8

Computing Optimal Randomized Resource Allocations for Massive Security Games

Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe

8.1 Introduction

Providing security for transportation systems, computer networks, and other critical infrastructure is a large and growing problem. Central to many of these security problems is a resource allocation task. For example, a police force may have limited personnel to conduct patrols, operate checkpoints, and conduct random searches. Other scarce resources including bomb-sniffing canines, vehicles, and security cameras. The key question is how to efficiently allocate these resources to protect against a wide variety of potential threats.

The adversarial aspect of security domains poses unique challenges for resource allocation. A motivated attacker can gather information about security measures using surveillance and plan more effective attacks. Predictable resource allocations may be exploited by an attacker, greatly reducing resource effectiveness. A better approach for deploying security resources is to use *randomization* to increase the uncertainty of potential attackers. We develop new computational methods that use game-theoretic analysis to generate optimal randomized resource allocations for security domains.

Game theory offers a more sophisticated approach to randomization than simply "rolling dice." It allows the analyst to factor differential risks and values into the game model, and incorporates game-theoretic predictions of how the attacker will respond to a given security policy. Recent work by Paruchuri et al. uses a game-theoretic approach to create randomized security policies for traffic checkpoints and canine patrols at the Los Angeles International Airport (LAX), which are deployed in the daily airport-security operations (Paruchuri et al., 2008; Pita et al., 2008). They build on recent advances in solution algorithms

Previously published in *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary. Copyright © 2009, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

for Bayesian Stackelberg games, which capture the surveillance aspect of the domain.

A significant limitation of existing solution methods is that they handle multiple security by enumerating all possible combinations of resource assignments. This grows combinatorially in the number of resources, which makes it computationally infeasible to solve large problems. Enumeration was feasible in the LAX application due to the relatively small size of the domain (on the order of ten resources). We are interested in application domains with thousands of attack targets and hundreds of resources. Section 8.3 describes one such domain – the problem of assigning federal air marshals (FAMs) to flights.

We introduce new techniques for randomized security resource allocation that scale to problems orders of magnitude larger than existing methods. The first algorithm introduces a compact representation to dramatically reduce both space and time requirements for the multiple-resource case. Two additional algorithms further improve performance by exploiting payoff regularities in a restricted class of security games. Finally, we extend the first algorithm to incorporate additional scheduling and resource constraints into the model. We demonstrate compelling performance improvements in both memory and runtime on random problem instances and realistic data from the LAX and FAMS domains.

8.2 Game-Theoretic Modeling of Security Games

Security problems are increasingly studied using game-theoretic analysis, ranging from computer network security (Wei Lye and Wing, 2005; Srivastava et al., 2005) to terrorism (Sandler and D.G.A.M., 2003). We model the security resource allocation problem as a Stackelberg game. Stackelberg games were introduced to study duopoly competition (von Stackelberg, 1934) and are now widely used to model leadership and commitment. A related family of Stackelberg games called inspection games includes models of arms inspections and border patrolling (Avenhaus, von Stengel, and Zamir, 2002). Stackelberg games have also been applied to resource allocation in a computer job-scheduling domain (Roughgarden, 2004). The most directly related work applies Stackelberg games to security patrolling, both in a generic "police and robbers" scenario (Gatti, 2008) and in a fielded application at LAX (Pita et al., 2008).

Motivated by these and other applications, there have been several recent algorithmic advances for Stackelberg games. Conitzer and Sandholm give complexity results and algorithms for computing optimal commitment strategies,

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

including both pure and mixed-strategy commitments and a Bayesian case (Conitzer and Sandholm, 2006). A new algorithm for solving Bayesian Stackelberg games (DOBSS) is central to the LAX application (Parachuri et al., 2008). We consider cases with multiple resources, which can be solved using the existing algorithms only by exhaustively enumerating an exponential number of joint assignments. Our algorithms use compact representations to dramatically reduce the time and space requirements to compute solutions for common classes of Stackelberg security games with multiple resources.

8.2.1 Stackelberg Security Games

We begin by defining a generic security problem as a normal-form Stackelberg game. A security game has two players, a *defender*, Θ, and an *attacker*, Ψ. These players need not represent individuals, but can also be groups that cooperate to execute a joint strategy, such as a police force or a terrorist organization. Each player has a set of possible *pure strategies*, denoted $\sigma_\Theta \in \Sigma_\Theta$ and $\sigma_\Psi \in \Sigma_\Psi$. A *mixed strategy* allows a player to play a probability distribution over pure strategies, denoted $\delta_\Theta \in \Delta_\Theta$ and $\delta_\Psi \in \Delta_\Psi$. Payoffs for each player are defined over all possible joint pure-strategy outcomes: $\Omega_\Theta : \Sigma_\Psi \times \Sigma_\Theta \to \mathcal{R}$ for the defender, and similarly for the attacker. The payoff functions are extended to mixed strategies in the standard way by taking the expectation over pure-strategy outcomes.

So far, our game description follows the standard normal-form game. Stackelberg games introduce a distinction between the players: a "leader" moves first, and the "follower" observes the leader's strategy before acting. In our security games, the defender is a Stackelberg leader, and the attacker is a follower. This models the capability of malicious attackers to employ surveillance in planning attacks. In this model, predictable defense strategies are vulnerable to exploitation by a determined adversary. Formally, the attacker's strategy in a Stackelberg security game is a function that selects a strategy in response to each leader strategy: $F_\Psi : \Delta_\Theta \to \Delta_\Psi$.

