Gradient Methods for Stackelberg Security Games

Kareem Amin
University of Michigan
amkareem@umich.edu

Satinder Singh
University of Michigan
baveja@umich.edu

Michael Wellman
University of Michigan
wellman@umich.edu

ABSTRACT

Stackelberg games are two-stage games in which the first player (called the leader) commits to a strategy, after which the other player (the follower) selects a best-response. These types of games have seen numerous practical application in security settings, where the leader (in this case, a defender) must allocate resources to protect various targets. Real world applications include the scheduling of US federal air marshals to international flights, and resource allocation at LAX airport. However, the best known algorithm for solving *general* Stackelberg games requires solving Integer Programs, and fails to scale beyond a few (significantly smaller than 100) number of leader actions, or follower types. In this paper, we present a new gradient-based approach for solving large Stackelberg games in security settings. Large-scale control problems are often solved by restricting the controller to a rich parameterized class of policies; the optimal control can then be computed using Monte Carlo gradient methods. We demonstrate that the same approach can be taken in a *strategic* setting. We evaluate our approach empirically, demonstrating that it can have negligible regret against the leader's true equilibrium strategy, while scaling to large games.

1. INTRODUCTION

Stackelberg games have received significant attention in the context of security applications, where a defender (the leader in the Stackelberg game) must deploy a limited number of security resources to protect a set of vulnerable targets to guard against an attacker (the follower in the Stackelberg game). Algorithms for these games have been deployed in real-world settings, e.g., to generate checkpoints and patrols at the Los Angeles International Airport [Pita et al., 2009], as well as to schedule US Federal Air Marshals (FAMS) to flights [Jain et al., 2010b, Tsai et al., 2009].

The best known solver for general Bayesian Stackelberg games is the DOBBS algorithm [Paruchuri et al., 2008], which was the first to formulate a Stackelberg game (given in normal form) as a Mixed Integer Linear Program (MILP). Since then, a great deal of research has been devoted to algorithms that scale to games of the size encountered in real settings. This has led to the development of a general class of Stackelberg Security Games (SSGs) [Korzhyk et al., 2011], which captures many of the key features of these real settings, along with algorithms including ORIGAMI, ERASER, ERASER-C [Kiekintveld et al., 2009], and ASPEN [Jain et al., 2010a], which can handle a large number of defender actions. With the exception of ASPEN, these algorithms work by assuming additional structure on the SSG, specifically on the pure strategy set of the defender. Similarly, the HBGS algorithm [Jain et al., 2011] scales to multiple attacker types by assuming hierarchical structure on the attacker types. Hence, these algorithms are able to solve MILPs more compact than would be generated from a game expressed in normal form.[1] Among approximate methods, Monte-Carlo approaches have been used to solve games with infinite types [Kiekintveld et al., 2011], but also makes assumptions on what defender pure strategies are feasible. The HUNTER algorithm [Yin and Tambe, 2012], in contrast, can solve Stackelberg games with limited additional assumptions, but searches a game tree whose depth scales with the number of attacker types, and whose branching factor is the number of available targets.

We introduce a different approach for finding good defender strategies in Stackelberg security games, which avoids imposing structure on the defender's *pure strategies*, the attackers' types, or the players' utilities. Instead, we propose restricting the search for defender strategies within a rich, but parameterized, class of mixed strategies. In a standard Stackelberg equilibrium the defender (who is the leader), plays an (unrestricted) mixed strategy, knowing that the attacker will select a best-response. In contrast, we will consider games where the defender's choice of mixed strategy must come from some fixed class of distributions.

This type of assumption is analogous to successful methods in AI and reinforcement learning such as policy gradient methods [Baxter and Bartlett, 2001], which avoid making assumptions about the dynamics of the environment, but rather, constrain the search for a good policy to within a parameterized family of policies. In this work, we will demonstrate how a similar approach can be applied to a *strategic* setting. We then apply Monte-Carlo gradient methods, a procedure we call STACKGRAD, to find solutions to the defender's optimization problem. In contrast to an ordinary optimal control problem, a unique feature of our derivation is that this gradient computation must pass through the attacker's response function. In order to maintain differentiability, we consider a smoothed version of the attacker's response function.

Figure 1 provides a schematic of our approach. We begin with a Stackelberg game, consisting of the defender's set of mixed strategies, Δ, the attacker's response function g, and the defender's utility for playing $\mathcal{D} \in \Delta$, denoted $U(\mathcal{D}, g(\mathcal{D}))$. We then formulate an approximate Stackelberg game by restricting $\mathcal{D} \in \Delta(\Theta) \subset \Delta$ and smoothing $g \Rightarrow \tilde{g}$, where Θ denotes the parameterization of the defender strategy space. Finally, we solve for the defender's best strategy in the restricted game via gradient ascent.

[1] ORIGAMI makes the most restrictive assumptions on the game, and does not need to solve an MILP at all.

Appears in: *The AAMAS-16 Workshop on Security and Multiagent Systems (SecMAS). Part of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*

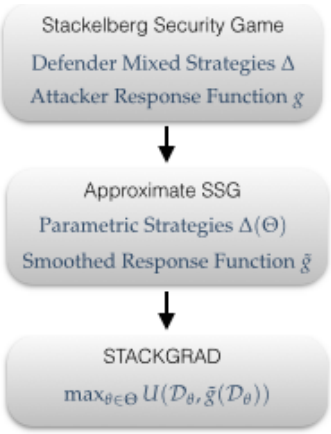

Figure 1: STACKGRAD optimizes the approximate game given considering a parametric class of leader (defender) strategies and by smoothing the follower's (attacker's) response function.

While any parameterized class of defender mixed strategies may be utilized, we propose two rich classes which yield heuristics with no computational dependence on the number of attacker types, and only mild dependencies on the size of the defender's pure strategy set. We then demonstrate empirically that STACKGRAD not only scales to large games, but also finds mixed strategies that are close to the defender's true Stackelberg optimal strategy.

Our primary contributions are in the derivation of a new algorithm for solving Stackelberg games approximately and in its empirical evaluation. Specifically, we define STACKGRAD, by deriving an approximate stochastic gradient of the defender's utility with respect to its strategy parameters through the (smoothed) best response function of the attacker. We argue analytically that STACKGRAD has no computational dependence on the number of attacker types in a Bayesian Stackelberg game, being able to handle any number (even infinite types). We then demonstrate empirically that, despite searching within a restricted class of mixed strategies, the solutions found by STACKGRAD have almost the same payoff to the defender as the true (unrestricted) Stackelberg optimum, computed directly using a solver for the unrestricted game. We then demonstrate the STACKGRAD has only a mild computational dependence on the number of pure strategies, by considering a game inspired by the Federal Air Marshal (FAMS) domain.

2. PRELIMINARIES

2.1 General Security Games

We consider the *Stackelberg security game* introduced by Kiekintveld et al. [2009] (see Korzhyk et al. [2011] for a good overview). An SSG is a two-player game between a defender and attacker. The attacker selects a target from a set $T = \{t_1, \ldots, t_N\}$. The defender prevents attacks by guarding targets with various resources $R = \{r_1, \ldots, r_K\}$. In the most general setting considered in previous work, resources may simultaneously cover a set of targets. For example, resources might be air marshals, and targets might be airline flights [Tsai et al., 2009]. An air marshal might protect a number of targets by flying a circuit which starts and ends at the marshal's home city; each flight along the circuit is considered covered. This is modeled by having the defender assign resources to *schedules*, where a schedule $S \subset T$ is just a subset of the possible targets. Assigning a resource r_k to schedule S corresponds to covering each $t \in S$ with resource r_k. The set of schedules to which a resource r_k may be assigned is denoted by $\mathcal{S}_k$. $\mathcal{S}_k$ specifies the constraints induced by the domain. For example, in the air marshal SSG, $\mathcal{S}_k$ comprises sets of flights that can form a circuit that starts and ends in the marshal's home city.

