Approximation Methods for Infinite Bayesian Stackelberg Games: Modeling Distributional Payoff Uncertainty

Christopher Kiekintveld
University of Texas at El Paso
Dept. of Computer Science
cdkiekintveld@utep.edu

Janusz Marecki
IBM Watson Research Lab
New York, NY
janusz.marecki@gmail.com

Milind Tambe
University of Southern California
Dept. of Computer Science
tambe@usc.edu

ABSTRACT

Game theory is fast becoming a vital tool for reasoning about complex real-world security problems, including critical infrastructure protection. The game models for these applications are constructed using expert analysis and historical data to estimate the values of key parameters, including the preferences and capabilities of terrorists. In many cases, it would be natural to represent uncertainty over these parameters using continuous distributions (such as uniform intervals or Gaussians). However, existing solution algorithms are limited to considering a small, finite number of possible attacker types with different payoffs. We introduce a general model of infinite Bayesian Stackelberg security games that allows payoffs to be represented using continuous payoff distributions. We then develop several techniques for finding approximate solutions for this class of games, and show empirically that our methods offer dramatic improvements over the current state of the art, providing new ways to improve the robustness of security game models.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Algorithms,Economics,Experimentation

Keywords

Game theory, Bayesian Stackelberg games, robustness, security, uncertainty, risk analysis

1. INTRODUCTION

Stackelberg games are increasingly important for informing real-world decision-making, including a growing body of work that applies these techniques in security domains such as critical infrastructure protection [22, 6], computer networks [3, 17], and robot patrolling strategies [10, 2, 5]. Two software systems that use this type of game modeling are in use by the the Los Angeles International Airport (LAX) [20] and the Federal Air Marshals Service (FAMS) [24] to assist with resource allocation decision. A key issue that has arisen in these applications is whether the models can accurately represent the uncertainty that domains experts have about the inputs used to construct the game models, including the preferences and capabilities of terrorist adversaries.

To apply game-theoretic reasoning, the first step in the analysis is to construct a precise game model. The typical approach (e.g., in the LAX and FAMS applications) is to construct a model using a combination of the available data and expert opinions. Unfortunately, the data is often limited or imprecise, especially in regards to information about the terrorist adversaries. For example, it can be difficult to predict precisely how attackers will weigh casualties, economic consequences, media exposure, and other factors when selecting targets. Our focus in this paper is on developing techniques to more accurately model the uncertainty about the parameters of the model to avoid poor decisions due to overconfidence.

Bayesian games [11] are the most common framework for reasoning about uncertainty in game-theoretic settings. Unfortunately, it is known that finding equilibria of finite Bayesian Stackelberg games is NP-hard [9]. The DOBSS algorithm [18] used in the ARMOR system at LAX is able to solve games with roughly 10 attacker types and up to 5 actions for each player. Until very recently with the development of HBGS [12], this was the fastest known algorithm for finite Bayesian Stackelberg games. Both DOBSS and HBGS are too slow to scale to domains such as FAMS with thousands of actions, and we show in our experimental results that restricting the model to a small number of attacker types generally leads to poor solution quality.

In this work we introduce a general model of infinite Bayesian Stackelberg security games that allows payoffs to be represented using continuous payoff distributions (e.g., Gaussian or uniform distributions). This model allows for a richer and more natural expression of uncertainty about the input parameters, leading to higher-quality and more robust solutions than finite Bayesian models. Our analysis of the model shows that finding exact analytic solutions is infeasible (and efficient algorithms are unlikely in any case, given the complexity results for the finite case). We focus instead on developing approximate solution methods that employ numerical methods, Monte-Carlo sampling, and approximate optimization. Our experiments show that even approximate solutions for the infinite case offer dramatic benefits in both solution quality and scalability over the existing approaches based on perfect information or small numbers of attacker types.

2. RELATED WORK

Stackelberg games have important applications in security domains. These include fielded applications at the Los Angeles International Airport [20] and the Federal Air Marshals Service [24], work on patrolling strategies for robots and unmanned vehicles [10, 2, 5], applications of game theory in network security [3, 26, 17],

Cite as: Approximation Methods for Infinite Bayesian Stackelberg Games: Modeling Distributional Payoff Uncertainty, Christopher Kiekintveld, Janusz Marecki and Milind Tambe, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 1005-1012.

and research that provides policy recommendations for allocation of security resources at a national level [22, 6]. Bayesian games [11] are a standard approach for modeling uncertainty, and there are many specific examples of infinite Bayesian games that have been solved analytically, including many types of auctions [14].

However, there is relatively little work on general algorithms for solving large and infinite Bayesian games. Recent interest in this class of games focuses on developing approximation algorithms [21, 4, 8]. Monte-Carlo sampling approaches similar to those we describe have been applied to some kinds of auctions [7]. In addition, the literature on stochastic choice [15, 16] studies problems that are simplified versions of the choice problem attackers face in our model. Closed-form solutions exist only for special cases with specific types of uncertainty, even in the single-agent stochastic choice literature. A alternative to Bayesian games that has been developed recently is robust equilibrium [1], which takes a worst-case approach inspired by the robust optimization literature.

3. BAYESIAN SECURITY GAMES

We define a new class of infinite Bayesian Security Games, extending the model in Kiekintveld et. al. [13] to include uncertainty about the attacker's payoffs. The key difference between our model and existing approaches (such as in Paruchuri et. al [18]) is that we allow the defender to have a continuous distribution over the possible payoffs of the attacker. Previous models have restricted this uncertainty to a small, finite number of possible attacker types, limiting the kinds of uncertainty that can be modeled.