8.2.2 Stackelberg Equilibrium

The standard solution concept in game theory is a *Nash equilibrium*: a strategy profile (strategy for each player) such that no player can gain by unilaterally deviating to another strategy (Osbourne and Rubinstein, 1994). Stackelberg equilibrium is a refinement of Nash equilibrium that is specific to Stackelberg games. It is a form of subgame perfect equilibrium, in that each player chooses a best-response in any subgame of the original (where subgames correspond to partial sequences of actions). This eliminates Nash equilibrium profiles supported by noncredible threats off the equilibrium path. Subgame perfection

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

alone does not guarantee a unique solution for Stackelberg games since the follower can be indifferent among a set of strategies. There are two types of unique Stackelberg equilibria, first proposed by Leitmann (1978), and 'typically called "strong" and "weak" after Breton et. al. (1988). The strong form assumes that the follower will always choose the optimal strategy for the leader in cases of indifference, while the weak form assumes that the follower will choose the worst strategy for the leader. A strong Stackelberg equilibrium exists in all Stackelberg games, but a weak Stackelberg equilibrium may not (Basar and Olsder, 1995). In addition, the leader can often induce the favorable strong equilibrium by selecting a strategy arbitrarily close to the equilibrium that causes the the follower to strictly prefer the desired strategy (von Stengel and Zamir, 2004). We adopt strong Stackelberg equilibrium (SSE) here due to the key existence result and because it is the most commonly adopted concept in the related literature (Conitzer and Sandholm, 2006; Osbourne and Rubinstein, 1994; Parachuri et al., 2008).

Definition 8.1. *A pair of strategies* (δ_Θ, F_Ψ) *form an SSE if they satisfy the following:*

1. *The leader plays a best-response:*
   $\Omega_\Theta(\delta_\Theta, F_\Psi(\delta_\Theta)) \geq \Omega_\Theta(\delta'_\Theta, F_\Psi(\delta'_\Theta)) \, \forall \, \delta'_\Theta \in \Delta_\Theta$.
2. *The follower play a best-response:*
   $\Omega_\Psi(\delta_\Theta, F_\Psi(\delta_\Theta)) \geq \Omega_\Psi(\delta_\Theta, \delta_\Psi) \, \forall \, \delta_\Theta \in \Delta_\Theta, \delta_\Psi \in \Delta_\Psi$.
3. *The follower breaks ties optimally for the leader:*
   $\Omega_\Theta(\delta_\Theta, F_\Psi(\delta_\Theta)) \geq \Omega_\Theta(\delta_\Theta, \delta_\Psi) \, \forall \, \delta_\Theta \in \Delta_\Theta, \delta_\Psi \in \Delta^*_\Psi(\delta_\Theta)$, *where* $\Delta^*_\Psi(\delta_\Theta)$ *is the set of follower best-responses, as in number 2.*

Whether or not the Stackelberg leader benefits from the ability to commit depends on whether commitment to mixed strategies is allowed. Committing to a pure strategy can be either good or bad for the leader; for example, in the "rock, paper, and scissors" game, committing to a pure strategy guarantees a loss. However, the ability to commit to a mixed strategy always weakly increases the leader's payoffs in the game's equilibrium profiles (von Stengel and Zamir, 2004). In the context of a Stackelberg security game, a deterministic policy is a liability for the defender (the leader), but a credible randomized security policy is an advantage. Our model allows commitment to mixed strategies by the defender.

8.3 Motivating Domains

The primary problem we address in this work is combinatorial explosion in the game representation for security games with multiple resources. Many security

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

domains feature multiple resources, including the LAX application described previously (Pita et al., 2008). In this case, multiple canine units (resources) are assigned to cover multiple targets (airport terminals). The authors solved this problem by enumerating all possible assignments of canines to terminals – roughly 800 possible assignments. We are interested in applications with thousands of resources and targets, such as subway systems, random baggage screening, container inspections at ports, and scheduling for the Federal Air Marshals Service (FAMS). By way of comparison, even 100 targets and 10 resources yields a problem with 1.7×10^{13} assignments; a massive increase from the LAX problem.

The Federal Air Marshal Service (FAMS) has law enforcement authority for commercial air transportation.[1] One important activity of the service is to deploy armed federal air marshals (FAMs) on commercial flights, where they are able to detect, deter, and defeat terrorist/criminal attempts to gain control of the aircraft. As U.S. commercial airlines fly 27,000 domestic flights and over 2000 international flight each day, FAMS lacks the resources to cover all flights and deployments must be risk-based. However, even the *possibility* that a FAM could be on any given flight is a powerful deterrent for terrorist activities. The effectiveness of this deterrence depends on the ability of the FAMS to randomize the flight schedules for air marshals. If a terrorist adversary were able to reliably predict which flights will not have marshals, the deterrence effect would be reduced.

Flights should not necessarily have equal weighting in a randomized schedule. Information about how flight risks are evaluated is not public, but it is easy to imagine that many factors contribute to the evaluation, ranging from specific intelligence to general risk factors. A game-theoretic approach is ideal for creating a randomized schedule that incorporates these risk factors. However, creating such a schedule is significantly more daunting than even the LAX problem. There are thousands of flights each day, departing from hundreds of airports worldwide, and a multiplicity of air marshals to schedule. Moreover, there are scheduling constraints that must be considered in generating an allocation. An individual air marshal's potential departures are constrained by his current location, and schedules must account for flight and transition times. The algorithms we develop in the sequel are motivated by these challenges.

[1] See the TSA websites http://www.tsa.dhs.gov/lawenforcement/programs/fams.shtm and http://www.tsa.dhs.gov/lawenforcement/programs/fams.shtm for additional information.

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

Table 8.1. *Example payoffs for an attack on a target*

	Covered	Uncovered
Defender	5	–20
Attacker	–10	30

8.4 A Compact Representation for Multiple Resources

Many security domains – including both LAX and FAMS – involve allocating multiple resources to cover many potential targets. They can be represented in normal form, but only at the cost of a combinatorial explosion in the size of the strategy space and payoff representation. We develop a compact representation for multiple resources and introduce an algorithm that exploits this representation. Our approach is similar in spirit to other compact representations for games (Koller and Milch, 2003; Jiang and Leyton-Brown, 2006) is but tailored to security domains.

