\sigma_L(a) \geq 0$ the associated probability of $a \in A_L$. Thus, $\sum_{a \in A_L} \sigma_L(a) = 1$. It is common to refer to σ_L as the leader's *mixed strategy*. We can extend utility functions to be functions of σ_L by taking the expectations with respect to this distribution, i.e.,

$$u_L(\sigma_L, a_F) = \sum_{a \in A_L} \sigma_L(a) u_L(a, a_F) \quad \text{and} \quad u_F(\sigma_L, a_F) = \sum_{a \in A_L} \sigma_L(a) u_F(a, a_F).$$

The follower is presumed to observe σ_L, but, crucially, *not the actual realization of random draws a from it*, and then respond by choosing a strategy $b \in A_F$. In particular, we assume that the follower chooses a strategy that is a *best response* $\phi(\sigma_L)$ to σ_L, that is,

$$\phi(\sigma_L) \in \arg\max_{b \in A_F} u_F(\sigma_L, b). \tag{1.1}$$

Our notation here already makes explicit that a best response strategy of the follower explicitly depends on the observed leader strategy σ_L; indeed, this dependence is what makes this game Stackelberg in nature, and the full strategy space of the follower in this game is the space of all functions that map σ_L to follower actions b.

At this point, one may observe that our formalization of the best response in Equation (1.1), we have effectively assumed that the follower need not consider randomized strategies. Indeed, it is not difficult to show that in the game model we consider here, there always exists a deterministic best response. Intuitively, if a randomized strategy puts non-zero weight on multiple follower actions, we can do no worse than shifting all of this weight to the action which yields the highest follower utility.

A common solution concept for Stackelberg games is a *Stackelberg equilibrium*, in which the follower chooses a best response strategy $\phi(\sigma_L)$, while the leader chooses σ_L to maximize its strategy *given the follower's best response* ϕ. Formally, (σ_L^*, ϕ) is a Stackelberg equilibrium if ϕ is the follower's best response in the sense of Equation (1.1), and $\sigma_L^* \in \arg\max_{\sigma \in \Delta_L} u_L(\sigma, \phi(\sigma))$, where Δ_L is the simplex (i.e., the set of all probability distributions) over A_L.

A limitation of a Stackelberg equilibrium is that the follower may have many ways to break ties, and the particular choices could be somewhat arbitrary. It is common, therefore, to make use of a refinement, referred to as the *Strong Stackelberg equilibrium (SSE)*, in which

the follower breaks ties in the leader's favor (commonly justified using the argument that the leader can induce this to be a strict best response for the follower by simply slightly changing the strategy σ_L). Specifically, (σ_L^*, ϕ) form an SSE if

$$\sigma_L^* \in \underset{\sigma \in \Delta_L; \phi(\sigma) \in \arg\max_{b \in A_F} u_F(\sigma, b)}{\arg\max} u_L(\sigma, \phi(\sigma)). \tag{1.2}$$

It turns out that computing SSE amounts to solving a collection of linear programs. The idea is this. Let us fix a follower strategy $b \in A_F$, and ask the following question: how can the leader maximize utility while at the same time incentivizing the follower to choose b (i.e., while b is constrained to be the follower's best response)? The solution to this problem can be formulated as the following linear program:

$$\max_{\sigma \geq 0} \sum_{a \in A_L} \sigma(a) u_L(a, b) \tag{1.3a}$$

$$\text{subject to :} \tag{1.3b}$$

$$\sum_{a \in A_L} \sigma(a) = 1 \tag{1.3c}$$

$$\sum_{a \in A_L} \sigma(a) u_L(a, b) \geq \sum_{a \in A_L} \sigma(a) u_L(a, b') \quad \forall b' \in A_F \setminus b. \tag{1.3d}$$

Of course, there may be no solution to the linear program (LP) (1.3) for particular choices of b, but that simply means that it is not possible for the leader to have b be the follower's best response (e.g., if b is dominated by other choices the follower has). However, suppose we solve it for every possible strategy $b \in A_F$, with the associated solution σ^b and leader utility $u_L(\sigma^b, b)$. Whenever LP (1.3) is infeasible, define σ^b arbitrarily and $u_L(\sigma^b, b) = -\infty$. Then $\sigma^* = \sigma^{b^*}$, where $b^* \in \arg\max_b u_L(\sigma^b, b)$. In other words, we simply choose the solution to the LP corresponding to the opponent strategy b for which the leader's solution yields the highest leader utility.

The Stackelberg game model above presumes that the leader has complete information about the follower, and vice-versa. An important generalization is to consider a leader's uncertainty about the follower's utility function, u_F, which we refer to as a *Bayesian Stackelberg game*. Specifically, suppose that there is a finite set of follower *types*, Θ. Each type $\theta \in \Theta$ is a parameter of the follower's utility function, which now takes the form $u_F(a_L, a_F; \theta)$ (with the extension to $u_F(\sigma_L, a_F; \theta)$ exactly as above). Let p_θ be the probability that the follower is of type θ; we suppose that although the leader does not know the actual type, the distribution itself is common knowledge. Let $\phi(\sigma; \theta)$ be the follower's *best response* strategy, which is now parametrized by θ, and is characterized by $\phi(\sigma; \theta) \in \arg\max_{b \in A_F} u_F(\sigma, b; \theta)$. The leader then maximizes its expected utility, $\sum_{\theta \in \Theta} p_\theta u_L(\sigma, \phi(\sigma, \theta))$, over randomized strategies σ. Now, we can again seek a SSE solution to this problem. However, the approach above where we solve multiple linear programs, one for each possible follower action, no longer works: since each

follower type can take any of the actions, we must solve $|\Theta|^{|A_F|}$ linear programs in the worst case!