A defender's pure strategy is an assignment of resources to schedules, which we denote by $\mathbf{s} \in \mathcal{S} \triangleq \mathcal{S}_1 \times \cdots \times \mathcal{S}_K$. Strategy component $\mathbf{s}_k$ is the schedule to which resource r_k is assigned. Strategy $\mathbf{s}$ induces a coverage vector $\mathbf{c} \in \{0,1\}^N$ (sometimes denoted $\mathbf{c}(\mathbf{s})$ when we wish to emphasize the dependence on $\mathbf{s}$) indicating which targets are covered: $\mathbf{c}_n = 1$ if $t_n \in \mathbf{s}_k$ for some k. An attacker's pure strategy is simply a target $t_n \in T$.

Our methods also allow for non-binary coverage. That is, we can take $\mathbf{c}(\mathbf{s})$ to be an arbitrary function from $\mathcal{S}$ to $[0,1]^K$ representing the assignment of varying resource quantities to achieve degrees of target coverage. However, to simplify our expressions, and to stay consistent with previous work, we assume binary $\mathbf{c}_n$ unless otherwise noted.

If the defender plays a mixed strategy—a distribution $\mathcal{D}$ over pure strategies—we use $\bar{\mathbf{c}} \in [0,1]^d$ to denote the probability that each target is covered. In other words, $\bar{\mathbf{c}}(\mathcal{D}) = \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}[\mathbf{c}(\mathbf{s})]$.

The players' utilities are functions of which target is attacked, and whether that target is covered (by any resource). Attacker and defender utilities are denoted $U_a : \{0,1\} \times T \to \mathbb{R}$ and $U_d : \{0,1\} \times T \to \mathbb{R}$ respectively. $U_a(0, t_n)$, for example, specifies the utility to the attacker if t_n is attacked and uncovered. The expected payoffs for playing defender mixed strategy $\mathcal{D}$ and attacker pure strategy t_n are given by:

$$\mathbf{U}_a(\mathcal{D}, t_n) = \bar{\mathbf{c}}(\mathcal{D})U_a(1, t_n) + (1 - \bar{\mathbf{c}}(\mathcal{D}))U_a(0, t_n)$$
$$\mathbf{U}_d(\mathcal{D}, t_n) = \bar{\mathbf{c}}(\mathcal{D})U_d(1, t_n) + (1 - \bar{\mathbf{c}}(\mathcal{D}))U_d(0, t_n)$$

In a security game, it is assumed that for all n, $\Delta_{a,n} \triangleq U_a(0, t_n) - U_a(1, t_n) \geq 0$ and $\Delta_{d,n} = U_d(1, t_n) - U_a(0, t_n) \geq 0$. The attacker never prefers that its target is covered. Similarly, the defender never prefers that the attacker's target is uncovered.

2.2 Stackelberg Equilibria

The solution concept typically applied to security games is that of a *Stackelberg Equilibrium*, as opposed to the more conventional *Nash Equilibrium*. In a two-player Stackelberg game, one player—termed *leader*—first commits to a strategy. The other player—termed *follower*—observes this commitment, and plays a best response to the leader's chosen strategy. In security games, the defender is the leader and the attacker follows. This leader-follower interaction is argued to be better suited than simultaneous moves for modeling security domains, since after the defender deploys its resources, that deployment is subject to scrutiny by a malicious agent, who can use this information when deciding on its point of attack.

Due to the two-stage nature of the game, it is convenient to think of the follower as selecting a *response function* which maps each of the leader's mixed strategies to a pure strategy. With Δ the set of defender mixed strategies, we denote a response function by $g : \Delta \to T$. Upon observing leader mixed strategy $\mathcal{D}$, the follower plays pure strategy $g(\mathcal{D})$.

We say that $(\mathcal{D}, g)$ forms a *Strong Stackelberg Equilibrium* (SSE) if, informally: (1) given response function g, the leader maximizes its payoff by playing $\mathcal{D}$, (2) g is not just a response function, but always returns a follower *best* response, and (3) if there are multiple follower best-response functions, g selects the one most beneficial to the leader.

Condition (3) is what distinguishes a *Strong* Stackelberg Equilibrium from an ordinary Stackelberg Equilibrium. It can be argued that this is a reasonable solution concept if one believes that the

leader can always force the attacker to break ties in its favor. For our purposes, it will be necessary only to observe that the payoff to the leader in a SSE is the most that a leader can hope to guarantee against a rational follower. Below we state the definition of an SSE formally in the context of a security game.

DEFINITION 1 (FOLLOWER BEST-RESPONSE). *A follower response function $g : \Delta \to T$ is a* best-response *function if for any other response function $g' : \Delta \to T$ and leader strategy $\mathcal{D} \in \Delta$, $\mathbf{U}_a(\mathcal{D}, g(\mathcal{D})) \geq \mathbf{U}_a(\mathcal{D}, g'(\mathcal{D}))$.*

DEFINITION 2 (STRONG STACKELBERG EQUILIBRIUM). *$(\mathcal{D}, g)$ where $\mathcal{D} \in \Delta$ and $g : \Delta \to T$ is a* Strong Stackelberg Equilibrium *iff:*

1. *For all $\mathcal{D}' \in \Delta$, $\mathbf{U}_d(\mathcal{D}, g(\mathcal{D})) \geq \mathbf{U}_d(\mathcal{D}', g(\mathcal{D}'))$*
2. *g is a best-response function.*
3. *For any $\mathcal{D}' \in \Delta$ and best-response function g', $\mathbf{U}_d(\mathcal{D}', g(\mathcal{D}')) \geq \mathbf{U}_d(\mathcal{D}', g'(\mathcal{D}'))$*

The definition of a SSE ensures that, although there may be multiple such equilibria, they all give the same payoff to the defender. Given an instance of a security game $\mathcal{G}$, let $V_{\mathcal{G}}$ denote the payoff to the defender in equilibrium.

2.3 Bayesian Stackelberg Games

An extension of the game described in the previous section allows the defender to model uncertainty over the potential attackers by a prior q over a set of attacker *types* Λ. Letting g_λ denote the best response of attacker λ, the defender's utility in the Bayesian setting is given by:

$$\mathbf{U}_d(\mathcal{D}, \{g_\lambda\}) = \sum_{\lambda \in \Lambda} q(\lambda) \mathbf{U}_d(\mathcal{D}, g_\lambda(\mathcal{D})). \tag{1}$$

As with a single attacker, if g_λ breaks ties in a consistent manner we can uniquely define $V_{\mathcal{G}}$ in the Bayesian setting. Given an arbitrary defender strategy $\mathcal{D}$, we define the competitive ratio of the defender's choice of strategy against the SSE solution.

DEFINITION 3 (COMPETITIVE RATIO). *Given an instance of a Bayesian Stackelberg game $\mathcal{G}$, and defender strategy $\mathcal{D}$, define $\mathrm{R}(\mathcal{D}) = \mathbf{U}_d(\mathcal{D}, \{g_\lambda\})/V_{\mathcal{G}}$.*

The defender would like to maximize equation (1), or short of doing so, find a strategy $\mathcal{D}$ which attains a large competitive ratio. In this work, we propose using gradient methods to search for such strategies. However, in order to do so $\mathbf{U}_d$ will have to be differentiable with respect to its argument. In the next section, we describe the approximations needed to ensure this.

3. STACKELBERG GAME APPROXIMATION

We now consider an approximation to general Bayesian Stackelberg games. We (1) assume a probabilistic model for the attacker, and (2) restrict the set of strategies available to the defender.

Recall that the *softmax* function $\sigma_\eta : \mathbb{R}^N \times \{1, \ldots, N\} \to [0, 1]$ provides a probabilistic approximation to the maximum element of the set. Given a vector $\mathbf{x} \in \mathbb{R}^N$, we let $\sigma_\eta(\mathbf{x}, i) = \frac{\exp(\eta \mathbf{x}_i)}{\sum_{i=1}^N \exp(\eta \mathbf{x}_i)}$. The categorical distribution over the set $\{1, \ldots, N\}$, which selects element i with probability $\sigma_\eta(\mathbf{x}, i)$ is called the *softmax distribution*. As the inverse temperature $\eta \to \infty$, softmax concentrates mass on the indices belonging to the maximum elements of $\mathbf{x}$.