A security game has two players, a *defender*, Θ, and an *attacker*, Ψ, a set of *targets* $T = \{t_1, \ldots, t_n\}$ that the defender wants to protect (the attacker wants to attack) and a set of *resources* $R = \{r_1, \ldots, r_m\}$ (e.g., police officers) that the defender may deploy to protect the targets. Resources are identical in that any resource can be deployed to protect any target, and any resource provides equivalent protection. A defender's pure strategy, denoted σ_Θ, is a subset of targets from T with size less than or equal to m An attacker's pure strategy, σ_Ψ, is exactly one target from T. Σ_Θ denotes the set of all defender's pure strategies and Σ_Ψ is the set of all attacker's pure strategies. We model the game as a Stackelberg game [25] which unfolds as follows: (1) the defender commits to a mixed strategy δ_Θ that is a probability distribution over the pure strategies from Σ_Θ, (2) nature chooses a random attacker type $\omega \in \Omega$ with probability $Pb(\omega)$, (3) the attacker observes the defender's mixed strategy δ_Θ, and (4) the attacker responds to δ_Θ with a best-response strategy from Σ_Ψ that provides the attacker (of type ω) with the highest *expected* payoff given δ_Θ.

The payoffs for the defender depend on which target is attacked and whether the target is protected (covered) or not. Specifically, for an attack on target t, the defender receives a payoff $U^u_\Theta(t)$ if the target is uncovered, and $U^c_\Theta(t)$ if the target is covered. The payoffs for an attacker of type $\omega \in \Omega$ is $U^u_\Psi(t, \omega)$ for an attack on an uncovered target, and $U^c_\Psi(t, \omega)$ for an attack on a covered target.We assume that both the defender and the attacker know the above payoff structure exactly. However, the defender is uncertain about the attacker's type, and can only estimate the expected payoffs for the attacker. We do not to model uncertainty that the attacker has about the defender's payoffs because we assume that the attacker is able to directly observe the defender's strategy.

3.1 Bayesian Stackelberg Equilibrium

A Bayesian Stackelberg Equilibrium (BSE) for a security game consists of a strategy profile in which every attacker type is playing a best response to the defender strategy, and the defender is playing a best response to the distribution of actions chosen by the attacker types. We first define the equilibrium condition for the attacker and for the defender. We represent the defender's mixed strategy δ_Θ by the compact *coverage vector* $C = (c_t)_{t \in T}$ that gives the probabilities c_t that each target $t \in T$ is covered by at least one resource. Note that $\sum_{t \in T} c_t \leq m$ because the defender has m resources available. In equilibrium each attacker type ω best-responds to the coverage C with a pure strategy $\sigma^*_\Psi(C, \omega)$ given by:

$$\sigma^*_\Psi(C, \omega) = \arg\max_{t \in T}(c_t \cdot U^c_\Psi(t, \omega) + (1 - c_t) \cdot U^u_\Psi(t, \omega)) \quad (1)$$

To define the equilibrium condition for the defender we first define the *attacker response function* $A(C) = (a_t(C))_{t \in T}$ that returns the probabilities $a_t(C)$ that each target $t \in T$ will be attacked, given the distribution of attacker types and a coverage vector C. Specifically:

$$a_t(C) = \int_{\omega \in \Omega} Pb(\omega) \mathbf{1}_t(\sigma^*_\Psi(C, \omega)) d\omega \quad (2)$$

where $\mathbf{1}_t(\sigma^*_\Psi(C, \omega))$ is the indicator function that returns 0 if $t = \sigma^*_\Psi(C, \omega)$ and 0 otherwise. Given the attacker response function $A(\cdot)$ and a set of all possible defender coverage vectors $\mathcal{C}$, the equilibrium condition for the defender is to execute its best-response mixed strategy $\delta^*_\Theta \equiv C^*$ given by:

$$\delta^*_\Theta = \arg\max_{C} \sum_{t \in T} a_t(C)(c_t \cdot U^c_\Theta(t) + (1 - c_t) \cdot U^u_\Theta(t)). \quad (3)$$

3.2 Attacker Payoff Distributions

When the set of attacker types is infinite, calculating the attacker response function directly from Equation (2) is impractical. For this case we instead replace each payoff in the original model with a continuous distribution over possible payoffs. Formally, for each target $t \in T$ we replace values $U^c_\Psi(t, \omega)$, $U^u_\Psi(t, \omega)$ over all $\omega \in \Omega$ with two continuous probability density functions:

$$f^c_\Psi(t, r) = \int_{\omega \in \Omega} Pb(\omega) U^c_\Psi(t, \omega) d\omega \quad (4)$$

$$f^u_\Psi(t, r) = \int_{\omega \in \Omega} Pb(\omega) U^u_\Psi(t, \omega) d\omega \quad (5)$$

that represent the defender's *beliefs* about the attacker payoffs. For example, the defender expects with probability $f^c_\Psi(t, r)$ that the attacker receives payoff r for attacking target t when it is covered. This provides a convenient and general way for domain experts to express uncertainty about payoffs in the game model, whether due to their own beliefs or based on uncertain evidence from intelligence reports. Given this representation, we can now derive an alternative formula for the attacker response function. For some coverage vector C, let $X_t(C)$ be a random variable that describes the *expected* attacker payoffs for attacking target t, given C. It then holds for each target $t \in T$ that:

$$a_t(C) \quad = \quad Pb[X_t(C) > X_{t'}(C) \text{ for all } t' \in T \setminus t] \quad (6)$$

because the attacker acts rationally. Equation 6 can be rewritten as:

$$a_t(C) = \int_{r=-\infty}^{r=+\infty} Pb[X_t(C) = r] \cdot \prod_{t' \in T \setminus t} Pb[X_{t'}(C) < r] dr \quad (7)$$

$$= \int_{r=-\infty}^{r=+\infty} Pb[X_t(C) = r] \cdot \prod_{t' \in T \setminus t} \int_{r'=-\infty}^{r'=r} Pb[X_{t'}(C) = r'] dr' \, dr.$$