Instead, we can compute an SSE of a Bayesian Stackelberg game using the following mixed-integer linear program (MILP):

$$\max_{\sigma \geq 0} \sum_{\theta \in \Theta} p_\theta u_\theta \tag{1.4a}$$

$$\text{subject to :} \tag{1.4b}$$

$$\sum_{a \in A_L} \sigma(a) = 1 \tag{1.4c}$$

$$\sum_{b \in A_F} x(b,\theta) = 1 \quad \forall \theta \in \Theta \tag{1.4d}$$

$$u_\theta - \sum_{a \in A_L} \sigma(a) u_L(a,b) \leq (1 - x(b,\theta))Z \quad \forall b \in A_F, \theta \in \Theta \tag{1.4e}$$

$$0 \leq v_\theta - \sum_{a \in A_L} \sigma(a) u_F(a,b) \leq (1 - x(b,\theta))Z \quad \forall b \in A_F, \theta \in \Theta. \tag{1.4f}$$

In this MILP, u_θ denotes the expected utility of the leader when the follower's type is θ, while v_θ denotes the corresponding expected utility for the follower. Since the leader does not actually know the follower's type, the objective also takes the expectation with respect to the probability distribution over the follower types. Turning to constraints, their main role is to compute the best response strategy $\phi(\sigma;\theta)$ of the follower. We encode it using a binary vector $x(b,\theta)$ such that $x(b,\theta) = 1$ whenever b is the follower's *actual* best response. Note that it depends on σ implicitly insofar as the strategy σ of the leader is itself being computed by the MILP. Constraints (1.4f) actually do the work of computing the follower's best response for each type θ, while Constraints (1.4e) use the best response strategies x of the follower to compute the leader's associated expected utility. In both of the sets of constraints above, Z is a large constant that we take conceptually to be $Z = \infty$.

1.3 Stackelberg Security Games

The Stackelberg game model can be applied to security settings directly as follows. Let the leader be the defender, with A_L the set of defender strategies, and the follower the attacker, with A_F the set of possible attacks. The machinery we developed goes through directly. As it turns out, however, there is more to say here.

Let us make the sets of strategies for the defender and attacker more concrete. In particular, let T be the set of targets that need to be defended from a possible attack. Suppose that the defender can defend—or *cover*—at most K targets at a time; that is, the defender has K protection resources (e.g., security guards). Then a defender strategy is a subset $C \subset T$ such that $|C| \leq K$, and the strategy set A_L is the set of all possible such subsets of targets. Similarly, if the attacker has L resources to attack the targets in T, the set of attack strategies A_F is the

set of all subsets of T of size at most L. Even if we suppose that both players saturate their resource use, both A_L and A_F grow exponentially in the number of targets. Consequently, the approaches developed above, which require us to enumerate the entire A_L and A_F will not scale. This means that we must exploit problem structure to develop effective approaches to practical security game models.

Let us add some structure to the utility functions. Let U_t^c be the utility the defender receives when a target t is attacked while it is covered by the defender, and V_t^c the associated utility of the attacker. Similarly, define U_t^u and V_t^u to be the utilities of the defender and attacker, respectively, when target t is attacked while it is not covered by the defender. Next, suppose that the attacker can only attack a single target; that is, $L = 1$, and the attacker strategy set is just the set of targets T. Further, suppose that instead of a hard constraint K on the number of defense resources, the defender incurs a cost c for each time it uses a resource to cover a target. Let us also represent the defender's choice as a coverage vector $d \in \{0,1\}^n$, where $n = |T|$ is the number of targets, and let σ be the probability distribution over d (i.e., the defender's mixed strategy). The defender's expected utility is then

$$u_D(\sigma,t) = \sum_d \sigma(d) \left((d_t U_t^c + (1-d_t)U_t^u) - c\sum_t d_t \right).$$

If we define $q_t \in [0,1]$ as the probability of covering target t induced by the mixed strategy σ, we can rewrite the utility as

$$u_D(q,t) = q_t U_t^c + (1-q_t)U_t^u - c\sum_t q_t.$$

Consequently, we no longer need to explicitly reason about the mixed strategy over all possible coverage vectors, but just over target-specific coverage probabilities q_t, aggregated into a vector q. This means that despite the combinatorial underlying strategy space of the defender, we can actually compute a SSE of the Stackelberg security game of this form in polynomial time using multiple LPs just as above, where each LP corresponding to a particular target t of an attack has the following form:

$$\max_q \quad q_t U_t^c + (1-q_t)U_t^u - c\sum_t q_t \tag{1.5a}$$

$$\text{subject to :} \tag{1.5b}$$

$$0 \le q_t \le 1 \quad \forall\, t \in T \tag{1.5c}$$

$$q_t V_t^c + (1-q_t)V_t^u \ge q_{t'} V_{t'}^c + (1-q_{t'})V_t^u \quad \forall t' \in T \setminus t. \tag{1.5d}$$

The key idea above is that we replace a hard constraint on the defender's resource use with a soft cost function that in principle allows unlimited resources. This approach can be adapted in a straightforward way to consider a constraint that the number of resources used *in expectation* is bounded by K instead of the soft cost function. What does not work,

however, is a hard constraint that *in every realization* of coverage vectors we can use at most K resources. In fact, imposing such a constraint effectively prevents us from using this compact representation of the problem, and we now must again enumerate all coverage vectors d.