The first step to approximating the security game of Section 2 is to have the attacker select its target according to the softmax distribution over its utilities. Let $\mathbf{U}_\lambda$ denote the utility function corresponding to attacker type λ. Let us also define

$$\mathbf{u}_\lambda(\mathcal{D}) = [\mathbf{U}_\lambda(\mathcal{D}, t_1), \ldots, \mathbf{U}_\lambda(\mathcal{D}, t_N)],$$

the vector of expected payoffs to the attacker for each choice of target, assuming the defender plays mixed strategy $\mathcal{D}$. In the security game approximation, we assume that conditioned on the defender's choice of $\mathcal{D}$, and a fixed choice of η (a parameter of the approximation), the attacker selects target t_n with probability $\sigma_\eta(\mathbf{u}_\lambda(\mathcal{D}), n)$. We note that while we utilize the softmax for purely analytic reasons (allowing us to compute gradients), previous work in security games has made similar assumptions in order to model attackers with bounded rationality, where this is referred to as the *quantal response* model [Yang et al., 2012].

Second, we restrict the defender to select distributions from a parameterized class $\Delta_\Theta \subset \Delta$, where Θ denotes a set of parameters, and for each $\theta \in \Theta$, there is a corresponding distribution $\mathcal{D}_\theta \in \Delta_\Theta$. We denote the probability mass function (over $\mathcal{S}$) of $\mathcal{D}_\theta$ by $p(\cdot \mid \theta)$.

Putting these approximations together gives us a stochastic optimization problem from the perspective of the defender (summarized in the model below). First nature draws attacker type λ according to q. Given λ, η, and $\theta \in \Theta$, an attacker's (now random) response t_n is determined by a draw from the categorical distribution defined by $\sigma_\eta(\mathbf{u}_\lambda(\mathcal{D}_\theta), \cdot)$, rather than an exact best-response function g_λ. Upon independently drawing $\mathbf{s} \sim \mathcal{D}_\theta$, the defender receives $U_d(\mathbf{c}_n(\mathbf{s}), t_n)$.

Security Game Approximation

Defender selects $\theta \in \Theta$.
Nature draws $\lambda \sim q$
$n \sim \sigma_\eta(\mathbf{u}_\lambda(\mathcal{D}_\theta), \cdot)$, $\mathbf{s} \sim \mathcal{D}_\theta$
Defender receives payoff $U_d(\mathbf{c}_n(\mathbf{s}), t_n)$.

The goal of the defender in the security game approximation is to maximize its expected payoff, which is given by the following function:

$$\tilde{\mathbf{U}}_d(\theta) = \mathbf{E}\left[U_d(\mathbf{c}_n(\mathbf{s}), t_n)\right], \tag{2}$$

The expectation here is taken according to the process just described.

If we take Δ_Θ identical to Δ and $\eta \to \infty$, then the logic of the model is that of a leader-follower game, and the defender strategy θ^* that maximizes $\tilde{\mathbf{U}}_d(\theta)$ is precisely its strategy in a Stackelberg equilibrium. However, maximizing $\tilde{\mathbf{U}}_d$ may not be tractable for larger games. If, on the other hand, Δ_Θ is restrictive ($\Delta_\Theta \neq \Delta$), the strategy $\mathcal{D}_{\theta^*}$ might suffer regret against an attacker best-response function g, but the parametric class may allow $\mathbf{U}_d$ to be efficiently maximized. Balancing these two effects is important. In the next section, we describe how given a fixed parametric class Θ, $\tilde{\mathbf{U}}_d$ can be maximized using gradient methods.

4. MONTE-CARLO GRADIENT ESTIMATE: STACKGRAD

If $\Theta \subset \mathbb{R}^d$ for some d, then gradient algorithms are natural candidates for maximizing Equation 2. However, to compute $\nabla \tilde{\mathbf{U}}_d(\theta)$ even at a single point θ, it might be necessary to sum over all of Λ, and the entire support of $\mathcal{D}_\theta$, which even for a parametric class of distributions might include all elements of $\mathcal{S}$. A more tractable approach is to take a Monte-Carlo estimate of $\nabla \tilde{\mathbf{U}}_d(\theta)$, which need not depend on the size of $\mathcal{S}$, and has *no* dependence on $|\Lambda|$.

The first step to producing such an estimate is to derive for any θ, an unbiased estimate $\tilde{\gamma}_\theta$ of $\nabla\tilde{\mathbf{U}}_d(\theta)$. Recall that given the defender's choice of parameter θ, the probability of schedule $\mathbf{s} \in \mathcal{S}$ is given by $p(\mathbf{s} \mid \theta)$. Let us denote the probability that the defender selects pure strategy $\mathbf{s}$, and an attacker of type λ selects target t_n by $p(\mathbf{s}, t_n \mid \theta, \lambda)$.

Thus, with probability $q(\lambda)p(\mathbf{s}, t_n \mid \theta, \lambda)$, the defender receives $U_d(\mathbf{s}, t_n)$, and so the gradient is

$$\nabla\tilde{\mathbf{U}}_d(\theta) = \sum_{\lambda\in\Lambda}\sum_{\mathbf{s}\in\mathcal{S}, t_n\in T} q(\lambda)U_d(\mathbf{c}_n(\mathbf{s}), t_n)\nabla p(\mathbf{s}, t_n \mid \theta, \lambda)$$

$$= \sum_{\lambda\in\Lambda}\sum_{\mathbf{s}\in\mathcal{S}, t_n\in T} \frac{p(\mathbf{s}, t_n \mid \theta, \lambda)}{p(\mathbf{s}, t_n \mid \theta, \lambda)} q(\lambda)U_d(\mathbf{c}_n(\mathbf{s}), t_n)\nabla p(\mathbf{s}, t_n \mid \theta, \lambda)$$

$$= \mathbb{E}\left[U_d(\mathbf{c}_n(\mathbf{s}), t_n)\frac{\nabla p(\mathbf{s}, t_n \mid \theta, \lambda)}{p(\mathbf{s}, t_n \mid \theta, \lambda)}\right]. \quad (3)$$

We can estimate $\nabla\tilde{\mathbf{U}}_d(\theta)$ by drawing m random samples according to the model. The gradient estimate for sample i is given by

$$\tilde{\gamma}_\theta^{(i)} = U_d(\mathbf{c}_n(\mathbf{s}^{(i)}), t_n^{(i)})\frac{\nabla p(\mathbf{s}^{(i)}, t_n^{(i)} \mid \theta, \lambda^{(i)})}{p(\mathbf{s}^{(i)}, t_n^{(i)} \mid \theta, \lambda^{(i)})},$$

and the overall gradient estimate by $\frac{1}{m}\sum_{i=1}^m \tilde{\gamma}_\theta^{(i)}$. This presumes that given realizations λ, $\mathbf{s}$ and t_n, the ratio $\frac{\nabla p(\mathbf{s}, t_n|\theta, \lambda)}{p(\mathbf{s}, t_n|\theta, \lambda)}$ can be efficiently computed. We examine classes Θ where this holds in the following sections.

For gradient methods to converge in expectation any unbiased estimate of $\nabla\tilde{\mathbf{U}}_d$ suffices (even taking $m = 1$), although lower variance estimates perform better in practice. Furthermore, $\frac{\nabla p(\mathbf{s}, t_n|\theta, \lambda)}{p(\mathbf{s}, t_n|\theta, \lambda)}$ depends on the realization $\lambda^{(i)}$, and not the entire set Λ. Thus, the computation required to estimate $\nabla\tilde{\mathbf{U}}_d$ is independent of $|\Lambda|$, and we expect such methods to scale to any number of attacker types (even infinitely many).