8.4.1 Compact Security Game Model

Let $T = \{t_1, \ldots, t_n\}$ be a set of *targets* that may be attacked, corresponding to pure strategies for the attacker. The defender has a set of resources available to *cover* these targets, $R = \{r_1, \ldots, r_m\}$ (e.g., in the FAMS domain targets could be flights and air marshals modeled as resources). Here, we assume that all resources are identical and may be assigned to any target, but we relax these assumptions in Section 8.6. Associated with each target are four payoffs defining the possible outcomes for an attack on the target, as shown in Table 8.1. There are two cases, depending on whether or not the target is covered by the defender. The defender's payoff for an uncovered attack is denoted $U_\Theta^u(t)$, and $U_\Theta^c(t)$ for a covered attack. Similarly, $U_\Psi^u(t)$ and $U_\Psi^c(t)$ are the attacker's payoffs.

A crucial feature of the model is that payoffs depend only on the identity of the attacked target and whether or not it is covered by the defender. For example, it does not matter whether or not any unattacked target is covered or not. From a payoff perspective, many resource allocations are identical. We exploit this by summarizing the payoff-relevant aspects of the defender's strategy in a *coverage* vector, C, that gives the probability that each target is covered, c_t. The analogous attack vector A gives the probability of attacking a target, which in the sequel we restrict to attack a single target with probability 1

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

(without loss of generality because a SSE solution still exists). The defender's expected payoff given attack and coverage vectors is shown in Equation 8.1. Equation 8.2 gives the expected payoff for an attack on target t, given C. The same notation applies for the follower, replacing Θ with Ψ. We also define the useful notion of the *attack set*, $\Gamma(C)$, which contains all targets that yield the maximum expected payoff for the attacker given coverage C.

$$U_\Theta(C,A) = \sum_{t \in T} a_t \cdot (c_t \cdot U^c_\Theta(t) + (1-c_t)U^u_\Theta(t)) \tag{8.1}$$

$$U_\Theta(t,C) = c_t U^c_\Theta(t) + (1-c_t)U^u_\Theta(t) \tag{8.2}$$

$$\Gamma(C) = \{t : U_\Psi(t,C) \geq U_\Psi(t',C)\, \forall\, t' \in T\}. \tag{8.3}$$

In an SSE, the attacker selects the target in the attack set with maximum payoff for the defender. Let t^* denote this optimal target. Then the expected SSE payoff for the defender is $\hat{U}_\Theta(C) = U_\Theta(t^*,C)$, and for the attacker is $\hat{U}_\Psi(C) = U_\Psi(t^*,C)$.

8.4.2 Compact versus Normal Form

Any security game represented in this compact form can also be represented in normal form. The attack vector A maps directly to the attacker's pure strategies, with one strategy per target. For the defender, each possible allocation of resources corresponds to a pure strategy in the normal form. A resource allocation maps each available resource to a target, so there are $n\,Choose\,m$ ways to allocate m resources to n targets (assuming at most one resource is assigned to a target). Equation 4 set gives an example of how a coverage vector corresponds to a mixed strategy, for two resources and four targets. The probability assigned to a pure strategy covering targets i and j is $\delta^{i,j}_\Theta$. The first row states that the probability of covering target 1 is the sum of the probability assigned to pure strategies that cover 1.

$$\begin{aligned}
\delta^{1,2}_\Theta + \delta^{1,3}_\Theta + \delta^{1,4}_\Theta &= c_1 \\
\delta^{1,2}_\Theta + \delta^{2,3}_\Theta + \delta^{2,4}_\Theta &= c_2 \\
\delta^{1,3}_\Theta + \delta^{2,3}_\Theta + \delta^{3,4}_\Theta &= c_3 \\
\delta^{1,4}_\Theta + \delta^{2,4}_\Theta + \delta^{3,4}_\Theta &= c_4
\end{aligned} \tag{8.4}$$

The payoff function Ω_Θ for the defender defines a payoff for each combination of a resource allocation schedule and target. If the target is covered by the allocation, the value is U^c_Θ, and if not it is U^u_Θ. The attacker payoff function is defined similarly. Comparing the size of the strategies and payoff functions in

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

these alternative representations is striking. In the compact form, each strategy is represented by n continuous variables, and the payoff function by $4n$ variables. In contrast, the defender's strategy in normal form requires $n\,Choose\,m$ variables, while the attacker strategy remains the same. The payoff function is of size $n \cdot (n\,Choose\,m)$.

8.4.3 ERASER Solution Algorithm

The ERASER (Efficient Randomized Allocation of Security Resources) algorithm takes as input a security game in compact form and solves for an optimal coverage vector corresponding to a SSE strategy for the defender. The algorithm is an MILP, presented in Equations 8.5 through 8.11. Equations 8.6 and 8.7 force the attack vector to assign a single target probability 1. Equation 8.8 restricts the coverage vector to probabilities in the range $[0,1]$, and Equation 8.9 constrains the coverage by the number of available resources.

In Equations 8.10 and 8.11, Z is a large constant relative to the maximum payoff value. Equation 8.10 defines the defender's expected payoff, contingent on the target attacked in A. The constraint places an upper bound of $U_\Theta(t,C)$ on d, but only for the attacked target. For all other targets, the right-hand side is arbitrarily large. Since the objective maximizes d, for any optimal solution $d = U_\Theta(C,A)$. This also implies that C is maximal, given A for any optimal solution, since d is maximized.

In a similar way, Equation 8.11 forces the attacker to select a strategy in the attack set of C. The first part of the constraint specifies that $k - U_\Psi(t,C) \geq 0$, which implies that k must be at least as large as the maximal payoff for attacking any target. The second part forces $k - U_\Psi(t,C) \leq 0$ for any target that is attacked in A. If the attack vector specifies a target that is not maximal, this constraint is violated. Taken together, the objective and Equations 8.10 through 8.11 imply that C and A are mutual best responses in any optimal solution.

$$\max \quad d \tag{8.5}$$

$$a_t \in \{0,1\} \quad \forall t \in T \tag{8.6}$$

$$\sum_{t \in T} a_t = 1 \tag{8.7}$$

$$c_t \in [0,1] \quad \forall t \in T \tag{8.8}$$

$$\sum_{t \in T} c_t \leq m \tag{8.9}$$

$$d - U_\Theta(t,C) \leq (1 - a_t) \cdot Z \quad \forall t \in T \tag{8.10}$$

$$0 \leq k - U_\Psi(t,C) \leq (1 - a_t) \cdot Z \quad \forall t \in T \tag{8.11}$$

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

We now show that an optimal solution to the ERASER MILP corresponds to an SSE of the security game. First, we show that the legal coverage vectors can be implemented by mixed strategies, and then we show how full a full SSE can be constructed from an optimal ERASER solution.