But what if we do wish to impose a hard constraint that we can use at most K resources? It is often the case that there is little flexibility on this account: for example, we cannot hire new security staff if it happens that our utilization exceeds the current staff. The problem of inducing the attacker to choose a particular target t to attack then becomes

$$\max_{\sigma \geq 0} \quad \sum_d \sigma(d) u_D(d,t) \tag{1.6a}$$

$$\text{subject to :} \tag{1.6b}$$

$$\sum_d \sigma(d) = 1 \tag{1.6c}$$

$$\sum_d \sigma(d) \Delta V(d,t,t') \geq 0 \quad \forall\, t' \in T, \tag{1.6d}$$

where $\Delta V(d,t,t') = d_t V_t^c + (1-d_t)V_t^u - d_{t'} V_{t'}^c - (1-d_{t'})V_{t'}^u$ and the consideration set of vectors d is restricted to those that involve the use of K resources (since we always wish to saturate resource use in this context, there is no need to include those using fewer than K resources). This set is exponential in the number of targets $|T|$. Nevertheless, we can make use of the problem structure here to avoid explicitly enumerating all variables $\sigma(d)$. Specifically, since the number of constraints is linear in the number of targets, we actually know that the support of the solution σ (i.e., the set of configurations d with $\sigma(d) > 0$) is also going to be linear in $|T|$ [Bertsimas and Tsitsiklis 1997]; the trick is just to find it. To do this, we can use a technique called *column generation*, in which we begin with an arbitrary small subset of l resource assignments $\hat{D} = \{d_1, \ldots, d_l\}$, and solve the linear program (1.6) considering only these. Suppose $\hat{\sigma}(\hat{D})$ is the resulting optimal solution. We can use this solution to compute the *reduced cost* of the standard form analog of this linear program [Bertsimas and Tsitsiklis 1997]. We now construct this standard form analog and derive the expression for reduced cost. First, we add slack variables $\lambda_{t'}$ to each of the inequality constraints to turn these into equality constraints, obtaining

$$\sum_d \sigma(d) \Delta V(d,t,t') - \lambda_{t'} = 0 \quad \forall\, t' \in T.$$

Let $u(t)$ be a vector associated with the security resource configurations in $\hat{D}$, and let $\Delta V(t,t')$ be a vector of $\Delta V(d,t,t')$ for all $d \in \hat{D}$. Turning each constraint into matrix-vector notation, we obtain

$$1^T \sigma = 1$$

for the first constraint, where 1^T is a row vector of 1s, and

$$\Delta V(t,t')^T \sigma - \lambda_{t'} = 0$$

for the rest. Let λ be a vector of all slack variables $\lambda_{t'}$. Define a matrix A as follows. Let the first row be $a_1 = [1^T, 0]$, where the number of 1s is the number of variables, and let each subsequent row be $a_{t'} = [\Delta V(t,t'), -1]$. Finally, define $b = [1, 0^T]^T$, where 0^T is a vector of $|T|$ 0s. The problem can then be rewritten as

$$\min_{\sigma,\lambda \geq 0} \quad -u(t)^T \sigma \tag{1.7a}$$

$$\text{subject to :} \tag{1.7b}$$

$$A[\sigma,\lambda]^T = b. \tag{1.7c}$$

From this standard form, we can then derive the expression for reduced cost associated with an arbitrary resource assignment d' that is not in $\hat{D}$ as follows. First, an optimal solution to the linear program (1.7) over variables in $\hat{D}$ will compute the set of variables d for which $\sigma(d) > 0$. Let B be the submatrix of A with the associated columns, and let $u_B(t)$ be the associated components of $u(t)$. Let $[\alpha,\beta] = u_B(t)^T B$, where α is a scalar and β a vector with one element for each target t'.

$$\bar{c}_{d'} = -d'_t U^c_t - (1-d'_t)U^u_t - \alpha - \sum_{t'} \beta_{t'}(d'_t V^c_t + (1-d'_t)V^u_t - d'_{t'}V^c_{t'} - (1-d'_{t'})V^u_{t'}).$$

Let $\bar{\beta} = \sum_{t'} \beta_{t'}$. Then this expression can be slightly simplified to

$$\bar{c}_{d'} = -d'_t(U^c_t + \bar{\beta}V^c_t) - (1-d'_t)(U^u_t + \bar{\beta}V^u_t) - \alpha + \sum_{t'} \beta_{t'}(d'_{t'}V^c_{t'} + (1-d'_{t'})V^u_{t'}).$$

Our goal is now to minimize the reduced cost, which is equivalent to solving the following integer linear program (ILP):

$$\min_{d'} \quad d'_t(U^u_t - U^c_t + \bar{\beta}(V^u_t - V^c_t)) - \sum_{t'} d'_{t'}\beta_{t'}(V^u_{t'} - V^c_{t'}) \tag{1.8}$$

$$\text{subject to :} \tag{1.9}$$

$$\sum_t d'_t = K \tag{1.10}$$

$$d'_t \in \{0,1\} \quad \forall t \in T. \tag{1.11}$$

Moreover, if for an optimal solution d' of this optimization problem we find that $\bar{c}_{d'} \geq 0$, we know that the solution we obtained in the last iteration is in fact optimal, and we can stop the column generation process. The power of this approach is that while ILPs are NP-hard to solve in general, in practice we can often solve them effectively and quickly using state-of-the-art ILP solvers.

1.4 Security Games on Networks

So far, we have studied both the rather general Stackelberg game model, and its specialization to security games with independent targets. One important class of problems in security entails

preventing, or *interdicting*, the flow of malicious activity on network. In its classical form, this problem is known as *network interdiction* [Cormican et al. 1998, Smith and Song 2020]. Many variations of this problem have been explored; here, I describe one of the first, *deterministic network flow interdiction* [Wood 1993], which well illustrates the main idea.