For completeness, we state the algorithm STACKGRAD, which is a projected stochastic gradient ascent, utilizing Monte-Carlo gradient estimates. Let $P_\Theta(x) = \min_{\theta\in\Theta}\|x - \theta\|_2$ denote the Euclidean projection of x onto Θ. We use the shorthand $\lambda, \mathbf{s}, t \sim M(\theta)$ to denote random variables sampled according to the stochastic approximation given defender parameter θ.

Algorithm STACKGRAD

Inputs: Class Θ, Horizon T, Sampling Parameter m
Initialize θ_0.
for $\tau = 1, \ldots, T$ **do**
  Sample $\lambda^{(i)}, \mathbf{s}^{(i)}, t_n^{(i)} \sim M(\theta_\tau)$ for $i = 1, \ldots, m$.
  Let $\tilde{\gamma}_\tau = \frac{1}{m}\sum_{i=1}^m U_d(\mathbf{c}_n(\mathbf{s}^{(i)}), t_n^{(i)})\frac{\nabla p(\mathbf{s}^{(i)}, t_n^{(i)}|\theta_\tau, \lambda^{(i)})}{p(\mathbf{s}^{(i)}, t_n^{(i)}|\theta_\tau, \lambda^{(i)})}$
  $\theta_\tau = P_\Theta(\theta_{\tau-1} + \frac{1}{\sqrt{\tau}}\tilde{\gamma}_\tau)$
end for
return θ_T

Standard results (e.g. Boyd et al. [2003]) tell us that after T iterations, the expected value of $\tilde{\mathbf{U}}_d(\theta_T)$ will be within $O(\frac{1}{\sqrt{T}})$ of a local maximum of the function $\tilde{\mathbf{U}}_d$. Therefore, if the quality of the defender strategy found by STACKGRAD, $R(\mathcal{D}_{\theta_T})$, is poor, it must either be because: (1) good mixed strategies for the *true* game exist only in $\Delta \setminus \Delta_\Theta$ (i.e. $\max_{\theta\in\Theta} V_\mathcal{G} - \tilde{\mathbf{U}}_d(\theta) \gg 0$), or (2) good defender strategies exist in Δ_Θ, but STACKGRAD converged to a suboptimal local maximum.

In what follows, we will see that in empirical validations, STACKGRAD exhibits good behavior on standard security games. This leads us to two questions, which we leave open for future theoretical research. (1) Are there classes of Stackelberg games, and parameterized strategies Θ, where the best strategy for the approximate game is guaranteed to be a good strategy for the true game? (2) Are there classes of Stackelberg games where STACKGRAD is guaranteed to exhibit good convergence properties?

5. INDEPENDENT RESOURCE ALLOCATION : CATEGORICAL DISTRIBUTIONS

In this section we demonstrate how to compute the ratio $\frac{\nabla p(\mathbf{s}, t_n|\theta, \lambda)}{p(\mathbf{s}, t_n|\theta, \lambda)}$ for a natural defender strategy class. We will then demonstrate how this parameterized class can be used to search for defender strategies in games with a large, even infinite number, of adversary types.

One simple, but rich class of defender strategies assumes that each defender resource is independently assigned to a schedule. For each resource k and schedule $s \in \mathcal{S}_k$, we assign resource k to s with some probability. This assignment is conducted independently across resources.

For each set of schedules $\mathcal{S}_k = \{s_{k,0}, \ldots, s_{k,d_k}\}$, we therefore introduce a parameter $\theta_{k,l}$ where for $l \geq 1$, $\theta_{k,l} \geq 0$ specifies the probability with which resource k is assigned to resource $s_{k,l}$. We require that $\sum_{l=1}^{d_k}\theta_{k,l} \leq 1$, and resource k is assigned to schedule $s_{k,0}$ with the remaining probability $\theta_{k,0} = 1 - \sum_{l=1}^{d_k}\theta_{k,l}$. Thus, this class of strategies is parameterized by $\Theta \subset \mathbb{R}^d$ where $d = \sum_{k=1}^K d_k$.

Notice that Δ_Θ is rich enough to describe any marginal distribution of an individual resource's assignment to schedules, including any pure strategy. However, Δ_Θ cannot capture mixed strategies which contain correlations between resource assignments.

In what follows, we derive $\frac{\nabla p(\mathbf{s}, t_n|\theta, \lambda)}{p(\mathbf{s}, t_n|\theta, \lambda)}$. We will need a helper lemma that characterizes the gradient of target coverage probabilities with respect to θ. In order to give the result, we will need some definitions. Given a joint assignment of resources to schedules $\mathbf{s}$, let $c_{-k,n}(\mathbf{s})$ indicate whether target n is covered by some resource other than resource k. In other words, $c_{-k,n}(\mathbf{s}) = \mathbf{1}[t_n \in \cup_{k'\neq k}\mathbf{s}_{k'}]$. Also let $c_{k,l,n}$ indicate whether the lth schedule of resource k contains target t_n; $c_{k,l,n} = \mathbf{1}[t_n \in s_{k,l}]$.

LEMMA 1. *For any target index n, resource index k, and schedule index $l \geq 1$,*

$$\frac{\partial}{\partial\theta_{k,l}}\bar{\mathbf{c}}_n(\mathcal{D}_\theta) = (c_{k,l,n} - c_{k,0,n})\,\mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(c_{-k,n}(\mathbf{s}) = 0)$$

PROOF. Fix a target n, for any resource k, we have:

$$\begin{aligned}\bar{\mathbf{c}}_n(\mathcal{D}_\theta) &= \mathbb{E}_{s\sim\mathcal{D}_\theta}[\mathbf{c}_n(\mathbf{s})] = \mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(\mathbf{c}_n(\mathbf{s}) = 1)\\ &= \mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(\mathbf{c}_n(\mathbf{s}) = 1 \mid c_{-k,n}(\mathbf{s}) = 0)\mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(c_{-k,n}(\mathbf{s}) = 0)\\ &+ \mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(\mathbf{c}_n(\mathbf{s}) = 1 \mid c_{-k,n}(\mathbf{s}) = 1)\mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(c_{-k,n}(\mathbf{s}) = 1)\\ &= \mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(\mathbf{c}_n(\mathbf{s}) = 1 \mid c_{-k,n}(\mathbf{s}) = 0)\mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(c_{-k,n}(\mathbf{s}) = 0)\\ &+ \mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(c_{-k,n}(\mathbf{s}) = 1)\end{aligned}$$

For the same resource k and any schedule l, observe that $\frac{\partial}{\partial\theta_{k,l}}\bar{\mathbf{c}}_n(\mathcal{D}_\theta)$ is therefore equal to:

$$\begin{aligned}&\frac{\partial}{\partial\theta_{k,l}}\mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(\mathbf{c}_n(\mathbf{s}) = 1 \mid c_{-k,n}(\mathbf{s}) = 0)\mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(c_{-k,n}(\mathbf{s}) = 0)\\ &= \mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(c_{-k,n}(\mathbf{s}) = 0)\frac{\partial}{\partial\theta_{k,l}}\sum_{l'=0}^{d_k} c_{k,l',n}\theta_{k,l'}\\ &= (c_{k,l,n} - c_{k,0,n})\,\mathbb{P}_{\mathbf{s}\sim\mathcal{D}_\theta}(c_{-k,n}(\mathbf{s}) = 0).\end{aligned}$$

□

The ratio $\frac{\nabla p(\mathbf{s},t_n|\theta,\lambda)}{p(\mathbf{s},t_n|\theta,\lambda)}$ can be written in terms of the derivative in Lemma 1. Let $\mathbf{s} = [s_{1,l_1}, \ldots, s_{K,l_K}]$, so that the indices of the selected schedules are given by $l_1, \ldots, l_K$. Define $z(\mathbf{s}, k, l) \in \{-1, 0, 1\}$ by letting $z(\mathbf{s}, k, l) = \mathbf{1}[l_k = k] - \mathbf{1}[l_k = 0]$. In other words $z(\mathbf{s}, k, l)$ provides a sign of 1 if resource k is assigned to schedule l, a sign of -1 if it is assigned to schedule 0 and a sign of 0 otherwise.