Theorem 8.1. *For any feasible ERASER coverage vector, there is a corresponding mixed strategy δ_Θ that implements the desired coverage probabilities.*

Proof sketch: Translating C into a mixed strategy involves solving a set of n linear equations with $\binom{n}{m}$ variables; in practice, we use a linear program. The claim is trivial when $m = 1$, since each pure strategy maps directly to a target. In the general case, we must map the feasible set of ERASER coverage vectors to the feasible set of the mixed strategies Δ_Θ. We provide the intuition for this mapping here. Each pure strategy σ_Θ can be represented by an m-dimensional indicator vector that selects m out of the possible n targets. The full set of pure strategies Σ_Θ consists of the $\binom{n}{m}$ indicator vectors of this form. The set of possible mixed strategies for the normal-form game is Δ_Θ, defined by valid probability distributions over Σ_Θ.

Now, let P_E be the polyhedron defined by the solution space of the ERASER coverage vector. We show that all extreme points of P_E are in Δ_Θ, which implies that P_E is a subset of the polyhedron defined over Δ_Θ. The extreme points of P_E are defined by n linearly independent equality constraints. Since they have to satisfy $\sum_{i=1}^{n} c_i = m$, $n-1$ of the constraints $0 \le c_i \le 1$ must be tight, so $n-1$ of the c_i variables are either 0 or 1. Since m is an integer, the other variable must also be either 0 or 1. This implies that exactly m of the $c_i = 1$ and the rest of $c_i = 0$ for any extreme point of P_E. This c vector is therefore one of the pure strategies σ_Θ that define the extreme points of Δ_Θ, proving the inclusion. We can similarly argue the other direction, proving equivalence of the feasibility sets. If ERASER has a valid solution, we will be able to find a corresponding mixed strategy.

Theorem 8.2. *A pair of attack and coverage vectors (C, A) is optimal for the ERASER MILP correspond to at least one SSE of the game.*

Proof. We claim above that C corresponds to a mixed strategy for the defender, but A is an incomplete description of the attacker's Stackelberg strategy F_Ψ; it does not specify choices for any coverage other than C. Here, we show that the conditions of the MILP imply the existence of a function F_Ψ extending A such that C and F_Ψ satisfy the conditions of SSE given in Definition 8.1. We have already shown above that C and A are mutual best-responses for an optimal MILP solution. It remains to describe the attacker's behavior off the equilibrium path, for any other feasible coverage vectors $C' \neq C$. Let $t^* \in \Gamma(C')$

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

be a target in the attack set for C' with maximal payoff for the defender, and let A' be the attack vector that places probability 1 on t^*. By construction, A' is feasible in the MILP and satisfies conditions 2 and 3 for a SSE. Since (C', A') is a feasible solution in the MILP, $U_\Theta(C', A') \leq U_\Theta(C, A)$ since (C, A) is optimal for the MILP. Let F_Ψ be a function constructed using this method for every possible $C' \neq C$. Then C is a best-response to F_Ψ since $U_\Theta(C', A') \leq U_\Theta(C, A)$, satisfying condition 1 of the SSE. ■

8.5 Exploiting Payoff Structure

We now consider a class of security games in which the defender always benefits by having additional resources covering an attacked target, while the attacker is always worse off attacking a more heavily defended target. These assumptions are quite reasonable in many security games. Formally, we restrict payoff functions so that $U^u_\Theta(t) < U^c_\Theta(t)$ and $U^u_\Psi(t) > U^c_\Psi(t)$ for all t (note the strict inequalities). This is similar in spirit to a zero-sum assumption, but somewhat less restrictive. It is well-known that zero-sum games often admit more efficient solution algorithms, such as Luce's polynomial method for two-player, zero-sum games (Luce and Raiffa, 1989). We introduce two algorithms that compute extremely fast solutions for security games with this restriction on payoffs by exploiting structural properties of the optimal solution. We begin with three observations about the properties of the optimal solution for this class of games.

Observation 8.1. *All else equal, increasing c_t for any target not in $\Gamma(C)$ has no effect on $\hat{U}_\Theta(C)$ or $\hat{U}_\Psi(C)$.*

Increasing c_t can only decrease $U_\Psi(t, C)$ (due to the payoff assumption), and cannot affect the payoffs for any other target. Since t was not in $\Gamma(C)$ before, decreasing the payoff cannot result in a change to $\Gamma(C)$, and therefore cannot influence the SSE payoffs.

Observation 8.2. *If $\Gamma(C) \subset \Gamma(C')$ and $c_t = c'_t$ for all $t \in \Gamma(C)$ then $\hat{U}_\Theta(C) \leq \hat{U}_\Theta(C')$.*

In other words, adding an additional target to the attack set cannot hurt the defender. This is a straightforward consequence of the SSE assumption that the defender receives the optimal payoff among targets in the attack set.

Observation 8.3. *If $\hat{U}_\Psi(C) = x$, then $c_t \geq \frac{x - U^u_\Psi(t)}{U^c_\Psi(t) - U^u_\Psi(t)}$ for every target t with $U^u_\Psi(t) > x$.*

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

The inequality comes from setting the expected payoff for the target equal to the payoff for targets in the attack set: $x = c_t(U^c_\Psi) + (1 - c_t)U^u_\Psi$. Solving for c_t gives the coverage probability necessary to induce indifference between attacking t and any target in the attack set. If this condition is not satisfied for some t with $U^u_\Psi(t) > x$, then the attacker strictly prefers an attack on t instead of the attack set, contradicting the definition of the attack set (or x).