Consider a network $G = (V, E)$ where V is the set of nodes and E the set of (directed) edges, with a particular edge denoted by $(i, j) \in E$. Additionally, we associate each edge (i, j) with a capacity u_{ij}. Suppose that a malicious actor aims to solve the *network flow* problem over this network, maximizing the total amount of flow (of goods, malicious activity, etc) between a source node $s \in V$ and a target node $t \in V$. Assume for the moment that both s and t are known. The problem of maximizing total flow from s to t can be represented as the following linear program [Bertsimas and Tsitsiklis 1997, Wood 1993]:

$$\max_{x} \quad x_{ts} \tag{1.12a}$$

$$\text{subject to :} \tag{1.12b}$$

$$\sum_j x_{sj} - \sum_j x_{js} - x_{ts} = 0 \tag{1.12c}$$

$$\sum_j x_{ij} - \sum_j x_{ji} = 0 \quad \forall i \in V \setminus \{s,t\} \tag{1.12d}$$

$$\sum_j x_{tj} - \sum_j x_{jt} + x_{ts} = 0 \tag{1.12e}$$

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in E \tag{1.12f}$$

$$x_{ts} \geq 0, \tag{1.12g}$$

where x_{ij} represent flows along edges $(i, j) \in E$ and x_{ts} is an artificial edge added to the network, going from t to s.

The network interdiction problem involves removing a subset of edges S from E, subject to a budget constraint, so as to minimize the maximum network flow as computed by the above linear program with edges restricted to $E \setminus S$. Suppose that removing an edge (i, j) incurs a cost c_{ij}, and let C be the total budget on cutting edges. We can encode this network interdiction problem as follows:

$$\min_{y} \max_{x} \quad x_{ts} \tag{1.13a}$$

$$\text{subject to :} \tag{1.13b}$$

$$\sum_j x_{sj} - \sum_j x_{js} - x_{ts} = 0 \tag{1.13c}$$

$$\sum_j x_{ij} - \sum_j x_{ji} = 0 \quad \forall i \in V \setminus \{s,t\} \tag{1.13d}$$

$$\sum_j x_{tj} - \sum_j x_{jt} + x_{ts} = 0 \tag{1.13e}$$

$$0 \leq x_{ij} \leq u_{ij}(1-y_{ij}) \quad \forall (i,j) \in E \tag{1.13f}$$

$$\sum_{ij} c_{ij} y_{ij} \leq C \tag{1.13g}$$

$$x_{ts} \geq 0 \tag{1.13h}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in E, \tag{1.13i}$$

where y_{ij} represents the choice of edges that are cut. Finally, we can note that the inner maximum flow problem admits a dual linear program, and after taking the dual and linearizing the objective, we obtain the following integer linear program for network interdiction:

$$\min_{y,\alpha,\beta} \quad \sum_{ij} u_{ij} \beta_{ij} \tag{1.14a}$$

$$\text{subject to :} \tag{1.14b}$$

$$\alpha_i - \alpha_j + \beta_{ij} + y_{ij} \quad \forall (i,j) \in E \tag{1.14c}$$

$$\alpha_t - \alpha_s \geq 1 \tag{1.14d}$$

$$\sum_{ij} c_{ij} y_{ij} \leq C \tag{1.14e}$$

$$y_{ij}, \alpha_i, \beta_{ij} \in \{0,1\} \quad \forall i \in V, (i,j) \in E. \tag{1.14f}$$

The problem of network interdiction has received a great deal of attention over the years, with many variations of the setting above explored, including problems which are not zero-sum (that is, when the interdictor's goals are not necessarily in opposition to the adversary) [Jain et al. 2011], uncertainty (e.g., about the s,t source and destination nodes) [Cormican et al. 1998, Pay et al. 2019], etc. In the end, the general problem structure tends to closely resemble the basic network flow interdiction problem above.

One substantive variation of network interdiction is *plan interdiction*, in which the goal of the defender is to prevent the attacker from achieving its goals, which the latter attempts to accomplish by solving a dynamic decision problem. A number of variations of this problem have been explored [Letchford and Vorobeychik 2013, Panda and Vorobeychik 2017, 2018, Vorobeychik and Pritchard 2020]; here we describe a very general setting—Markov decision process (MDP) initial state interdiction problem (MDPSI) which admits a relatively simple solution approach.

Formally, the MDPSI is defined by 1) an MDP $\mathcal{M} = \{X, A, r(x,a), \gamma\}$, where $X, A, r(x,a)$, and γ are the state space, action space, reward function, and the discount factor of an infinite-horizon discounted MDP which the attacker is solving, respectively, and 2) interdiction costs $\rho(x'_0, x_0) = \sum_i \rho_i |x'_{i0} - x_{i0}|$ of modifying an initial state x_0 into x'_0, where ρ_i is the cost of modifying variable i. Note that this cost can also capture inability to modify specific state variables (the corresponding $\rho_i = \infty$). We assume that the game is zero-sum (modulo interdiction costs), so that the defender aims to minimize the sum of discounted attacker

rewards. Define $V(x_0, \pi)$ as the attacker's value function for a policy π starting at state x_0 in MDP $\mathcal{M}$. Let Π be the set of deterministic stationary policies of the MDP. The defender then solves

$$\min_{x_0' \in X} V(x_0') + \rho(x_0', x_0), \tag{1.15}$$

where $V(x)$ is the optimal value function of the MDP. The key observation is that the optimal policy is *independent* of the interdiction decision x_0', since the optimal policy of an MDP is independent of starting state. Consequently, we can decompose this problem into two parts: first, we compute or approximate the value function $V(x)$, and second, fix $V(x)$ and solve the optimization problem (1.15). The former can be done using standard reinforcement learning techniques, while the latter can subsequently be done using, for example, gradient-based methods (if the value function is differentiable in input state x).