THEOREM 1. *For any target t_n, schedule assignment $\mathbf{s}$, as well as resource index k and schedule index $l \geq 1$, we have that:*

$$\frac{\frac{\partial}{\theta_{k,l}} p(\mathbf{s}, t_n \mid \theta)}{p(\mathbf{s}, t_n \mid \theta)} = \frac{z(\mathbf{s}, k, l)}{\theta_{k,l_k}} - \eta \Delta_{a,n} \frac{\partial}{\partial \theta_{k,l}} \bar{c}_n(\mathcal{D}_\theta) + \eta \sum_{n'=1}^{N} \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n') \Delta_{a,n'} \frac{\partial}{\partial \theta_{k,l}} \bar{c}_{n'}(\mathcal{D}_\theta)$$

PROOF. Fix $\mathbf{s} = [s_{1,l_1}, \ldots, s_{K,l_K}]$. We have that $p(\mathbf{s} \mid \theta) = \prod_{k'=1}^{K} \theta_{k',l_{k'}}$. Now fix a resource index k and schedule index $l \geq 1$. We have that $\frac{\partial}{\partial \theta_{k,l}} p(\mathbf{s} \mid \theta) = \prod_{k' \neq k} \theta_{k',l_{k'}}$ if $l_k = l$; $\frac{\partial}{\partial \theta_{k,l}} p(\mathbf{s} \mid \theta) = -\prod_{k' \neq k} \theta_{k',l_{k'}}$ if $l_k = 0$; and $\frac{\partial}{\partial \theta_{k,l}} p(\mathbf{s} \mid \theta) = 0$ otherwise. From the definition of z:

$$\frac{\frac{\partial}{\partial \theta_{k,l}} p(\mathbf{s} \mid \theta)}{p(\mathbf{s} \mid \theta)} = \frac{z(\mathbf{s}, k, l)}{\theta_{k,l_k}} \tag{4}$$

Next fix a target index n, and consider $\frac{\partial}{\partial \theta_{k,l}} \exp(\eta \mathbf{u}_n(\mathcal{D}_\theta)))$.

$$\begin{aligned}
\frac{\partial}{\partial \theta_{k,l}} \exp(\eta \mathbf{u}_n(\mathcal{D}_\theta)) &= \exp(\eta \mathbf{u}_n(\mathcal{D}_\theta)) \frac{\partial}{\partial \theta_{k,l}} \eta \mathbf{u}_n(\mathcal{D}_\theta) \\
&= \eta \exp(\eta \mathbf{u}_n(\mathcal{D}_\theta)) \frac{\partial}{\partial \theta_{k,l}} U_a(\mathcal{D}_\theta, t_n) \\
&= -\eta \exp(\eta \mathbf{u}_n(\mathcal{D}_\theta)) \Delta_{a,n} \frac{\partial}{\partial \theta_{k,l}} \bar{c}_n(\mathcal{D}_\theta)
\end{aligned}$$

This lets us differentiate $\sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n)$:

$$\begin{aligned}
\frac{\partial}{\partial \theta_{k,l}} \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n) &= \frac{\partial}{\partial \theta_{k,l}} \frac{\exp(\eta \mathbf{u}_n(\mathcal{D}_\theta)))}{\sum_{n'=1}^{N} \exp(\eta \mathbf{u}_{n'}(\mathcal{D}_\theta)))} \\
&= \frac{1}{\sum_{n'=1}^{N} \exp(\eta \mathbf{u}_{n'}(\mathcal{D}_\theta))} \frac{\partial}{\partial \theta_{k,l}} \exp(\eta \mathbf{u}_{n'}(\mathcal{D}_\theta)) \\
&+ \exp(\eta \mathbf{u}_n(\mathcal{D}_\theta)) \frac{\partial}{\partial \theta_{k,l}} \frac{1}{\sum_{n'=1}^{N} \exp(\eta \mathbf{u}_{n'}(\mathcal{D}_\theta))} \\
&= -\eta \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n) \Delta_{a,n} \frac{\partial}{\partial \theta_{k,l}} \bar{c}_n(\mathcal{D}_\theta) \\
&- \frac{\sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n)}{\sum_{n'=1}^{N} \exp(\eta \mathbf{u}_{n'}(\mathcal{D}_\theta))} \frac{\partial}{\partial \theta_{k,l}} \sum_{n'=1}^{N} \exp(\eta \mathbf{u}_{n'}(\mathcal{D}_\theta)) \\
&= -\eta \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n) \Delta_{a,n} \frac{\partial}{\partial \theta_{k,l}} \bar{c}_n(\mathcal{D}_\theta) \\
&+ \eta \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n) \sum_{n'=1}^{N} \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n') \Delta_{a,n'} \frac{\partial}{\partial \theta_{k,l}} \bar{c}_{n'}(\mathcal{D}_\theta)
\end{aligned} \tag{5}$$

Using equations (4) and (5), we have:

$$\begin{aligned}
\frac{\frac{\partial}{\partial \theta_{k,l}} p(\mathbf{s}, t_n \mid \theta)}{p(\mathbf{s}, t_n \mid \theta)} &= \frac{\sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n) \frac{\partial}{\partial \theta_{k,l}} p(\mathbf{s} \mid \theta)}{p(\mathbf{s} \mid \theta) \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n)} + \frac{p(\mathbf{s} \mid \theta) \frac{\partial}{\partial \theta_{k,l}} \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n)}{p(\mathbf{s} \mid \theta) \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n)} \\
&= \frac{z(\mathbf{s}, k, l)}{\theta_{k,l_k}} - \eta \Delta_{a,n} \frac{\partial}{\partial \theta_{k,l}} \bar{c}_n(\mathcal{D}_\theta) + \eta \sum_{n'=1}^{N} \sigma_\eta(\mathbf{u}(\mathcal{D}_\theta), n') \Delta_{a,n'} \frac{\partial}{\partial \theta_{k,l}} \bar{c}_{n'}(\mathcal{D}_\theta)
\end{aligned}$$

□

6. EXPERIMENTS: SCALING WITH ATTACKER TYPES

The derivation from the previous section allows us to implement the STACKGRAD algorithm for a particular class of defender mixed strategies, specifically those that assign resources to schedules independently. We call the resulting algorithm STACKGRAD-I (where I follows from the independence assumption).

We now demonstrate that STACKGRAD-I scales very well on games with a large numbers of attacker types. Specifically, we will see that for a fixed choice of parameters T and m, both the runtime and quality of the solution θ_T found by STACKGRAD-I are *constant* in the number of attacker types.

Experiments were conducted on the patrolling domain introduced by Paruchuri et al. [2007], (see Paruchuri et al. [2008]). The goal is to assign a single security resource (such as a UAV or robot) to a sequence of targets $[t_{i_1}, \ldots, t_{i_d}]$, called its patrol. Thus, for patrol length d, the set of defender pure strategies $\mathcal{S}$ consists of all length d permutations of the target set.

In the original game, the coverage induced by strategy $\mathbf{s} = [t_{n_1}, \ldots, t_{n_d}]$ is not binary, and depends on the location of a target t_n in $\mathbf{s}$. That is, there are parameters $p_1, \ldots, p_d$ where $\mathbf{c}_n(\mathbf{s}) = p_j$ if $n = n_j$ and $\mathbf{c}_n(\mathbf{s}) = 0$ if $n \neq \{n_1, \ldots, n_d\}$. Target t_n is covered with probability p_j if it appears jth in the patrol, otherwise it is not covered at all.