Algorithm 1 ORIGAMI

```
targets ← T sorted by U^u_Ψ(t)
payoff[t] ← U^u_Ψ(t), coverage[t] ← 0
left ← m, next ← 2
covBound ← −∞
while next ≤ n do
  addedCov[t] ← (payoff[next] − U^u_Ψ(t)) / (U^c_Ψ(t) − U^u_Ψ(t)) - coverage[t]
  if coverage[t] + addedCov[t] ≥ 1 then
    covBound ← Max(covBound, U^c_Ψ(t))
  end if
  if covBound ≥ −∞ OR Σ_{t∈T} addedCov[t] ≤ left then
    BREAK
  end if
  coverage[t] += addedCov[t]
  left -= Σ_{t∈T} addedCov[t]
  next++
end while
ratio[t] ← 1 / (U^u_Ψ(t) − U^c_Ψ(t))
coverage[t] += ratio[t]·left / Σ_{t∈T} ratio[t]
ifcoverage[t] ≥ 1 then
  covBound ← Max(covBound, U^c_Ψ(t))
end if
if covBound ≥ −∞ then
  coverage[t] ← (covBound − U^u_Ψ(t)) / (U^c_Ψ(t) − U^u_Ψ(t))
end if
```

We exploit these observations in the ORIGAMI (Optimizing Resources in Games Using Maximal Indifference) algorithm, which we present pseudocode for in Algorithm 8.5. The idea is to directly compute the attack set for the attacker, using the indifference equation in Observation 8.3. Starting with a target that has maximal $U^u_\Psi(t)$, the attack set is expanded at each iteration in order of decreasing $U^u_\Psi(t)$.[2] Each time the attack set is expanded, the coverage

[2] It is not strictly necessary to start from the maximal value and expand the set in order. A faster but slightly more complicated variation of the algorithm could be implemented using a binary

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

of each target is updates to maintain the indifference of attacker payoffs within the attack set.

There are two termination conditions. The first occurs when adding the next target to the attack set requires more total coverage probability than the defender has resources available. At this point, the size of the attack set cannot be increased further, but additional probability can still be added to the targets in the attack set in the specific ratio necessary to maintain indifference. The second termination condition occurs when any target t is covered with probability 1. The expected value for an attack on this target cannot be reduced below $U^c_\Psi(t)$, so this defines the final expected payoffs for the attack set. The final coverage probabilities are computed setting the coverages so that as many targets as possible have an expected payoff of $U^c_\Psi(t)$. In both cases, the solution maximizes the number of targets in the final attack set. Within the attack set, it maximizes the total coverage probability assigned while maintaining the attacker's indifference between the targets. The coverage probability for all targets outside of the attack set is 0. We show below that these properties suffice to identify a coverage vector that is an SSE of the security game.

Theorem 8.3. *ORIGAMI computes a coverage vector C that is optimal for the ERASER MILP, and is therefore consistent with an SSE of the security game.*

Proof. Let (C, A) be an optimal solution for ERASER MILP and C' be a coverage vector generated by ORIGAMI. C' is feasible in the MILP by construction. We first show that A must attack a target in $\Gamma(C')$, or it violates the optimality constraint for the attacker. Suppose ORIGAMI terminates because a target t is assigned $c'_t = 1$. By construction, $t \in \Gamma(C')$, and $U^c_\Psi(t) \geq U^u_\Psi(t')$ for any t' outside of $\Gamma(C')$. Since $c'_t = 1$ it cannot be greater in any coverage vector, and $c'_{t'} = 0$ so it cannot be smaller. Therefore, t' cannot be part of $\Gamma(C)$ for any feasible C. Now, suppose ORIGAMI terminates because all resources are assigned. Since maximal coverage is assigned to targets in $\Gamma(C')$, in any coverage vector C, $c'_t \leq c_t$ for at least one, t, or C violates the constraint on total resources available. Now, let t' be any target not in $\Gamma(C')$. We know that $U_\Psi(t, C') > U_\Psi(t', C')$, and since $c'_t \leq c_t$, then $U_\Psi(t, C) > U_\Psi(t', C)$ and t' is not in $\Gamma(C)$.

Having established that $\Gamma(C) \subset \Gamma(C')$ for any feasible C, we now consider whether any feasible C can improve the defender's payoff for an attack within $\Gamma(C')$. By Observation 8.1, we need to consider only changes in coverage probability within $\Gamma(C')$. To improve the defenders payoff over $\hat{U}_\Theta(C')$,

search to find the attack set of maximal size that can be induced using the available coverage resources.

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

the coverage probability for the new attack target t must increase. Otherwise, $\hat{U}_\Theta(C')$ would already achieve at least the payoff for t. First, take the case where C' assigns maximal coverage. It is not possible to have $c_t > c'_t$ for some t without having $c_{t'} < c'_{t'}$ for at least one other t', since the sum of c'_t is already maximal. Since $U_\Psi(t, C') = U_\Psi(t', C')$, an attack on t' is strictly preferred and the target t with higher coverage is no longer in the attack set. Similarly, in the case where a target is assigned coverage probability 1, it is not possible to increase the coverage of that target. Increasing the coverage of any other target reduces the attacker's payoff and removes it from the attack set. ■

$$\min \quad k \tag{8.12}$$

$$\gamma_t \in \{0,1\} \quad \forall t \in T \tag{8.13}$$

$$c_t \in [0,1] \quad \forall t \in T \tag{8.14}$$

$$\sum_{t \in T} c_t \leq m \tag{8.15}$$

$$U_\Psi(t, C) \leq k \quad \forall t \in T \tag{8.16}$$

$$k - U_\Psi(t, C) \leq (1 - \gamma_t) \cdot Z \quad \forall t \in T \tag{8.17}$$

$$c_t \leq \gamma_t \quad \forall t \in T \tag{8.18}$$

We have also implemented an MILP that applies the same principles as ORIGAMI, which we call ORIGAMI-MILP. It is presented in Equations 8.12 through 8.18. The vector γ represents the attack set, and replaces A in ERASER. γ_t is 1 for targets in the attack set, and 0 for all other targets. ORIGAMI-MILP is similar to ERASER, but does not optimize the defender's payoff. Instead, it minimizes the attacker's payoff, and adds a constraint that restricts c_t for any t not in $\Gamma(C)$ to 0, consistent with Observation 8.1. This constraint forces the attack set to include the maximal number of targets.