1.5 Stochastic Stackelberg Games and Adversarial Patrolling

Stackelberg game models we had discussed thus far assume that attacker knows the probability that each target is covered by the defender, but is oblivious to the actual sequence of defender moves. For example, the defender may in fact visit targets according to some fixed (but randomly generated) patrolling schedule, but the attacker is presumed to be unable to observe the defender's location at any point during the patrol. In many realistic settings, such as the US Coast Guard [An et al. 2011], it is likely that the attacker can in fact observe the patrol while it is in progress (e.g., the coast guard ships can be quite overt). Thus, a more plausible model in such a setting would allow the attacker to observe both the randomized policy of the defender (i.e., probability distribution over moves) as well as current defender location. We can model this setting as an *adversarial patrolling game*, or APG. APGs, in turn, are a special case of *discounted stochastic Stackelberg games (DSSGs)*. We begin by discussing DSSGs generally, and then describe the APG model.

1.5.1 Stochastic Discounted Stackelberg Games

Two-player infinite-horizon DSSGs involve one player who is a "leader" and the other who is a "follower", just as in general Stackelberg games. The leader commits to a *policy* that becomes known to the follower who plays a best-response policy. These games have a finite state space S, finite action spaces A_L for the leader and A_F for the follower, payoff functions $R_L(s, a_l, a_f)$ and $R_F(s, a_l, a_f)$ for leader and follower respectively, and a transition function $T_{ss'}^{a_l a_f}$, where $s, s' \in S$, $a_l \in A_L$ and $a_f \in A_F$. The discount factors are $\gamma_L, \gamma_F < 1$ for the leader and follower, respectively. Finally, $\beta(s)$ is the probability that the initial state is s.

The history of play at time t is $h_t = \{s_1 a_{l,1} a_{f,1} \cdots s_{t-1} a_{l,t-1} a_{f,t-1} s_t\}$ where t denotes time. Let Π and Φ be the set of nonstationary and non-Markov policies (i.e., mappings from histories to distributions over actions) for the leader and follower, respectively. Let U_L and U_F

denote the utility functions for leader and follower respectively. For arbitrary policies $\pi \in \Pi$ and $\phi \in \Phi$,

$$U_L(s,\pi,\phi) = \mathbb{E}\Big[\sum_{t=1}^{\infty} \gamma_L^{t-1} R_L(s_t,\pi(h_t),\phi(h_t))|s_1 = s\Big],$$

where the expectation is over the stochastic evolution of the states, and where (abusing notation)

$$R_L(s_t,\pi(h_t),\phi(h_t)) = \sum_{a_l \in A_L} \sum_{a_f \in A_F} \pi(a_l|h_t)\phi(a_f|h_t) R_L(s_t,a_l,a_f),$$

with $\pi(a_l|h_t)$ the probability of leader-action a_l in history h_t under policy π, and $\phi(a_f|h_t)$ the probability of follower-action a_f in history h_t under policy ϕ. The utility of the follower, $U_F(s,\pi,\phi)$, is defined analogously.

For any leader policy $\pi \in \Pi$, the follower plays the best-response policy defined as

$$\phi_\pi^{BR} \in \arg\max_{\phi \in \Phi} \sum_s \beta(s) U_F(s,\pi,\phi).$$

The leader's optimal policy is then

$$\pi^* \in \arg\max_{\pi \in \Pi} \sum_s \beta(s) U_L(s,\pi,\phi_\pi^{BR})$$

Together $(\pi^*, \phi_{\pi^*}^{BR})$ constitute a Stackelberg equilibrium (SE). If, additionally, the follower breaks ties in the leader's favor, these are a Strong Stackelberg equilibrium (SSE).

A natural question arises: is the leader's optimal policy in a DSSG guaranteed to be Markov stationary (that is, only depend on current state)? Turns out, it is not, as the following example illustrates. Suppose that the DSSG has three states, i.e., $S = \{1,2,3\}$, and the leader and the follower have two actions each, $A_L = \{U,D\}$ for the leader and $A_F = \{L,R\}$ for the follower. Let initial state be $s = 1$ and suppose that the following transitions happen deterministically and independently of either player's decisions: $T_{12} = 1$, $T_{23} = 1$, $T_{33} = 1$, that is, the process starts at state 1, then moves to state 2, then, finally, to state 3, which is an absorbing state. In state $s = 1$ only the follower's actions have an effect on payoffs, which is as follows: $R_L(1,\cdot,L) = -M$, $R_L(1,\cdot,R) = 0$, $R_F(1,\cdot,L) = \varepsilon$, $R_F(1,\cdot,R) = 0$, where M is an arbitrarily large number and $\varepsilon << M$. In state $s = 2$, in contrast, only the leader's actions have an effect on payoffs: $R_L(2,U,\cdot) = R_L(2,D,\cdot) = 0$, $R_F(2,U,\cdot) = -M$, $R_F(2,D,\cdot) = 0$. Suppose that the discount factors $\gamma_L = \gamma_F$ are close to 1. First, note that a Markov stationary policy for the leader would be independent of the follower's action in state 2, and, consequently, the follower's best response is to play L, giving the leader a payoff of $-M$. On the other hand, if the leader plays the following non-Markov policy: play U when the follower plays L and D otherwise, the follower's optimal policy is to play R, and the leader receives a payoff of 0. Since M is arbitrarily large, the difference between an optimal and best stationary policy is arbitrarily large.