Hence, we now show how to determine the random variables $X_t(C)$ used in Equation (7). That is, we provide a derivation of values $Pb[X_t(C) = r]$ for all $t \in T$ and $-\infty < r < +\infty$. To this end, we represent each $X_t(C)$ using two random variables, $X_t^-(C)$ and $X_t^+(C)$. $X_t^-(C)$ describes the expected attacker payoffs for *being caught* when attacking target t while $X_t^+(C)$ describes the expected attacker payoffs for *not being caught* when attacking target t, given coverage vector C. It then holds that $X_t(C) = r$ if $X_t^-(C) = x$ and $X_t^+(C) = r - x$ for some $-\infty < x < +\infty$. (Note, that in a trivial case where $c_t = 1$ it holds that $Pb[X_t^+(C) = 0] = 1$ and consequently $X_t^-(C) = X_t(C)$. Similarly, if $c_t = 0$ then $Pb[X_t^-(C) = 0] = 1$ and $X_t^+(C) = X_t(C)$.) We can hence derive $Pb[X_t(C) = r]$ as follows:

$$Pb[X_t(C) = r] = \int_{x=-\infty}^{x=+\infty} Pb[X_t^-(C) = x] \cdot Pb[X_t^+(C) = r - x]dx$$

$$= \int_{x=-\infty}^{x=+\infty} \frac{Pb[X_t^-(C) = x]dx \cdot Pb[X_t^+(C) = r - x]dx}{dx}$$

$$= \int_{x=-\infty}^{x=+\infty} \frac{Pb[x \le X_t^-(C) \le x + dx] \cdot Pb[r - x \le X_t^+(C) \le r - x + dx]}{dx}$$

If a random event provides payoff $y := \frac{x}{c_t}$ with probability c_t, the expected payoff of that event is $y \cdot c_t = x$. Therefore:

$$= \int_{x=-\infty}^{x=+\infty} \frac{1}{dx} \int_{y=\frac{x}{c_t}}^{y=\frac{(x+dx)}{c_t}} f_\psi^c(t, y)dy \int_{y=\frac{r-x}{1-c_t}}^{y=\frac{r-x+dx}{1-c_t}} f_\psi^u(t, y)dy$$

Substituting $u := c_t y$, $v := (1 - c_t)y$ in the inner integrals we get:

$$= \int_{x=-\infty}^{x=+\infty} \frac{1}{dx} \int_{u=x}^{u=x+dx} f_\psi^c\left(t, \frac{u}{c_t}\right) \frac{1}{c_t} du \int_{v=r-x}^{v=r-x+dx} f_\psi^u\left(t, \frac{v}{1-c_t}\right) \frac{1}{1-c_t} dv$$

$$= \int_{x=-\infty}^{x=+\infty} \frac{1}{dx} f_\psi^c\left(t, \frac{x}{c_t}\right) \frac{1}{c_t} dx \cdot f_\psi^u\left(t, \frac{r-x}{1-c_t}\right) \frac{1}{1-c_t} dx$$

$$= \int_{x=-\infty}^{x=+\infty} \frac{1}{c_t} f_\psi^c\left(t, \frac{x}{c_t}\right) \cdot \frac{1}{1-c_t} f_\phi^u\left(t, \frac{r-x}{1-c_t}\right) dx.$$

Using this derived formula for $Pb[X_t(C) = r]$ in (7) we obtain:

$$a_t(C) = \int_{r=-\infty}^{r=+\infty} \int_{x=-\infty}^{x=+\infty} \frac{1}{c_t} f_\psi^c\left(t, \frac{x}{c_t}\right) \cdot \frac{1}{1-c_t} f_\phi^u\left(t, \frac{r-x}{1-c_t}\right) dx\, dr$$

$$\cdot \prod_{t' \in T \setminus t} \int_{r'=-\infty}^{r'=r} \int_{x=-\infty}^{x=+\infty} \frac{1}{c_{t'}} f_\psi^c\left(t', \frac{x}{c_{t'}}\right) \cdot \frac{1}{1-c_{t'}} f_\phi^u\left(t', \frac{r'-x}{1-c_{t'}}\right) dx\, dr'$$

Also written as $a_t(C) = \int g_t \prod_{t' \in T \setminus t} G_{t'}$ where $G_t := \int g_t$ and

$$g_t(r) := \int_{x=-\infty}^{x=+\infty} \frac{1}{c_t} f_\psi^c\left(t, \frac{x}{c_t}\right) \cdot \frac{1}{1-c_t} f_\phi^u\left(t, \frac{r-x}{1-c_t}\right) dx$$

While a direct analytic solution of these equations is not tractable, we can use numerical techniques to compute g_t, G_t and $a_t(C)$. In our experiments we test two methods, one using straightforward Monte-Carlo simulation and the second using piecewise-constant functions to approximate f_ϕ^u and f_ϕ^u. The argument-wise multiplication $f_\phi^u \cdot f_\phi^u$ still results in a piecewise constant function which, after the integration operation, results in a piecewise linear function $g_t(r)$. We then re-approximate $g_t(r)$ with a piecewise constant function, integrate $g_t(r)$ to obtain a piecewise linear function $G_t(r)$ and again re-approximate $G_t(r)$ with a piecewise constant function. Each product $g_t \prod_{t' \in T \setminus t} G_{t'}$ is then a piecewise constant function which after the integration operation is represented as a piecewise linear function. The value of that last function approaches $a_t(C)$ as the number of segments approaches infinity. By varying the accuracy of these computations one can trade off optimality for speed, as shown in our experiments.

4. SOLUTION METHODS

To solve the model described in the previous section we need to find a Bayesian Stackelberg equilibrium which gives and optimal coverage strategy for the defender and optimal response for every attacker type. If there are a finite number of attacker types, an optimal defender strategy can be found using DOBSS [18]. Unfortunately, there are no known methods for finding exact equilibrium solutions for infinite Bayesian Stackelberg games, and DOBSS only scales to small numbers of types. Here we focus on methods for approximating solutions to infinite Bayesian Stackelberg games. The problem can be broken down into two parts:

1. Computing/estimating the attacker response function (Eqn 7)
2. Optimizing over the space of defender strategies, given the attacker response function

In the previous section we were able to derive the form of the attacker response function, but we lack any means to compute this function analytically. As described above, we explore both brute-force Monte-Carlo sampling and a piecewise-constant function approximation method to approximate this function. In addition, we explore a variety of different approaches for optimizing the defender strategy. Overall, we describe five different approximate solution methods.

4.1 Sampled Bayesian ERASER

Our first method combines Monte-Carlo sampling from the space of attacker types with an exact optimization over the space of defender strategies. This approach is based on the DOBSS solver [18] for finite Bayesian Stackelberg games. However, we also incorporate several improvements from the ERASER solver [13] that offer faster solutions for the restricted class of security games. The resulting method can be encoded as a mixed-integer linear program (MIP), which we call *Bayesian ERASER* (not presented here due to space constraints).