Theorem 8.4. *ORIGAMI-MILP generates an optimal solution for the ERASER MILP.*

Proof sketch: ORIGAMI-MILP generates solutions with the same properties as ORIGAMI. In particular, no coverage probability is assigned to targets outside of the attack set. This implies that any target in the attack set is assigned exactly the coverage probability necessary to induce indifference with all other targets in the attack set, as in Observation 8.3. The objective of minimizing the attacker's payoff forces an attack set with the lowest expected payoff for the attacker. Any target with an uncovered payoff higher than this value must be included in the attack set, which forces the attack set to be maximal. This in turn maximizes the defender's SSE payoff.

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

8.6 Scheduling and Resource Constraints

We now introduce ERASER-C ("Constrained"), an extension of ERASER which adds the capability to represent certain kinds of resource and scheduling constraints, motivated by the real example domains described previously. We demonstrate that the basic idea of using a compact representation of the defender's strategy space and of the payoff functions for both players is still useful in this setting when resources are heterogeneous.

The first extension allows resources to be assigned to *schedules* covering multiple targets. The set of legal schedules $S = \{s_1 \dots s_l\}$ is a subset of the power set of the targets, with restrictions on this set representing scheduling constraints. We define the relationship between targets and schedules with the function $M : S \times T \rightarrow \{0,1\}$, which evaluates to 1 if and only if t is covered in s.[3] The defender's strategy is an assignment of resources to schedules, rather than targets. A second extension introduces *resource types*, $\Omega = \{\omega_1, \dots, \omega_v\}$, each with the capability to cover a different subset of S. The number of available resources of each type is given by the function $\mathcal{R}(\omega)$. Coverage capabilities for each type are given by the function $Ca : S \times \Omega \rightarrow \{0,1\}$, which is 1 if the type is able to cover the given schedule, and 0 otherwise.[4]

The combination of schedules and resource types captures key elements of the FAMS domain. Suppose we model air marshals as resources, flights as targets, and defined payoffs by expert risk analysis. A single marshal cannot be on all possible flights due to location and timing constraints. We could use legal schedules to define the set of feasible flights for a particular air marshal. Resource types can be used to specify different sets of legal schedules for each resource (e.g., based on initial location). Adding these constraints effectively reduces the space of feasible coverage vectors. Consider an example with a single resource defending three targets. There are two legal schedules, covering targets $\{1,2\}$ and $\{2,3\}$. Given only these schedules, it is not possible to implement a coverage vector that places 50% probability on both targets 1 and 3, with no coverage of target 2.

An MILP implementing ERASER-C is presented in Equations 8.19 through 8.30. The MILP is very similar to the original ERASER, but it enforces additional constraints on the legal schedules and coverage for each resource type. The q variables represent the total probability assigned to each schedule by all resource types, and the h variables are the probability assigned to a schedule

[3] For the purposes of the FAMS domain and the version of ERASER-C presented here, all schedules are of size 2 and there are no odd cycles in the graph where targets are vertices and edges are schedules.

[4] Our implementation uses complete matrices for M and Ca, but sparse representations could improve performance.

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

by a specific type of resource. In Equations 8.29 and 8.30, Z is a large constant relative to the maximum payoff.

Constraint 8.20 restricts the attack vector to binary variables, which correspond to pure strategies for the attacker. Constraints 8.21 through 8.23 restrict the defender's strategy so that no target is assigned probability greater than 1. The coverage of each schedule must sum to the contributions of the individual resource types, specified in Equation 8.25. The mapping between the coverage of schedules and coverage of targets is enforced in Constraint 8.26. Constraint 8.27 restricts the schedule so that only the available number of resources of each type are used. No probability may be assigned to disallowed schedules for each resource type, which is explicitly enforced by Constraint 8.28. The final three constraints specify the maximization performed by both the attacker and defender, exactly as in the ERASER (see Section 8.4.3).

$$\max \quad d \tag{8.19}$$

$$a_t \in \{0,1\} \quad \forall t \in T \tag{8.20}$$

$$c_t \in [0,1] \quad \forall t \in T \tag{8.21}$$

$$q_s \in [0,1] \quad \forall s \in S \tag{8.22}$$

$$h_{s,\omega} \in [0,1] \quad \forall s,\omega \in S \times \Omega \tag{8.23}$$

$$\sum_{t \in T} a_t = 1 \tag{8.24}$$

$$\sum_{\omega \in \Omega} h_{s,\omega} = q_s \quad \forall s \in S \tag{8.25}$$

$$\sum_{s \in S} q_s M(s,t) = c_t \quad \forall t \in T \tag{8.26}$$

$$\sum_{s \in S} h_{s,\omega} Ca(s,\omega) \leq \mathcal{R}(\omega) \quad \forall \omega \in \Omega \tag{8.27}$$

$$h_{s,\omega} \leq Ca(s,\omega) \quad \forall s,\omega \in S \times \Omega \tag{8.28}$$

$$d - U_\Theta(t,C) \leq (1 - a_t) \cdot Z \; \forall t \in T \tag{8.29}$$

$$0 \leq k - U_\Psi(t,C) \leq (1 - a_t) \cdot Z \; \forall t \in T \tag{8.30}$$

An optimal coverage vector of ERASER-C meets the equilibrium conditions, following the same line of reasoning as Theorem 8.2. However, the MILP as written can result in coverage vectors that cannot be implemented by mixed strategies in the original solution space if arbitrary schedules are allowed (as noted above, in the FAMS domain schedules have a restricted form). In particular, additional constraints are necessary if odd cycles are possible in the

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

schedules. We defer full analysis and discussion of this issue to future work, but note that in empirical testing with realistic data even simple heuristic methods are able to generate sample joint assignments that closely approximate the optimal coverage probabilities identified by this MILP.