The consequence of the fact that the leader's optimal policy can have arbitrary dependence on history is that DSSGs are extremely difficult to solve, except in special cases, such as when the game is zero-sum (in which case it is equivalent to a stochastic game, which can be solved in polynomial time [Littman 1994]).

Next, we consider the important special case of APGs [Vorobeychik et al. 2012, 2014], and present solution techniques for these.

1.5.2 Adversarial Patrolling Games

Formally, an *adversarial patrolling game (APG)* can be described by the tuple $\{T, U_d^c(i), U_d^u(i), U_a^c(i), U_a^u(i), \gamma, G\}$, where T is the set of n targets patrolled by the defender, $U_d^c(i)$ and $U_d^u(i)$ are the utilities to the defender if an attacker chooses a target $i \in T$ when it is patrolled and not, respectively, while $U_a^c(i)$ and $U_a^u(i)$ are the corresponding attacker utilities, $\gamma \in (0,1)$ is the discount factor (in some cases, we also allow $\gamma = 1$), and $G = (T,E)$ is a graph with targets as vertices and E the set of directed edges constraining defender patrolling moves between targets. It is useful to consider the representation of this graph as an adjacency matrix A, where $A_{ij} = 1$ if and only if there is an edge from target i to target j.

The game proceeds in a (possibly infinite) sequence of steps in which the defender moves between targets (subject to the constraints imposed by G), while the attacker chooses the time and target of attack. The defender's (stochastic) patrolling policy is a schedule π which can in general be an arbitrary function from all observed history (i.e., the sequence of targets patrolled in the past) to a probability distribution over the targets patrolled in the next iteration. The attacker is presumed to know the defender's policy π at the time of decision. At each time step t the attacker observes the defender's current location i and may choose to wait or to attack an arbitrary target $j \in T$. If an attacker waits, he receives no immediate utility, while attacking a target j gains the attacker $U_a^c(i)$ if it is covered by the defender at time $t+1$ and $U_a^u(i)$ if it is not. We denote the attacker's policy by ϕ. We say that a policy (π or ϕ) is *Markovian* if it only depends on the current location of the defender, and we call it *stationary Markovian* if it additionally has no dependence on time.

US Coast Guard's Patrolling Problem as an APG: US Coast Guard (USCG) safeguards important infrastructure at US coasts, ports, and inland waterway. Given a particular port and a variety of critical infrastructure that an adversary may choose to attack, USCG conducts patrols to detect an adversary and protect this infrastructure. However, while the adversary has the opportunity to observe patrol patterns, limited security resources imply that USCG patrols cannot be at every location at all times [An et al. 2011]. In the APG framework, USCG is the defender, while a terrorist group (for example) is an attacker who can conduct surveillance and can both observe the current location of the patrols and obtain a good estimate of the stochastic patrolling policy deployed.

We now observe that adversarial patrolling games can be naturally represented as DSSGs. Specifically, states in APGs correspond to the set of targets T, as well as an absorbing state s.

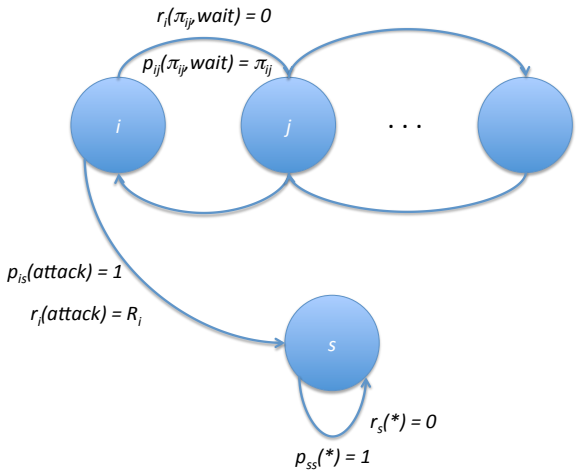

Figure 1.1 Schematic illustration of APG as a DSSG, showing example targets-states i and j, as well the absorbing state s. $p_{ij}(\cdot)$ denotes the transition probability, as a function of the probability π_{ij} that the defender moves from i to j and whether or not the attacker chooses "wait" or "attack". Note that if the attacker attacks, the DSSG transitions to the absorbing state with probability 1, independent of π_{ij}.

Defender actions in each state are the targets j that he can move to in a single time step, while attacker actions are to wait or to attack (for the moment, we will assume that we can compute expected utilities when attacker chooses to attack; we deal with the issue of which targets are attacked below). The state transitions are actually deterministic, conditional on player actions: if the attacker chooses to attack, the system always transitions to the absorbing state s; otherwise, the next target is completely determined by the defender's action. Finally, if the attacker waits, our baseline model involves zero reward accruing to both players. Figure 1.1 offers a schematic illustration of APG as a stochastic game.

1.5.3 Solving Zero-Sum APGs

Consider a special case of zero-sum APGs, where $U_d^c(i) = -U_a^c(i)$ and $U_d^u(i) = -U_a^u(i)$. In this case, APGs (and associated DSSGs) become equivalent to *stochastic games*, since the order of player moves is no longer important. A particularly important consequence is that while in general SSE solutions to DSSGs need not be in Markov stationary policies (as our example above illustrated), stochastic games always have equilibria that are Markov stationary [Filar and Vrieze 1997]. Thus, for zero-sum APGs, we can focus exclusively on stationary policies, and denote by π_{ij} the (stationary) probability that the defender moves from target i to j.