To use Bayesian ERASER to approximate a solution for an infinite game we draw a finite number of sample attacker types from the type distribution, assuming that each occurs with equal probability. The payoffs for each type are determined by drawing from the payoff distributions specified in Equations 4 and 5. This results in a constrained, finite version of the infinite game that can be solved using the Bayesian ERASER MIP. We refer to this method as *Sampled Bayesian ERASER* (SBE) and use SBE-x to denote this method with x sample attacker types. Armantier et al. [4] develop an approach for approximating general infinite Bayesian games that relies on solving constrained versions of the original game. Given certain technical conditions, a sequence of equilibria of constrained games will converge to the equilibrium of the original game. Here, increasing the number of sample types corresponds to such a sequence of constrained games, so in the limit as the number of samples goes to infinity the equilibrium of SBE-∞ will converge to the true Bayesian Nash equilibrium.

4.2 Sampled Replicator Dynamics

The second algorithm uses a local search method (replicator dynamics) to approximate the defender's optimal strategy, given the attacker response function. Given that we are already using numerical techniques to estimate the attacker response, it is sensible to explore approximations for the defender's optimization problem as well. This allows us to trade off whether to use additional computational resource to improve the attacker response estimation or the defender strategy optimization.

Sampled Replicator Dynamics (SRD) is based on replicator dynamics [23]. Since this is a form of local search, all we require is a black-box method to estimate the attacker response function. We could use either Monte-Carlo sampling or piecewise-constant approximation, but use Monte-Carlo in our experiments. As above, we use SRD-x to denote SRD with x sample attacker types. SRD proceeds in a sequence of iterations. At each step the current coverage strategy $C^n = (c_t^n)_{t \in T}$ is used to estimate the attacker response function, which in turn is used to estimate the expected payoffs for both players. A new coverage strategy $C^{n+1} = (c_t^{n+1})_{t \in T}$ is computed according to the replicator equation:

$$c_t^{n+1} \propto c_t^n \cdot (E_t(C) - U_\Theta^{min}), \tag{8}$$

where U_Θ^{min} represents the minimum possible payoff for the defender, and $E_t(C)$ is the expected payoff the defender gets for covering target t with probability 1 and all other targets with probability 0, given the estimated attacker response to C^n. The search runs for a fixed number of iterations, and returns the coverage vector with the highest expected payoff. We introduce a learning rate parameter α that interpolates between C^n and C^{n+1}, with C^{n+1} receiving weight α in the next population and C^n having weight $1 - \alpha$. Finally, we introduce random restarts to avoid becoming stuck in local optima. After initial experiments, we settled on a learning rate of $\alpha = 0.8$ and random restarts every 15 iterations, which generally yielded good results (though the solution quality was not highly sensitive to these settings).

4.3 Greedy Monte Carlo

Our next algorithm combines a greedy heuristic for allocating defender resources with a very fast method for updating the attacker response function estimated using Monte-Carlo type sampling. We call this algorithm Greedy Monte-Carlo (GMC). The idea of the greedy heuristic is to start from a coverage vector that assigns 0 probability to every target. At each iteration, the algorithm evaluates the prospect of adding some small increment (Δ) of coverage probability to each target. The algorithm computes the difference between the defender's expected payoff for the current coverage vector C and the new coverage vector that differs only in the coverage for a single target t such that $c'_t = c_t + \Delta$. The target with the maximum payoff gain for the defender is selected, Δ is added to the coverage for that target, and the algorithm proceeds to the next iteration. It terminates when all of the available resources have been allocated.

The idea of using a greedy heuristic for allocating coverage probability is motivated in part by the ORIGAMI algorithm [13] that is known to be optimal for the case without uncertainty about attacker payoffs. That algorithm proceeds by sequentially allocating coverage probability to the set of targets that give the attacker the maximal expected payoff. In the Bayesian case there is no well-defined set of targets with maximal payoff for the attacker since each type may have a different optimal target to attack, so we choose instead to base the allocation strategy on the defender's payoff.

In principle, any method for estimating the attacker response function could be used to implement this greedy algorithm. However, we take advantage of the fact that the algorithm only requires adding coverage to a single target at a time to implement a very fast method for estimating the attacker response function. We begin by using Monte-Carlo sampling to generate a large number of sample attacker types. For each target we maintain a list containing the individual attacker types that will attack that target, given the current coverage vector. For each type ω we track the current expected payoff for each target, the *best* target to attack, and the *second* best target to attack. These can be used to calculate the minimum amount of coverage δ that would need to be added to current coverage c_{best} of the *best* target to induce type ω to switch to attacking the *second* best target instead. Formally, the target switching condition:

$$\begin{aligned}(c_{best} + \delta)U_\Psi^c(\mathit{best}, \omega) \quad &+ \quad (1 - (c_{best} + \delta))U_\Psi^u(\mathit{best}, \omega) \\ = (c_{second})U_\Psi^c(\mathit{second}, \omega) \quad &+ \quad (1 - c_{second})U_\Psi^u(\mathit{second}, \omega)\end{aligned}$$

Allows us to derive:

$$\delta = \frac{(c_{second})U_\Psi^c(\mathit{second}, \omega) + (1 - c_{second})U_\Psi^u(\mathit{second}, \omega)}{U_\Psi^c(\mathit{best}, \omega) - U_\Psi^u(\mathit{best}, \omega)} - \frac{(c_{best})U_\Psi^c(\mathit{best}, \omega) - (c_{best})U_\Psi^u(\mathit{best}, \omega)}{U_\Psi^c(\mathit{best}, \omega) - U_\Psi^u(\mathit{best}, \omega)}. \tag{9}$$

Using this data structure we can quickly compute the change in the defender's expected payoff for adding Δ coverage to a target t. There are three factors to account for:

1. The defender's expected payoff for an attack on t increases
2. The probability that the attacker will choose t may decrease, as some types may no longer have t as a best response
3. The probability that other targets are attacked may increase if types that were attacking t choose different targets instead

For every type in the list for target t we determine whether or not the type will change using Eqn. 9. If the type changes we update the payoff against that type to be the expected defender payoff associated with the second best target for that type. If not, the payoff against that type is the new defender expected payoff for target t with coverage $c_t + \Delta$. After adjusting the payoffs for every type that was attacking target t in this way we have the change in the defender expected payoff for adding Δ for target t.