We use this formulation, but note that for this (non-binary) coverage function Lemma 1 must re-derived. Using the categorical distribution, there is a single parameter θ_l for each permutation in $\mathcal{S}$, and it is not difficult to show that $\frac{\partial}{\partial \theta_l} \bar{\mathbf{c}}_n(\mathcal{D}_\theta) = \mathbf{c}_n(\mathbf{s}_l) - \mathbf{c}_n(\mathbf{s}_0)$. The result of Theorem 1 is unchanged. Secondly, we note that for this particular game, the categorical distribution completely characterizes the set of mixed defender strategies ($\Delta = \Delta_\Theta$). Thus, for large enough inverse temperature in the softmax function ($\eta \to \infty$), maximizing $\tilde{U}_d(\theta)$ is equivalent to finding a Stackelberg equilibrium defender strategy.

Experiments where run on randomly generated instances of these games; $U_d(0, t_n)$, $\Delta_{d,n}$, $U_\lambda(0, t_n)$ and $\Delta_{\lambda,n}$ are chosen at random. In all instances STACKGRAD-I is run with inverse temperature $\eta = 20$, and sampling parameter $m = 50$. Every data point is the average of 20 experiments. Experiments were run on a 2.5 GHz Intel Xeon E5-2680v3 processor, and Integer programs were solved using IBM CPLEX 12.4.

In Figure 2, we compare the running time of STACKGRAD-I and the DOBBS algorithm of Paruchuri et al. [2008], which is the fastest algorithm for general Stackelberg games, but must solve an integer program. Experiments were conducted on instances with 5 targets and patrol lengths of 2, for a total of 40 defender pure strategies. While the game is small enough to be tractable with a small number of defender pure strategies, as the number of attacker types is increased, the DOBBS algorithm begins to experience an exponential increase in its running time. In contrast, STACKGAD-I exhibits near constant average running time, with a range between 38 seconds and 59 seconds (against 55 attackers and 45 attackers respectively). Furthermore, scaling the game has no effect on the quality of the solution found by STACKGRAD-I. The competitive ratio of the solution θ^* found by STACKGRAD-I against the true

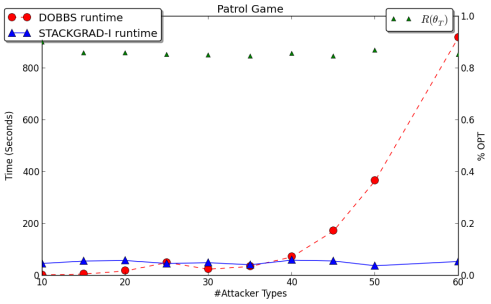

Figure 2: STACKGRAD-I vs. DOBBS runtime.

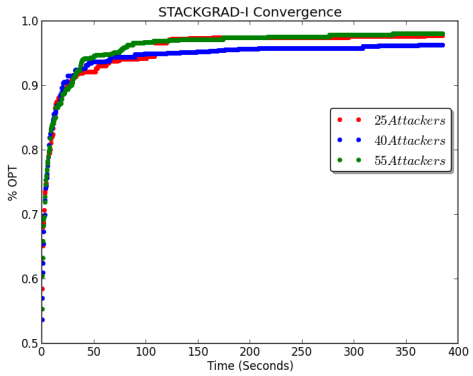

Figure 3: Competitive Ratio vs Time

optimum found by DOBBS ranges between 84.5% and 86.8%.

Of course both the running time and the quality of the solution found by STACKGRAD-I is a function of T, the number of iterations the gradient ascent is run. For the results in Figure 2, STACKGRAD-I was terminated after $T = 1000$ iterations.

In Figure 3, we show the results of running STACKGRAD-I for up to $T = 5000$ iterations, for $25, 40, 55$ attacker types. Here we see that the competitive ratio of the best solution found by STACKGRAD-I against the Stackelberg optimum quickly (displayed on the y axis) converges to 1. After 100 seconds (corresponding to about $T = 2500$), the average competitive ratio has reached above 95%.

STACKGRAD's constant running time, and performance, as the number of attackers is scaled up (but T is held constant), can be attributed to the fact that neither the parameterization nor the computation of the gradient depend on the size of Λ. Furthermore, the parameterization is complete, in the sense that all mixed strategies for the game are representable. The number of pure strategies, however, was kept small; with five targets and patrol of length two, there are a total of 20 defender pure strategies. In the next section, we demonstrate how STACKGRAD can be deployed to find solutions in games where the set of pure strategies is very large.

7. LARGE STRATEGY SETS

7.1 Structured Pure Strategies

While the categorical distribution model of Section 5 demonstrates the power of gradient methods when the number of pure strategies are small, and the number of attacker types is large, most security games are concerned with settings in which there is a very large space of defender pure strategies. Recall that under the categorical model $\Theta \subset \mathbb{R}^d$, where $d = \sum_{k=1}^{K}(|\mathcal{S}_k| - 1)$. Thus, if $|\mathcal{S}_k|$ tends to be large, we would not expect STACKGRAD-I to perform very well. This is often the case in real security domains where the set of pure strategies exhibits a combinatorial explosion with the number of defender resources or potential targets.

In this Section, we introduce a new class of defender mixed strategies Θ which has a compact representation even as $|S_k|$ grows large. We consider a setting in which schedules can be iteratively constructed. In the FAMS domain, for example, an air marshal r_k is assigned a schedule consisting of a sequence of flights $\mathbf{s}_k = [t_{n_1}, \ldots, t_{n_L}]$ (which we take to be ordered). The length of the schedule is bounded (there is some upper bound $L \leq B$), and must land the marshal back to its origin airport. Although the set of feasible schedules for a marshal r_k can be an exponentially large in B, it has a natural combinatorial structure. In particular, given a subsequence of flights $\mathbf{s}_{k,1:l} = [t_{n_1}, \ldots, t_{n_l}]$, $l < L$, the set of feasible "next flights" can be efficiently computed. Specifically, the subsequence of flights specified by $\mathbf{s}_{k,1:l}$ lands the marshal at some airport A at some time τ. The viable choices for the $l+1$ flight are those flights leaving airport A after time τ, that can get the marshal back home in $B - l$ hops or fewer.[2]

More generally, we consider games where the schedules for some resource r_k is given by an ordered set of targets $\mathbf{s}_k$. We will further assume that given some subseqence $\mathbf{s}_{k,1:l}$, the set of feasible next targets $F(\mathbf{s}_{k,1:l})$ can be efficiently computed. Formally, t_n belongs to $F(\mathbf{s}_{k,1:l})$ if and only if there is some schedule $\mathbf{s}_k \in \mathcal{S}_k$, where $\mathbf{s}_k = [\mathbf{s}_{k,1:l}, t_n, \ldots]$ ($\mathbf{s}_k$ begins with the prefix $\mathbf{s}_{k,1:l}$).

7.2 Parametric Model

Following the recipe outlined in Section 4, we introduce a new parametric model Θ for games that exhibit the sequential structure just described.

We will consider a simple logistic model. For each resource k we introduce a vector $\mathbf{w}_k$ of dimension N, where N is the number of targets. We denote the nth element of $\mathbf{w}_k$ by $\mathbf{w}_{k,n}$. A resource r_k is assigned to schedule $\mathbf{s}_k$ according to a sequential stochastic process. Given a subsequence of targets of length l, $\mathbf{s}_{k,1:l}$, the next target in resource r_k's schedule is selected with probability proportional to $\exp(\mathbf{w}_{k,n})$ when $t_n \in F(\mathbf{s}_{k,1:l})$ and with probability 0 otherwise. For $t_n \in F(\mathbf{s}_{k,1:l})$:

$$\mathbb{P}[\mathbf{s}_k(l+1) = t_n \mid \mathbf{s}_{k,1:l}] = \frac{\exp(\mathbf{w}_{k,n})}{\sum_{t_{n'} \in F(\mathbf{s}_{k,1:l})} \exp(\mathbf{w}_{k,n'})} \quad (6)$$

Thus, $\mathbf{w}_{k,n}$ indicates the propensity for target t_n to be covered whenever that target is available in the feasible set of next targets. As was the case with the model of Section 5, each resource will be independently assigned to a schedule. Therefore $\theta = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$, and $\Delta(\Theta)$ indicates the set of distributions over resource assignments where resources are independently assigned to schedules according to the aforementioned stochastic process. Notice that the dimension of θ is KN, where K is the number of resources and N is the number of targets. Therefore, the parameterization remains compact even if the size of $|\mathcal{S}|$ explodes.