Since APG is a special case of a stochastic game, we can adapt the non-linear programming formulation for computing a Nash equilibrium in general zero-sum stochastic games by Filar

and Vrieze [Filar and Vrieze 1997] to our setting. One minor addition to their formulation that we find useful below is to represent the constraints on the defender's action imposed by the graph G as a set of constraints

$$\pi_{ij} \leq A_{ij} \quad \forall\, i,j \in T. \tag{1.16}$$

Let v_i be the expected attacker value of starting in state i. We can formulate the defender's problem as the following mathematical program:

$$\min_{\pi,v} \sum_i v_i \tag{1.17a}$$

s.t. :

$$\pi_{ij} \geq 0 \qquad \forall\, i,j \in T \tag{1.17b}$$

$$\sum_j \pi_{ij} = 1 \qquad \forall\, i \in T \tag{1.17c}$$

$$\pi_{ij} \leq A_{ij} \qquad \forall\, i,j \in T \tag{1.17d}$$

$$v_i \geq (1-\pi_{ij})U_a^u(j) + \pi_{ij}U_a^c(j) \quad \forall\, i,j \in T \tag{1.17e}$$

$$v_i \geq \gamma \sum_j \pi_{ij} v_j \qquad \forall i \in T. \tag{1.17f}$$

Constraints 1.17b and 1.17c simply constrain defender policy to be a valid probability distribution. The key constraints 1.17e and 1.17f are easiest to think about if we fix defender policy π and just consider the MDP faced by the attacker. The right-hand-side of Constraint **??** corresponds to expected utility of attacking (since attack moves the MDP into an absorbing state s and no further earnings follow), while right-hand-side of Constraint 1.17f is the expected value of waiting (immediate reward is 0 for a waiting action). The constraints then arise because the state v_i must be the expected utility of making the best action choice, and minimizing the objective ensures that these values bind to *some* action in every state. While this formulation is a non-convex non-linear program, it turns out that all local optima are also global optima [Filar and Vrieze 1997].

1.5.4 Solving General-Sum APGs

The approach above for effectively computing equilibria in zero-sum APGs relies strongly on the particular structure of the zero-sum game. We now present a general formulation of the problem of computing SSE in APGs which does not require the game to ber zero sum. First, let $\beta(i)$ be the distibution over the initial location of the defender (which in many cases may simply be a fixed initial location). Let $v_d(i)$ be the defender's value (expected discounted cumulative reward) of the defender being in location i, and let $v_a(i)$ be the attacker's value when defender is at location i. For the attacker, we now also need to explicitly represent their policy best response. We again use ϕ to denote the attacker's policy, but our notation for it

will be somewhat different than earlier. Specifically, we use ϕ_{ij} to denote the decision of the attacker whether or not to attack a location (target) j when the defender is at location i, and ϕ_{iw} to indicate whether the attacker chooses to wait when seeing the defeder in location i (treating the decision to wait is effectively an artificial target w). The full mixed-integer non-linear program (MINLP) for solving general-sum APGs can be represented as follows:

$$\max_{\pi,\phi,v_a,v_d} \sum_i \beta(i) v_d(i) \tag{1.18a}$$

s.t. :

$$\pi_{ij} \geq 0 \quad \phi_{ij} \in \{0,1\}, \phi_{iw} \in \{0,1\} \qquad \forall\, i,j \in T \tag{1.18b}$$

$$\sum_j \pi_{ij} = 1 \quad \sum_j \phi_{ij} + \phi_{iw} = 1 \qquad \forall\, i \in T \tag{1.18c}$$

$$\pi_{ij} \leq A_{ij} \qquad \forall\, i,j \in T \tag{1.18d}$$

$$v_i \geq (1-\pi_{ij}) U_a^u(j) + \pi_{ij} U_a^c(j) \quad \forall\, i,j \in T \tag{1.18e}$$

$$v_i \geq \gamma \sum_j \pi_{ij} v_j \qquad \forall i \in T \tag{1.18f}$$

$$0 \leq v_a(i) - [(1-\pi_{ij}) U_a^u(j) + \pi_{ij} U_a^c(j)] \leq (1-\phi_{ij}) Z \qquad \forall i,j \in T \tag{1.18g}$$

$$v_a(i) - \gamma \sum_j \pi_{ij} v_a(j) \leq (1-\phi_{iw}) Z \qquad \forall\, i \in T \tag{1.18h}$$

$$v_d(i) - [(1-\pi_{ij}) U_d^u(j) + \pi_{ij} U_d^c(j)] \leq (1-\phi_{ij}) Z \qquad \forall i,j \in T \tag{1.18i}$$

$$v_d(i) - \gamma \sum_j \pi_{ij} v_d(j) \leq (1-\phi_{iw}) Z \qquad \forall\, i \in T. \tag{1.18j}$$

Note that this representation echoes back to our solution of Bayesian Stackelberg games in Section 1.2; in particular, the nature of the constraints used to compute the attacker's best response has very much the same structure.

Much as we can represent the general-sum APGs equilibrium computation as a MINLP, that is of only marginal help: such problems are extremely challenging to solve in practice. Even if we were to discretize the defender's policy π, turning this into a mixed-integer linear program, the resulting problems are still difficult to solve. Indeed, scalable approaches for solving general sum APGs—and all the more so, DSSGs—remain elusive.