After computing the potential change for each target we select the target with the maximum gain for the defender and add the Δ coverage units to that target. We update the data structure containing the types by updating the expected value for the changed target for every type (regardless of which target it is currently attacking). If the target updated was either the best or second best target for a type, we recompute the best and second best targets and, if necessary, move the type to the list for the new best target.

Based on our initial experiences with the GMC method we added two modifications to prevent the algorithm from becoming stuck in local optima in specific cases. First, we placed a lower bound of 1% on the Δ used during the calculations to compute the value of adding coverage to each target, even through the actual amount of coverage added once the best target is selected may be much smaller. In practice, this smoothes out the estimated impact of types changing to attack different targets by averaging over a larger number of types. Second, for cases with a very small numbers of types we use an "optimistic" version of the heuristic in which we assume that the new value for any type that changes to attacking a new target gives the maximum of the current value or the value for the new target (for the defender). The intuition for this heuristic is that it assumes that additional coverage could later be added to the second-best target to make the type to switch back.

4.4 Worst-Case Interval Uncertainty

We also consider an approach based on minimizing the worst-case outcome, assuming interval uncertainty over the attacker's payoffs. The BRASS algorithm [19] was originally designed to model bounded rationality in humans. Rather than the standard assumption that attackers will choose an optimal response, BRASS assumes that attackers will choose any response in the set of responses with expected value within ϵ units of the optimal response, where ϵ is a parameter of the algorithm. The algorithm optimizes the defender's optimal payoff for the worst-case selection of the attacker within the set of feasible responses defined by ϵ.

While this technique was originally motivated as a way to capture deviations from perfect rationality in human decision-making, here we reinterpret the method as a worst-case approach for payoff uncertainty. Suppose that the defender does not know the attacker's payoffs with certainty, but knows only that each payoff is within an interval of $mean \pm \frac{\epsilon}{2}$. Then an attacker playing optimally could attack any target within ϵ of the target with the best expected value based on the means (since the "best" value could be up to $\frac{\epsilon}{2}$ too high, and the value for another target could be up to $\frac{\epsilon}{2}$ too low).

4.5 Decoupled Target Sets

Our last method for solving Infinite Bayesian Stackelberg Games is called Decoupled Target Sets (DTS). DTS is an approximate solver, for it assumes that the attacker preference as to which target $t \in D \subset \{1, 2, ..., T\}$ to attack depends on the probabilities c_t of targets $t \in D$ being covered, but does *not* depend on the probabilities $c_{\bar{t}}$ of targets $\bar{t} \in \overline{D} := \{1, 2, ..., T\} \setminus D$ being covered. For example, let $D = \{1, 2\} \subset \{1, 2, 3\}$. Here, DTS assumes that when the attacker evaluates whether it is more profitable to attack target 1 than to attack target 2, the attacker needs to know the probabilities c_1, c_2 but does *not* have to reason about the probability c_3 of target 3 being covered. While this attacker strategy appears sound (after all, "Why should the attacker bother about target 3 when it debates whether it is better to attack target 1 than to attack target 2?"), it can be shown that it is not always optimal. In general then, DTS assumes that for any two coverage vectors $C = (c_t)_{t \in D \cup \overline{D}}$, $C' = (c'_t)_{t \in D \cup \overline{D}}$ such that $c_t = c'_t$ for all $t \in D$, it holds that

$$\frac{a_t(C)}{a_{t'}(C)} = \frac{a_t(C')}{a_{t'}(C')} \quad \text{for any } t, t' \in D. \tag{10}$$

The immediate consequence of this assumption is that a systematic search for the optimal coverage vector can be performed incrementally, considering larger and larger sets of targets $D \subset \{1, 2, \ldots T\}$ (by adding to D a target from $\{1, 2, \ldots, T\} \setminus D$ in each algorithm iteration). In particular, to find an optimal coverage vector for targets $\{1, 2, \ldots d\}$, DTS reuses the optimal coverage vectors (for coverage probability sums $c_1 + c_2 + \ldots + c_{d-1}$ ranging from 0 to 1) for targets $\{1, 2, \ldots, d-1\}$ alone (found at previous algorithm iteration) while ignoring the targets $\{d+1, d+2, \ldots, T\}$. Assuming that a probability of covering a target is a multiple of ϵ, DTS's search for the optimal—modulo assumption (10)—coverage vector can be performed in time $O(\epsilon \cdot T)$. Our implementation of DTS uses the piecewise-constant attacker response approximation method.

5. EXPERIMENTAL EVALUATION

We present experimental results comparing the solution quality and computational requirements of the different classes of approximation methods introduced previously.

5.1 Experimental Setup

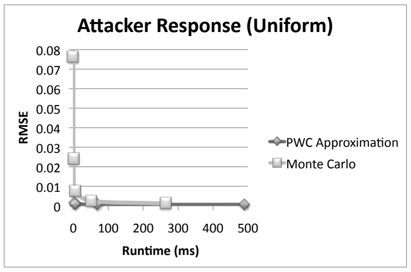
Our experiments span three classes of security games, each with a different method for selecting the distributions for attacker payoffs. In every case we first draw both penalty and reward payoffs for both the attacker and defender. All rewards are drawn from $U[6, 8]$ and penalties are drawn from $U[2, 4]$. We then generate payoff distributions for the attacker's payoffs using the values drawn above as the mean for the distribution. In *uniform games* the attacker's payoff is a uniform distribution around the mean, and we vary the length of the intervals to increase or decrease uncertainty. For *Gaussian games* the distributions are Gaussian around the mean payoff, with varying standard deviation. In both cases, all distributions for a particular game have the same interval size or standard deviation. The final class of games, *Gaussian Variable*, models a situation where some payoffs are more or less certain by using Gaussian distributions with different standard deviations for each payoff. The standard deviations themselves are drawn from either $U[0, 0.5]$ or $U[0.2, 1.5]$ to generate classes with "low" or "high" uncertainty on average.