We state $\frac{\nabla p(\mathbf{s}, t_n|\theta)}{p(\mathbf{s}, t_n|\theta)}$ for this new parametric class of strategies in the following Theorem.

THEOREM 2. *Let $t_{n_{k,l}}$ denote the lth target covered by the kth resource chosen by the defender. For any target t_{n^*} chosen by the*

[2]This computation is via a modification of Djikstra's algorithm.

attacker, schedule assignments $\mathbf{s}$ *chosen by defender, and any parameter* $\mathbf{w}_{k,n}$*:*

$$\frac{\frac{\partial}{\partial \mathbf{w}_{k,n}} p(\mathbf{s}, t_{n^*} \mid \theta)}{p(\mathbf{s}, t_{n^*} \mid \theta)} = \sum_{l=1}^{L} \mathbf{1}[n = n_{k,l}] - \mathbb{P}[\mathbf{s}_k(l) = t_n \mid \mathbf{s}_{k,1:(l-1)}, \theta] - \eta \sum_{n'=1}^{N} \sigma_\eta(\mathbf{u}_\lambda(\mathcal{D}_\theta), n') \Delta_{a,n'} \frac{\partial}{\partial \mathbf{w}_{k,n}} \bar{\mathbf{c}}_{n'}(\mathcal{D}_\theta) + \eta \Delta_{a,n^*} \frac{\partial}{\partial \mathbf{w}_{k,n}} \bar{\mathbf{c}}_{n^*}(\mathcal{D}_\theta)$$

PROOF. Given a parameterization θ, denote the probability that agent r_k is assigned to some schedule $\mathbf{s}_k = [t_{n_{k,1}}, \ldots, t_{n_{k,L}}]$ by $p(\mathbf{s}_k \mid \theta)$. Fixing a parameter $\mathbf{w}_{k,n}$, and subsequence of length $l-1$, we can apply equation 6, to conclude that:

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{w}_{k,n}} \mathbb{P}[\mathbf{s}_k(l) = t_{n_{k,l}} \mid \mathbf{s}_{k,1:(l-1)}, \theta] \\
&= \frac{\partial}{\partial \mathbf{w}_{k,n}} \frac{\exp(\mathbf{w}_{k,n_{k,l}})}{\sum_{t_{n'} \in F(\mathbf{s}_{k,1:(l-1)})} \exp(\mathbf{w}_{k,n'})} \\
&= \mathbf{1}[n = n_{k,l}] \frac{\exp(\mathbf{w}_{k,n_{k,l}})}{\sum_{t_{n'} \in F(\mathbf{s}_{k,1:(l-1)})} \exp(\mathbf{w}_{k,n'})} \\
&- \frac{\exp(\mathbf{w}_{k,n_{k,l}}) \frac{\partial}{\partial \mathbf{w}_{k,n}} \sum_{t_{n'} \in F(\mathbf{s}_{k,1:(l-1)})} \exp(\mathbf{w}_{k,n'})}{\left(\sum_{t_{n'} \in F(\mathbf{s}_{k,1:(l-1)})} \exp(\mathbf{w}_{k,n'})\right)^2} \\
&= \mathbf{1}[n = n_{k,l}] \mathbb{P}[\mathbf{s}_k(i) = t_{n_{k,l}} \mid \mathbf{s}_{k,1:(l-1)}, \theta] \\
&- \frac{\exp(\mathbf{w}_{k,n_{k,l}}) \mathbf{1}[t_n \in F(\mathbf{s}_{k,1:(l-1)})] \exp(\mathbf{w}_{k,n})}{\left(\sum_{t_{n'} \in F(\mathbf{s}_{k,1:(l-1)})} \exp(\mathbf{w}_{k,n'})\right)^2} \\
&= \mathbb{P}[\mathbf{s}_k(l) = t_{n_{k,l}} \mid \mathbf{s}_{k,1:(l-1)}, \theta] \mathbf{1}[n = n_{k,l}] \\
&- \mathbb{P}[\mathbf{s}_k(l) = t_{n_{k,l}} \mid \mathbf{s}_{k,1:(l-1)}, \theta] \mathbb{P}[\mathbf{s}_k(l) = t_n \mid \mathbf{s}_{k,1:(l-1)}, \theta]
\end{aligned}$$

Equation 6 also tells us that:

$$p(\mathbf{s}_k \mid \theta) = \prod_{l=1}^{L} \mathbb{P}[\mathbf{s}_k(l) = t_{n_{k,l}} \mid \mathbf{s}_{k,1:(l-1)}, \theta]$$

Applying the chain rule gives us:

$$\begin{aligned}
\frac{\frac{\partial}{\partial \mathbf{w}_{k,n}} p(\mathbf{s}_k \mid \theta)}{p(\mathbf{s}_k \mid \theta)} &= \sum_{l=1}^{L} \frac{\frac{\partial}{\partial \mathbf{w}_{k,n}} \mathbb{P}[\mathbf{s}_k(l) = t_{n_{k,l}} \mid \mathbf{s}_{k,1:(l-1)}, \theta]}{\mathbb{P}[\mathbf{s}_k(l) = t_{n_{k,l}} \mid \mathbf{s}_{k,1:(l-1)}, \theta]} \\
&= \sum_{l=1}^{L} \mathbf{1}[n = n_{k,l}] - \mathbb{P}[\mathbf{s}_k(l) = t_n \mid \mathbf{s}_{k,1:(l-1)}, \theta]
\end{aligned}$$

Borrowing equation 5 from the proof of Theorem 1, concludes the proof. □

Unlike the categorical model of Section 5, the partial derivative of $\bar{\mathbf{c}}_n(\mathcal{D}_\theta)$ cannot be easily computed in closed form. Changing the value of parameter $\mathbf{w}_{k,n}$, might effect the coverage of targets $t_{n'} \neq t_n$. In the federal air marshal (FAMS) domain, $\mathbf{w}_{k,n}$ corresponds to the likelihood that marshal r_k takes flight t_n. Increasing $\mathbf{w}_{k,n}$, however, also changes the coverage probabilities of other flights; flights departing from the destination airport of t_n also become more likely, as the marshal needs to return home.

Therefore, instead of computing the derivative in closed form, we use numerical differentiation. Given a set of parameters $\theta = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$, let $\theta + \delta_{n,k}$ denote the set of parameters, where $\mathbf{w}_{k,n} = \mathbf{w}_{k,n} + \delta$, and all other parameters are unchanged. We estimate $\frac{\partial}{\partial \mathbf{w}_{k,n}} \bar{\mathbf{c}}_n(\theta)$ by sampling schedules $\mathbf{s}^{(i)}$ according to θ and schedules $\mathbf{s}^{(i)}_{n,k}$ according to $\theta + \delta^{(k)}_n$ for $i = 1, \ldots, m$. We then estimate $\frac{\partial}{\partial \mathbf{w}_{k,n}} \bar{c}_n(\theta)$ using $\frac{1}{m} \sum_{i=1}^{m} \frac{\mathbf{c}_n(\mathbf{s}^{(i)}) - \mathbf{c}_n(\mathbf{s}^{(i)}_{k,n})}{\delta}$.

8. EXPERIMENTS: SCALING WITH LARGE STRATEGY SETS

The derivation of the gradient for the model in the previous section gives us another instantiation of STACKGRAD, which we call STACKGRAD-L (where L follows from the logistic model). We now demonstrate the scalability of STACKGRAD-L in domains with a large number of defender pure strategies.