8.7 Experimental Evaluation

We evaluate the four algorithms using both randomly generated security games and real examples from the LAX and FAMS domains. Our baseline for comparing with existing methods is DOBSS (Paruchuri et al., 2008), which is the fastest known algorithm for general Bayesian Stackelberg games. While we do not consider the Bayesian case here, DOBSS is also comparable to other methods (notably [189]) for the non-Bayesian case. All of our algorithms generate optimal SSE solutions, so the primary metrics of comparison are the computational requirements to compute solutions, in both time and memory. We note that the algorithms are applicable to different classes of games, with faster algorithms generally able to solve a smaller class of games. The ordering of the algorithms in terms of the size of the class of games is given by ORIGAMI/ORIGAMI-MILP $\subset$ ERASER $\subset$ ERASER-C $\subset$ DOBSS.

All of our experiments were run on a machine with dual Xeon 3.2Ghz processors and 2GB of RAM, running RHEL 3. We use CPLEX 9.0.0 with default parameter settings to solve MILPs. All data points are based on twenty sample game instances, except when explicitly stated otherwise. Our first set of experiments compares the performance of DOBSS, ERASER, and ERASER-C on random game instances. We next compare ERASER, ORIGAMI, and ORIGAMI-MILP on much larger instances that DOBSS is unable to solve, given memory limitations. The final experiment compares the algorithms on relevant example games for the LAX and FAMS domains described in Section 8.3.

For the first set of tests we generate random instances of compact security games of the form used by ERASER (see Section 8.4.1). To generate a game with a given number of targets and resources, we independently randomly draw four integer payoffs for each target. Thus, U^c_Θ and U^u_Ψ are drawn from $Uniform[0,100]$, while U^u_Θ and U^c_Ψ are drawn from $Uniform[-100,0]$. We use a value of five resources for this set of results and vary the number of targets. The generated game instances are translated into the representations used by DOBSS and ERASER-C.

Figure 8.1a compares runtime performance of DOBSS, ERASER and ERASER-C on this set of games. The x-axis is the size of the game (in targets),

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

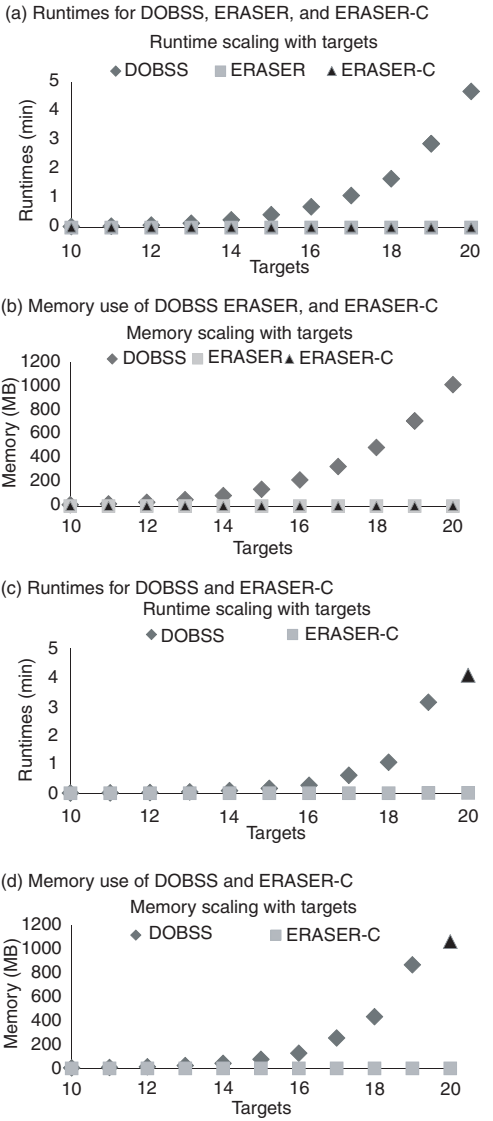

Figure 8.1. Runtime and memory scaling.

and the y-axis is runtime in minutes. For example, the point (20, 4.68) indicates that DOBSS has an average runtime of 4.68 m on problems with twenty targets; ERASER and ERASER-C each run for 0.002s for twenty targets. The data show an exponential increase in time necessary to compute DOBSS solutions,

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

and essentially no change in runtimes for ERASER and ERASER-C up to 20 targets. The differences between both variants of ERASER and DOBSS are statistically significant (using Yuen's test) for the larger games, while there is no significant difference between ERASER and ERASER-C for games of this size. Figure 8.1b compares the memory performance on the same set of games; there the y-axis represents the memory usage of AMPL in MB. Runtime performance roughly tracks memory performance, and the same exponential behavior is observed for DOBSS. Memory limitations become prohibitive for DOBSS before runtimes become extremely long (though the growth trend is already clear); we were unable to successfully complete game instances beyond roughly 1GB on the memory measure.

We also compare the performance of ERASER-C and DOBSS on games that require the additional capabilities of ERASER-C; ERASER is not included in this test because it cannot solve the relevant games. Our random game instances now include schedules, resource types, and the schedule and coverage mappings, as described in Section 8.6. We test games with three resource types, and availability of $[3, 3, 2]$ for each type. There are twice as many schedules as targets, and each schedule covers a randomly selected set of two targets (we also ensure that each target is covered in at least one schedule). Each resource type covers approximately 33% of the legal schedules, again selected randomly.