Our solution methods generate coverage strategies that must be evaluated based on the attacker response. Since we do not have a way to compute this exactly, we compute the expected payoffs for any particular strategy by finding an extremely accurate estimate of the attacker response using 100000 Monte-Carlo samples. We employ two baseline methods in our experiments. The first simply plays a uniform random coverage strategy, such that each target is covered with equal probability using all available resources. The second uses the mean of each attacker distribution as a point estimate of the payoff. This is a proxy for models in which experts are forced to specify a specific value for each payoff, rather than directly modeling any uncertainty about the payoff. This can be solved using the SBE method, using the mean payoffs to define a single attacker type.

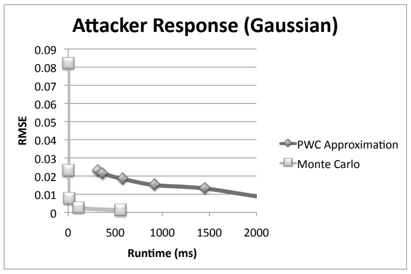
5.2 Attacker Response Estimation

We implemented two different methods for estimating the attacker response function. The first uses Monte-Carlo sampling to generate a finite set of attacker types. To estimate the response probabilities we calculate the best response for each sample type and use the observed distribution of targets attacked as the estimated probabilities. The second method approximates each distribution using a piecewise constant (PWC) function and directly computes the result of Equation 7 for these functions.

Figures 1(a) and 1(b) compare the estimation accuracy for these two methods. Results are averaged over 100 sample games, each with 10 targets and 1 defender resource. For each game we draw a random coverage vector uniformly from the space of defender strategies to evaluate. For the uniform case, mean attacker payoffs are drawn from U[5,15] for the covered case and U[25,35] for the uncovered case, and every distribution has a range of 10 centered on the mean. For the Gaussian case, mean payoffs are drawn from U[2.5,3.5] for the covered case and U[5,6] for the uncovered case, with standard deviations for each distribution drawn from U[0,0.5]. Each method has a parameter controlling the tradeoff between solution quality and computation time. For Monte-Carlo sampling this is the number of sample types, and for the PWC approximation it is the absolute difference in function values between two adjacent constant intervals. To enable easy comparison, we plot the solution time on the x-axis, and the solution quality for each method on the y-axis (rather than the raw parameter settings). Solution quality is measured based on the root mean squared error from an estimate of the true distribution based on 100000 sample attacker types. We see that in the uniform case, PWC approximation generally offers a better tradeoff between solution time and quality. However, for



(a) Estimation time vs. accuracy for uniform distributions.



(b) Estimation time vs. accuracy for Gaussian distributions.

Figure 1: Comparison of Monte-Carlo and piecewise-constant estimation methods for the attacker response function.

the more complex Gaussian distributions the Monte-Carlo method gives better performance.

5.3 Approximation Algorithms

We next compare the performance of the full approximation algorithms, evaluating both the quality of the solutions they produce and the computational properties of the algorithms. The first set of experiments compares all of the algorithms and the two baseline methods (uniform and mean) on small game instances with 5 targets and 1 defender resource. We generated random instances from each of the three classes of games described in Section 5.1: Uniform, Gaussian, and Gaussian Variable, varying the level of payoff uncertainty using the parameters described above. We used 100 games instances for every different level of payoff uncertainty in each class of games. The tests are paired, so every algorithm is run on the same set of game instances to improve the reliability of the comparisons.[1]

The first three plots, Figures 2(a), 2(b), and 2(c) show a comparison of the best solution quality achieved by each algorithm in the three classes of games. The y-axis shows the average expected defender reward for the computed strategies, and the x-axis represents the degree of uncertainty about the attacker's payoffs. Each algorithm has parameters that can affect the solution quality and computational costs of generating a solution. We tested a variety of parameter settings for each algorithm, which are listed in Table 1. For cases with more than one parameter we tested all combinations of the parameter settings shown in the table. The first set of results reports the *maximum* solution quality achieved by each algorithm over any of the parameter settings to show the potential

[1]In general, there is substantial variance in the overall payoffs due to large differences in the payoffs for each game instance (i.e., some games are inherently more favorable than others). However, the differences in performance between the algorithms on each individual instance are much smaller and very consistent.

Table 1: Parameter settings for the algorithms tested in the first experiment.

Parameter	Values
SBE num types	1, 3, 5, 7
BRASS epsilon	0.1, 0.2, 0.3, 0.5, 1.0
SRD num types	10, 50, 100, 1000
SRD num iterations	1000, 10000
GMC num types	100, 1000, 10000
GMC coverage increment	0.01, 0.001, 0.0001
DTS max error	0.02, 0.002
DTS step size	0.05, 0.02
DTS coverage increment	0.05, 0.02

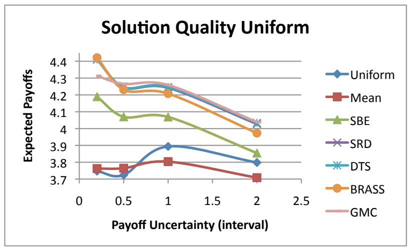
quality given under ideal settings. The settings that yield the best performance may differ in the different types of games and level of uncertainty.

The results are remarkably consistent in all of the conditions included in our experiment. First, we observe that the baseline method "mean" that uses point estimates of payoff distributions performs extremely poorly in these games–in many cases it is actually worse than playing a uniform random strategy! SBE performs somewhat better, but is severely limited by an exponential growth in solution time required to find an exact optimal defender strategy as the number of sample attacker types increases. The maximum number of types we were able to run in this experiment was only seven (many orders of magnitude smaller than the number of sample types used for the other methods).