Recall that the STACKGRAD-L was derived to have no dependency on $|\mathcal{S}_k|$, the number of schedules available for any resource k. We compare against the state-of-the-art ASPEN algorithm [Jain et al., 2010a]. We note that the ASPEN algorithm was designed to eliminate the combinatorial explosion in *joint* schedules. If all $|\mathcal{S}_k| = X$, then there are X^K possible joint schedules if the Stackelberg game were to be expressed in normal form. Nevertheless, ASPEN still has a computational dependency the size of an individual set of schedules $\mathcal{S}_k$. In particular, ASPEN uses a column generation technique, where selecting each new column requires solving a network flow problem with at least $\sum_k^K |\mathcal{S}_k|$ edges.

We conduct experiments on games inspired by the FAMS domain. We begin with a fixed weighted network G^*, representing actual air travel between the 500 US airports with the most traffic [Colizza et al., 2007]. Nodes represent airports and the weights in G^* represent how many tickets were available between two airports in a given year. Given a parameter N, we generate a game instance as follows. We create a new network G, where G is generated by sampling 30 airports from G^* where at least 5 of these airports are chosen to be "hub" airports. We then create N edges in G corresponding to flights, where a flight between aiports v_1 and v_2 is present in G with probability proportional the weight on (v_1, v_2) in G^*. The edges in G are meant to be representative of actual flights between 30 US airports on a given day. Each edge represents a flight, and therefore a target. In this game, we have only a single attacker type. In all our experiments there are 3 marshals, who can take a tour of length $L \leq 5$ before returning to their home city. The home city is selected uniformly from one of the hubs for each of the marshals. We allow any such tour on G. [3]

We conduct experiments for various values of $N \in \{500, 1000, 1500, 2000, 2500, 5000, 10000\}$. $\mathcal{S}_k$ for a single marshal k consists of all tours from a start vertex of length at most 5 in a network consisting of up to 10000 directed edges. Even simply enumerating $\mathcal{S}_k$ for our larger instances is impossible. However, running ASPEN requires $\{\mathcal{S}_k\}$ to be given as input. Therefore, in order to allow ASPEN to complete in a reasonable amount of time, we generate subsets $\hat{\mathcal{S}}_k \subset \mathcal{S}_k$ by randomly sampling $Y = 10,000$ tours for each marshal. As a result, the performance of STACKGRAD-L is not reported as a competitive ratio to the true Stackelberg optimum, since ASPEN is also solving a restricted game.

Nevertheless, we can compare the performance of STACKGRAD-

[3]This permissiveness is somewhat artificial; in reality, a flight lands at its destination at a specified time, eliminating flights that leave sooner than that from consideration at the destination airport. This realism can be accounted for in the definition of F in the previous section. To keep the experiments simple, we allow any tour on G, and comment that this does not affect the scalability of STACKGRAD-L.

L to that of ASPEN as the size of the game is increased. In Figure 4 we compare the runtime of ASPEN with STACKGRAD-L for various values of N. Once Y is fixed, the size of the pure strategy set of ASPEN is constant. ASPEN is unaware of any of the pure strategies outside the set $\hat{\mathcal{S}}_1 \times \hat{\mathcal{S}}_2 \times \hat{\mathcal{S}}_3$. Nevertheless ASPEN must still solve a mixed integer linear program where the number of constraints scale with N. The average runtime of STACKGRAD-L increases as N grows as well, as the parameterization of STACKGRAD-L is in $\mathbb{R}^N$, but remains bounded by that of ASPEN for large N.

In Figure 4 we also display the average performance of each algorithm. Performance is measured as the direct payoff to the defender in the unrestricted game, in contrast to the results in Section 6, where we computed competitive ratios directly. We see that for smaller number of targets ASPEN finds a higher quality solution than STACKGRAD-L on average. However, as N is increased the solution found by STACKGRAD-L overtakes. We suspect this is due to the fact that fixing Y in ASPEN eliminates pure strategies that might be necessary for the defender, while any pure strategy can be represented by STACKGRAD-L. For large N, $\hat{\mathcal{S}}_k$ is a very small subset of all the possible tours of length 5.

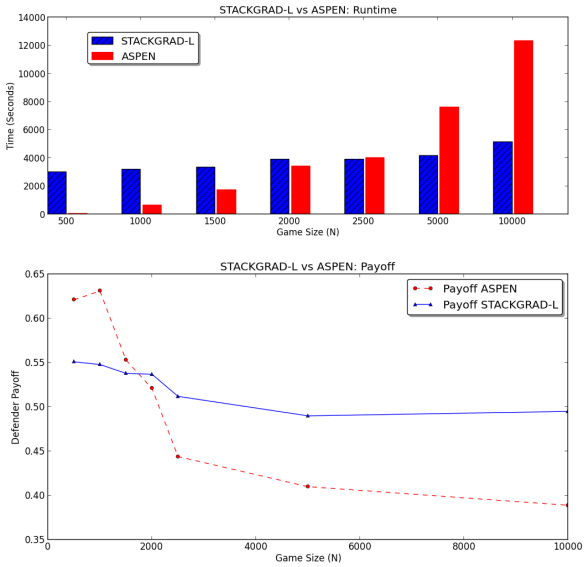

Figure 4: STACKGRAD-L vs ASPEN

9. CONCLUSIONS AND FUTURE WORK

We present a new algorithm, STACKGRAD, for solving Stackelberg security games, which works by performing a gradient ascent, rather than solving an integer program. We demonstrate a version of STACKGRAD with no computational dependence on the number of attacker types, and another version with a mild dependence on the number of defender pure strategies. Since the procedure restricts its search to within a parametric class of defender mixed strategies, and might find strategies bounded away from the true Stackelberg optimum, we also provide empirical evidence that the solutions found by STACKGRAD are of comparable quality to solutions in the unrestricted game. These empirical successes invite open questions for future research, chief among these being whether there are parametric classes of defender mixed strategies which are provably competitive with the true Stackelberg optimum.

References

Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, pages 319–350, 2001.

Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005, 2003.

Vittoria Colizza, Romualdo Pastor-Satorras, and Alessandro Vespignani. Reaction–diffusion processes and metapopulation models in heterogeneous networks. *Nature Physics*, 3(4):276–282, 2007.

Manish Jain, Erem Kardeş, and Fernando Ordóñez. Security games with arbitrary schedules: A branch and price approach. In *24th AAAI Conference on Artificial Intelligence*, pages 792–797, 2010a.

Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóñez. Software assistants for randomized patrol planning for the LAX airport police and the federal air marshal service. *Interfaces*, 40 (4):267–290, 2010b.

Manish Jain, Christopher Kiekintveld, and Milind Tambe. Quality-bounded solutions for finite Bayesian Stackelberg games: Scaling up. In *10th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 997–1004, 2011.

Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *8th International Conference on Autonomous Agents and Multiagent Systems*, pages 689–696, 2009.

Christopher Kiekintveld, Janusz Marecki, and Milind Tambe. Approximation methods for infinite bayesian stackelberg games: Modeling distributional payoff uncertainty. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1005–1012. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

Praveen Paruchuri, Jonathan P Pearce, Milind Tambe, Fernando Ordonez, and Sarit Kraus. An efficient heuristic approach for security against multiple adversaries. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 181. ACM, 2007.

Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 895–902, 2008.

James Pita, Harish Bellamane, Manish Jain, Chris Kiekintveld, Jason Tsai, Fernando Ordóñez, and Milind Tambe. Security applications: Lessons of real-world deployment. *ACM SIGecom Exchanges*, 8(2):5, 2009.

Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordóñez, and Milind Tambe. IRIS: A tool for strategic security allocation in transportation networks. In *8th International Conference on Autonomous Agents and Multiagent Systems (Industrial Track)*, pages 37–44, 2009.

Rong Yang, Fernando Ordonez, and Milind Tambe. Computing optimal strategy against quantal response in security games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 847–854. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

Zhengyu Yin and Milind Tambe. A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 855–862. International Foundation for Autonomous Agents and Multiagent Systems, 2012.