Figures 8.1c and 8.1d compare the performance of DOBSS and ERASER-C on this set of games, using the same metrics as the previous set of results. DOBSS was unable to complete all games with twenty targets due to memory limitations; the black triangle gives the result for the three complete trials (which are likely biased low). All comparisons are statistically significant for large games. We observe the same patterns of performance for both DOBSS and ERASER-C for this set of games as in the more restricted class solvable by ERASER. ERASER-C adds representational power, and retains substantial performance improvements over the baseline DOBSS algorithm.

We now compare the performance of ERASER, ORIGAMI, and ORIGAMI-MILP on very large games well beyond the limits of DOBSS. Random game instances are generated as before for the experiment including ERASER; the random payoffs generated already meet the restrictions of the ORIGAMI algorithms. Figure 8.2 compares the runtimes of the three algorithms on games with 25 resources and up to 3000 targets. Figure 8.2b extends the data out to 1,000 resources and 40,000 targets for the two ORIGAMI algorithms. In both figures, the x-axis is the number of targets, and the y-axis is runtime, as before.

The ERASER algorithm was able to solve games of 3000 targets in 13.30 minutes, which is quite impressive. The ORIGAMI algorithms were even

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

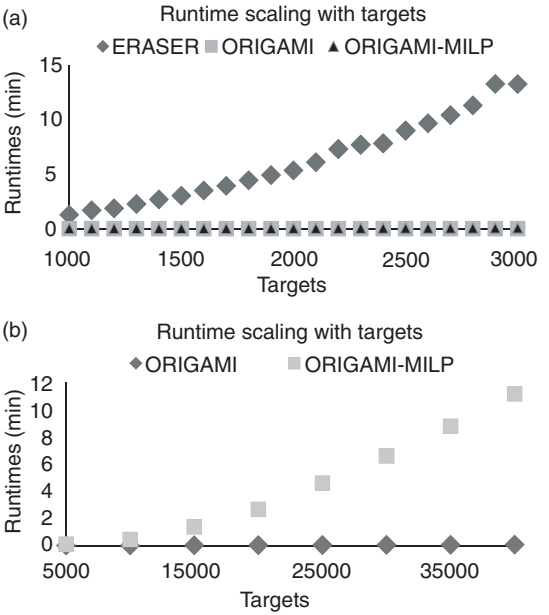

Figure 8.2. Runtime scaling of ERASER, ORIGAMI, and ORIGAMI-MILP.

more impressive, solving these games in seconds. Yuen's test confirms that the ORIGAMI times for large games are significantly different than the ERASER times. As a further point of comparison, for the 3000 target, 25 resource game, the size of the defender's strategy space in normal form is roughly 10^{60} – clearly infeasible to represent, let alone solve. The results for the two ORIGAMI algorithms given in Figure 8.2b show that as the size of the game scales to very large instances, ORIGAMI outperforms ORIGAMI-MILP. To test the ultimate scalability of ORIGAMI, we ran a single trial of a game with 1,000,000 targets and 10,000 resources. ORIGAMI successfully computed a solution in 1.51 hours.

In the final set of experiments we test the algorithms on real data acquired for both the LAX canine and FAMS scheduling domains. The LAX data consists of a single game instance with six resources (canine units) and eight targets (terminals). For FAMS we generated two hypothetical examples using real (public) flight information and hypothetical information about resources and coverage capabilities, and hypothetical payoffs. Both examples cover a one week period, but they cover different foreign and domestic airports to generate "small" and "large" tests.

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

Table 8.2. *Runtimes on real data*

	Actions	DOBSS	ERASER (-C)
LAX (6 canines)	784	0.94s	0.23s
FAMS (small)	∼6,000	4.74s	0.09s
FAMS (large)	∼85,000	435.6s*	1.57s

Table 8.2 shows runtimes and DOBSS action space size for these problem instances, averaged over twenty trials of the same game instance. The first line compares DOBSS and ERASER for the LAX security domain. The second and third lines compare DOBSS and ERASER-C on the two instances of the FAMS domain (ERASER is not capable of representing these problems). DOBSS did not complete the large problem instance, reaching a memory limit after 435.6s of runtime. All differences in means are statistically significant (though we emphasize that these are repeated trials on the same game instance). Our algorithms show dramatic performance improvements over DOBSS on these real data sets, in addition to the randomly generated data sets presented previously.

8.8 Conclusion

Allocating limited resources is an important problem in many security domains. Airport security, the federal air marshals, screening incoming shipments at ports, patrolling subway systems, and random checks at customs are all examples of this. Increasingly, game-theoretic analysis is seen as a valuable tool for analyzing these problems, and especially for determining effective randomization strategies. We apply the theory of Stackelberg games to this problem, following the successful application of similar technology at the LAX airport by Pita et al. (2008).

We contribute new algorithms for computing optimal solutions to security games that scale to massive games with many resources and many targets. While the best existing algorithm was unable to solve games larger than twenty targets because of memory limitations, our algorithms scale to thousands – and in some cases *millions* – of targets. The first method, ERASER, introduces a compact representation for security games with multiple resources, avoiding a combinatorial explosion in representation size of the normal-form game. We present two algorithm which offer even more dramatic performance improvements for a class of games with plausible restrictions on the payoff functions for the players. These algorithms exploit structural properties of optimal solutions under

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.

the payoff restrictions. Finally, we extend the ERASER algorithm to incorporate scheduling and resource constraints motivated by the FAMS domain. The resulting ERASER-C algorithm is more expressive than ERASER, but still improves performance over the existing baseline methods. Together, these four algorithms offer a powerful set of computational tools for solving massive security games. In many cases, they offer improvements of several orders of magnitude in computational scalability.

Acknowledgment

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. We are also grateful to the United States Federal Air Marshal Service for their exceptional collaboration. However, any opinions, conclusions or recommendations herein are solely those of the authors and do not necessarily reflect views of the Department of Homeland Security or Federal Air Marshal Service.

Copyright © 2011. Cambridge University Press. All rights reserved.

Tambe, M. (2011). Security and game theory : Algorithms, deployed systems, lessons learned. Retrieved from http://ebookcentral.proquest.com
Created from binghamton on 2019-12-10 04:51:15.