1.6 Conclusion

Computational game theory for security has been a very active field of research for a number of years now, and the volume of both modeling and computational approaches far exceeds what can be reasonably covered in a short chapter. Consequently, this chapter was scoped somewhat narrowly on a collection of models and computational methods for several classes of Stackelberg security games. It is nevertheless worth specifically mentioning two important paradigms that the chapter has ignored entirely. First, there is now considerable and impor-

tant line of work on game-theoretic models of deception in security settings. Some of this work pertains to the strategic deployment of honeypots [Anwar and Kamhoua 2022, Garg and Grosu 2007, Kiekintveld et al. 2015], while other efforts consider problems involving choosing configurations of devices on a network as well as deceptive presentation of such configurations [Schlenker et al. 2018, Shi et al. 2020, Thakoor et al. 2019, Wu et al. 2021]. Second, there is much work that aims to relax the assumption of perfectly rational behavior by adversaries. In this line of work, a common consideration is the merging of behavioral game-theoretic models with Stackelberg-style approaches for computing optimal defender strategies [Abbas et al. 2017, Nguyen et al. 2013, Yang et al. 2013].

Bibliography

A. E. Abbas, M. Tambe, and D. Von Winterfeldt. 2017. *Improving Homeland Security Decisions*. Cambridge University Press.

B. An, J. Pita, E. Shieh, M. Tambe, C. Kiekintveld, and J. Marecki. March 2011. Guards and protect: Next generation applications of security games. In *SIGECOM*, volume 10, pp. 31–34.

A. H. Anwar and C. A. Kamhoua. 2022. Cyber deception using honeypot allocation and diversity: A game theoretic approach. In *Annual Consumer Communications & Networking Conference*, pp. 543–549.

D. Bertsimas and J. N. Tsitsiklis. 1997. *Introduction to Linear Optimization*. Athena Scientific.

K. J. Cormican, D. P. Morton, and R. K. Wood. 1998. Stochastic network interdiction. *Operations Research*, 46(2): 184–197.

F. Fang, T. Nguyen, B. Ford, N. Sintov, and M. Tambe. 2015. Introduction to green security games. In *International Joint Conference on Artificial Intelligence*.

J. Filar and K. Vrieze. 1997. *Competitive Markov Decision Processes*. Springer-Verlag.

N. Garg and D. Grosu. 2007. Deception in honeynets: A game-theoretic analysis. In *IEEE SMC Information Assurance and Security Workshop*, pp. 107–113.

M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe. 2011. A double oracle algorithm for zero-sum security games on graphs. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 327–334. Citeseer.

C. Kiekintveld, V. Lisỳ, and R. Píbil. 2015. Game-theoretic foundations for the strategic use of honeypots in network security. In *Cyber Warfare: Building the Scientific Foundation*, pp. 81–101. Springer.

J. Letchford and Y. Vorobeychik. 2013. Optimal interdiction of attack plans. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 199–206.

M. L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 157–163.

T. Nguyen, R. Yang, A. Azaria, S. Kraus, and M. Tambe. 2013. Analyzing the effectiveness of adversary modeling in security games. In *AAAI Conference on Artificial Intelligence*, volume 27, pp. 718–724.

S. Panda and Y. Vorobeychik. 2017. Near-optimal interdiction of factored mdps. In *Conference on Uncertainty in Artificial Intelligence*.

S. Panda and Y. Vorobeychik. 2018. Scalable initial state interdiction for factored mdps. In *International Joint Conference on Artificial Intelligence*.

B. S. Pay, J. R. Merrick, and Y. Song. 2019. Stochastic network interdiction with incomplete preference. *Networks*, 73(1): 3–22.

A. Schlenker, O. Thakoor, H. Xu, F. Fang, M. Tambe, L. Tran-Thanh, P. Vayanos, and Y. Vorobeychik. 2018. Deceiving cyber adversaries: A game theoretic approach. In *International Conference on Autonomous Agents and MultiAgent Systems*, pp. 892–900.

Z. R. Shi, A. D. Procaccia, K. S. Chan, S. Venkatesan, N. Ben-Asher, N. O. Leslie, C. Kamhoua, and F. Fang. 2020. Learning and planning in the feature deception problem. In *International Conference on Decision and Game Theory for Security*, pp. 23–44.

J. C. Smith and Y. Song. 2020. A survey of network interdiction models and algorithms. *European Journal of Operational Research*, 283(3): 797–811.

M. Tambe. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

O. Thakoor, M. Tambe, P. Vayanos, H. Xu, C. Kiekintveld, and F. Fang. 2019. Cyber camouflage games for strategic deception. In *International Conference on Decision and Game Theory for Security*, pp. 525–541.

Y. Vorobeychik and M. Pritchard. 2020. Plan interdiction games. In *Adaptive Autonomous Secure Cyber Systems*, pp. 159–182. Springer.

Y. Vorobeychik, B. An, and M. Tambe. 2012. Adversarial patrolling games. In *2012 AAAI Spring Symposium Series*.

Y. Vorobeychik, B. An, M. Tambe, and S. Singh. 2014. Computing solutions in infinite-horizon discounted adversarial patrolling games. In *International Conference on Automated Planning and Scheduling*.

R. K. Wood. 1993. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2): 1–18.

J. Wu, C. Kamhoua, M. Kantarcioglu, and Y. Vorobeychik. 2021. Learning generative deception strategies in combinatorial masking games. In *International Conference on Decision and Game Theory for Security*, pp. 98–117.

R. Yang, A. X. Jiang, M. Tambe, and F. Ordonez. 2013. Scaling-up security games with boundedly rational adversaries: A cutting-plane approach. In *International Joint Conference on Artificial Intelligence*.