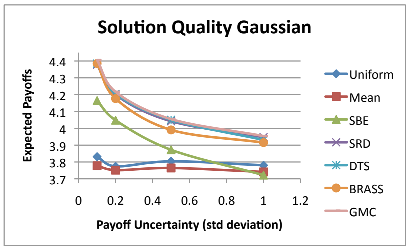
All four of the remaining methods (SRD, BRASS, GMC, and DTS) give much higher solutions quality than either of the baselines or the SBE method in all cases. These methods are similar in that all four rely on approximation when computing the defender's strategy, but they use very different approaches. It is therefore quite surprising that the expected payoffs for all four methods are so close for these small games. This is true when we look at the data for individual game instances as well as in aggregate. On any individual instance, the difference between the best and worst solution generated by one of these four is almost always less than 0.05 units. This suggests that the strategies generated by all of these algorithms are very close to optimal in these games. Overall, the GMC method does outperform the others by a very small margin. This is also consistent on a game-by-game basis, with GMC generating the best strategy in over 90% of the game instances.

To this point we have focused on the the best solution quality possible with each method. We now extend the analysis to include the tradeoff of computational speed versus increased solution quality. This is particularly complex because of the large number of potential parameter settings for each algorithm and the fact that these parameters do not have the same interpretation. To analyze this tradeoff, we plot the solution quality against the solution time for each of the parameter settings of the different algorithms. The data for Gaussian games with attacker standard deviations of 0.2 is presented in Figure 3. Other classes of games have similar results. Solution time (in ms) is given on the x-axis in a log scale, and solution quality is reported on the y-axis as before.

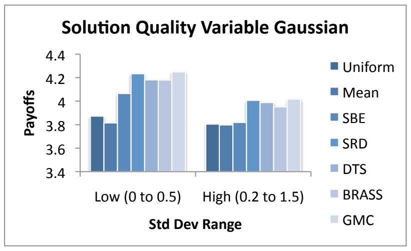
The upper-left corner of the plot corresponds to high solution quality and low computational efforts, so it is most desirable. Points from the GMC and SRD methods dominate this part of the figure, indicating that these methods are computationally scalable and give high-quality solutions. In constrast, SBE scales very poorly; even



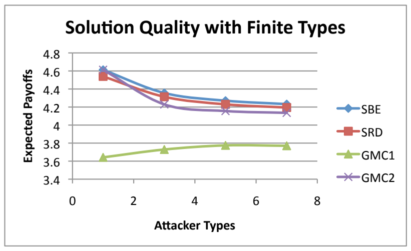
(a) Solution quality comparison for infinite games with uniform attacker payoff distributions.



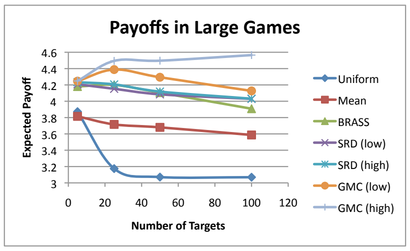
(b) Solution quality comparison for infinite games with Gaussian attacker payoff distributions with identical standard deviations.



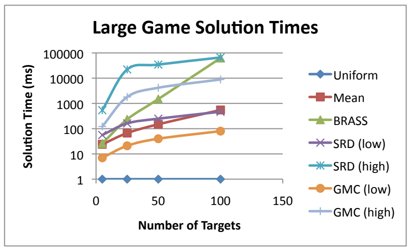
(c) Solution quality comparison for infinite games with Gaussian attacker payoff distributions with varying standard deviations.



(d) Solution quality results for small games with a finite set of known attacker types.



(e) Solution quality results when scaling to larger game instances for Gaussian attacker payoff distributions with varying standard deviations.



(f) Computational cost comparison for solving large games instances of Gaussian variable games.

Figure 2: Solution quality and computation time comparisons.

after 10000ms SBE still has a lower solution quality than any of the data points for GMC, SRD, or DTS. DTS consistently has high solution quality, but takes much longer than GMC or SRD even in the best case. BRASS has a different pattern of performance than the other methods. Every parameter setting takes roughly the same amount of time, they vary dramatically in solution quality. This is because the best setting for the ϵ parameter depends on the amount of uncertainty in the game, and is not directly related to the quality of approximation in the same way as the parameters for the other algorithms. In practice this is a significant disadvantage, since it is not obvious how to set the value of ϵ for any particular problem. This can be determined empirically (as in our experiments), but it requires running BRASS multiple times with different settings to find a good value.

Our next experiment focuses on the quality of the approximations for SRD and GMC in a situation where an optimal solution can be computed. For finite Bayesian Stackelberg games with a small number of types we can compute an exact optimal response using SBE. Since both SRD and GMC use Monte-Carlo sampling to approximate the attacker type distribution for infinite games, we can also apply these methods to finite games with known types. In this experiment, we test SBE, SRD, and GMC on finite games with exactly the same types. The games are generated from the Gaussian infinite games with standard deviations of 0.2, but once the types are drawn, these are interpreted as known finite games. Results are shown in Figure 2(d), with the number of attacker types on the x-axis and solution quality on the y-axis. GMC1 is GMC with the original greedy heuristic, and GMC2 uses the modified optimistic greedy heuristic. We can see in this experiment that SRD and GMC2 both achieve very close to the true optimal defender strategy in these games, but GMC1 performs poorly. In general, GMC1 performs very well in games with large numbers of types (such as when we are approximating the infinite case), but GMC2 is preferable when there is a very small number of types.

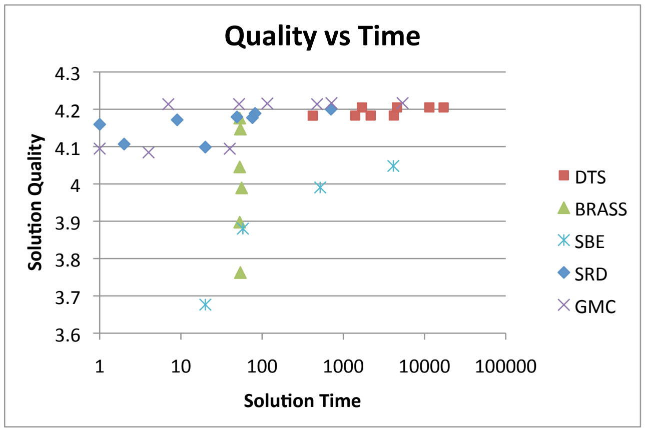


Figure 3: Comparison of the tradeoff in solution quality and computational cost for each of the algorithms, exploring the effects of different parameter settings.

The final experiment we report takes the three most scalable methods (SRD, GMC, and BRASS) and tests them on much larger game instances. We run this experiment on the Gaussian variable class of games with standard deviations drawn $U[0, 0.5]$. The number of targets varies between 5 and 100 in this experiment, with the number of resources set to 20% of the number of targets in each case. Due to the increased computational time to run experiments, we use only 30 sample games for each number of targets in this experiment. For SRD and GMC we tested "low" and "high" computational effort parameter settings. Solution quality results are shown in Figure 2(e), and timing results are presented in Figure 2(f).

The three approximate methods all clearly outperform both the uniform and mean baselines. As the number of targets increases, the mean method shows some improvement over the uniform random strategy. BRASS and the two variants of SRD both have similar solution quality scores. The most striking result is that both the low and high effort version of GMC significantly outperform all of the other methods for larger games, while also having relatively faster solution times.

6. CONCLUSION

Developing the capability to solve large game models with rich representations of uncertainty is critical to expanding the reach of game-theoretic solutions to more real-world problems. This cuts to the central concern of ensuring that users have confidence that their knowledge is accurately represented in the model. Our experiments reinforce that experts and game theorists should not be comfortable relying on perfect-information approximations when there is uncertainty in the domain. Relying on a perfect information approximation such as the mean baseline in our experiments resulted in very poor decisions—closer in quality to the uniform random baseline than to our approximate solvers that account for distributional uncertainty.

In this work we developed and evaluated a wide variety of different approximation techniques for solving infinite Bayesian Stackelberg games. These algorithms have very different properties, but all show compelling improvements over existing methods. Of the approximate methods, Greedy Monte-Carlo (GMC) has the best performance in solution quality and scalability, and Sampled Replicator Dynamics (SRD) also performs very well. As a group, the approximate solvers introduced here constitute the only scalable algorithms for solving a very challenging class of games with important real-world applications.

Acknowledgements

This research was supported by the United States Department of Homeland Security through the Center for Risk and Economic Analysis of Terrorism Events (CREATE) under grant number 2007-ST-061-000001. This research was also sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-06-3-0001. The views and conclusions in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

7. REFERENCES

[1] M. Aghassi and D. Bertsimas. Robust game theory. *Mathematical Programming: Series A and B*, 107(1):231–273, 2006.

[2] N. Agmon, S. Kraus, G. A. Kaminka, and V. Sadov. Adversarial uncertainty in multi-robot patrol. In *IJCAI*, 2009.

[3] T. Alpcan and T. Basar. A game theoretic approach to decision and analysis in network intrusion detection. In *Proc. of the 42nd IEEE Conference on Decision and Control*, pages 2595–2600, 2003.

[4] O. Armantier, J.-P. Florens, and J.-F. Richard. Approximation of Bayesian Nash equilibrium. *Journal of Applied Econometrics*, 23(7):965–981, December 2008.

[5] N. Basiloco, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, 2009.

[6] V. M. Bier. Choosing what to protect. *Risk Analysis*, 27(3):607–620, 2007.

[7] G. Cai and P. R. Wurman. Monte Carlo approximation in incomplete information, sequential auction games. *Decision Support Systems*, 39(2):153–168, 2005.

[8] S. Ceppi, N. Gatti, and N. Basilico. Computing Bayes-Nash equilibria through support enumeration methods in Bayesian two-player strategic-form games. In *Proceedings of the ACM/IEEE International Conference on Intelligent Agent Technology (IAT)*, pages 541–548, Milan, Italy, September 15-18 2009.

[9] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *ACM EC*, pages 82–90, 2006.

[10] N. Gatti. Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. In *ECAI*, pages 403–407, 2008.

[11] J. C. Harsanyi. Games with incomplete information played by Bayesian players (parts i–iii). *Management Science*, 14, 1967–8.

[12] M. Jain, C. Kiekintveld, and M. Tambe. Quality-bounded solutions for finite Bayesian Stackelberg games: Scaling up. In *AAMAS*, 2011.

[13] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, 2009.

[14] V. Krishna. *Auction Theory*. Academic Press, 2002.

[15] R. D. Luce and H. Raiffa. *Games and Decisions*. John Wiley and Sons, New York, 1957. Dover republication 1989.

[16] D. McFadden. Quantal choice analysis: A survey. *Annals of Economic and Social Measurement*, 5(4):363–390, 1976.

[17] K. C. Nguyen and T. A. T. Basar. Security games with incomplete information. In *Proc. of IEEE Intl. Conf. on Communications (ICC 2009)*, 2009.

[18] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *AAMAS*, pages 895–902, 2008.

[19] J. Pita, M. Jain, F. Ordóñez, M. Tambe, S. Kraus, and R. Magori-Cohen. Effective solutions for real-world Stackelberg games: When agents must deal with human uncertainties. In *AAMAS*, 2009.

[20] J. Pita, M. Jain, C. Western, C. Portway, M. Tambe, F. Ordonez, S. Kraus, and P. Paruchuri. Depoloyed ARMOR protection: The application of a game-theoretic model for security at the Los Angeles International Airport. In *AAMAS (Industry Track)*, 2008.

[21] D. M. Reeves and M. P. Wellman. Computing best-response strategies in infinite games of incomplete information. In *UAI*, 2004.

[22] T. Sandler and D. G. A. M. Terrorism and game theory. *Simulation and Gaming*, 34(3):319–337, 2003.

[23] P. Taylor and L. Jonker. Evolutionary stable strategies and game dynamics. *Mathematical Biosciences*, 16:76–83, 1978.

[24] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe. IRIS - A tools for strategic security allocation in transportation networks. In *AAMAS (Industry Track)*, 2009.

[25] H. von Stackelberg. *Marktform und Gleichgewicht*. Springer, Vienna, 1934.

[26] K. wei Lye and J. M. Wing. Game strategies in network security. *International Journal of Information Security*, 4(1–2):71–86, 2005.