# Chapter 3
# Deployed Security Games for Patrol Planning

**Fernando Ordóñez, Milind Tambe, Juan F. Jara, Manish Jain, Christopher Kiekintveld, and Jason Tsai**

**Abstract** Nations and organizations need to secure locations of economic, military, or political importance from groups or individuals that can cause harm. The fact that there are limited security resources prevents complete security coverage, which allows adversaries to observe and exploit patterns in patrolling or monitoring and enables them to plan attacks that avoid existing patrols. The use of randomized security policies that are more difficult for adversaries to predict and exploit can counter their surveillance capabilities and improve security. In this chapter we describe the recent development of models to assist security forces in randomizing their patrols and their deployment in real applications. The systems deployed are based on fast algorithms for solving large instances of Bayesian Stackelberg games that capture the interaction between security forces and adversaries. Here we describe a generic mathematical formulation of these models, present some of the results that have allowed these systems to be deployed in practice, and outline remaining future challenges. We discuss the deployment of these systems in two real-world security applications: (1) The police at the Los Angeles International Airport uses these models to randomize the placement of checkpoints on roads entering the airport and the routes of canine unit patrols within the airport terminals. (2) The Federal Air Marshal Service (FAMS) uses these models to randomize the schedules of air marshals on international flights.

F. Ordóñez (✉) • J.F. Jara
Industrial Engineering Department, University of Chile, Republica 701, Santiago, Chile
e-mail: fordon@dii.uchile.cl

M. Tambe • M. Jain • J. Tsai
Computer Science Department, University of Southern California, Los Angeles, CA 90089, USA
e-mail: milind@usc.edu; manishja@usc.edu; jasontts@usc.edu

C. Kiekintveld
Computer Science Department, University of Texas, El Paso, TX 79968, USA
e-mail: cdkiekintveld@utep.edu

J.W. Herrmann (ed.), *Handbook of Operations Research for Homeland Security*,
International Series in Operations Research & Management Science 183,
DOI 10.1007/978-1-4614-5278-2_3, © Springer Science+Business Media New York 2013

## 3.1 Introduction

Nations and organizations need to secure locations of economic, military, or political importance from groups or individuals that can cause harm. Protecting such critical sites and targets, such as airports, historical landmarks, power generation facilities, and political figures, is a challenging task for police and security agencies worldwide. The growing threat of international terrorism has exacerbated this challenge in recent years. For instance, transportation networks such as buses, trains, and airplanes carry millions of people per day to their destinations, making them a prime target for terrorists and extremely difficult to protect for law enforcement agencies. The September 11, 2001 attack on the World Trade Center in New York City via commercial airliners resulted in $27.2 billion of direct short-term costs (Looney, 2002) as well as a government-reported 2,974 lives lost. The 2004 Madrid commuter train bombings resulted in 191 lives lost, 1,755 wounded, and an estimated cost of 212 million Euros (Blanco et al., 2007). Finally, in the 2005 London subway and bus bombings, 52 lives were lost, 700 were wounded, and there was an estimated economic cost of two billion pounds (Thornton, 2005).

Measures for protecting potential target areas include monitoring entrances or inbound roads and patrolling the network at transfer points and aboard transportation vehicles. However, limited resources imply that it is typically impossible to provide full security coverage at all times. Furthermore, adversaries can observe security arrangements over time and exploit any predictable patterns to their advantage. One way to mitigate the ability of adversaries to exploit patterns is the judicious use of randomization in scheduling the actions of security forces. For example, police patrols, baggage screenings, vehicle checkpoints, and other security procedures are often randomized. However, security forces face many difficulties in effectively randomizing their operations. One of these difficulties is how to weigh the different actions the defender could take. A strategy in which all targets are equally likely to be defended fails to take into account that some targets are more attractive or vulnerable than others. A defense strategy that weighs the protection of each target against the value of that target still fails to account for the possibility that the attacker is intelligent and will update their strategy based on the actions of the defender. Asking a human to generate a random security policy has additional difficulties as humans are not good at generating truly random behavior (Wagenaar, 1972; Treisman and Faulkner, 1987) and can easily fall into predictable patterns. Furthermore, in transportation networks and many other security domains, the problem of scheduling security forces is prohibitively large, even without considering randomization. Creating a schedule by hand is a costly and labor-intensive process.

Our work on randomized patrol planning has lead to a number of deployed software assistants that address many of these key difficulties of randomization and provide an easy-to-use solution for security forces. These assistants use game-theoretic models and solution algorithms to determine good randomization strategies that take into account target values and assume intelligent adversary responses to security measures. Game theory is a well-established paradigm for reasoning about situations with multiple self-interested decision makers (Fudenberg and Tirole, 1991). We model security games as Stackelberg games (von Stackelberg, 1934) between the defender (i.e., the security forces) and the attacker (i.e., a terrorist adversary). Stackelberg games are a bilevel model (Bard, 1999) that account for the ability of an attacker to gather information about the defense strategy before planning an attack. These games specify different payoff values for both players in the event of an attack on every potential target. Extending these games to Bayesian Stackelberg games (Conitzer and Sandholm, 2006) allows us to capture uncertainty about these payoffs in the game model. Solutions to these games provide a randomized policy for the defense strategy, which can be used to generate specific schedules for security patrols.

In this chapter we describe how we applied this game-theoretic approach in two different software solutions that provide assistance in scheduling real security operations. The ARMOR program (Pita et al., 2008), developed for the Los Angeles international airport (LAX) police, randomizes checkpoints on the roadways entering the airport and canine patrol routes within the airport terminals. The IRIS program (Tsai et al., 2009) was developed for the Federal Air Marshal Service (FAMS) to assist with randomly scheduling air marshals on flights. These software assistants are interactive, and domain experts can change domain parameters when necessary. Underlying each of these tools is a model of the domain as a Bayesian Stackelberg game, along with fast solution algorithms for computing an optimal solution to the game model. These algorithms use various techniques for exploiting structure in the security domains to speed up the computation and enable large real-world problem instances to be solved in reasonable amounts of time (Paruchuri et al., 2006, 2007; Kiekintveld et al., 2009). As we highlight later in this chapter, developing these assistants requires a substantial amount of work in calibrating the Stackelberg game model to capture expert's knowledge of the security domain. This is critical so that the defender strategies proposed are reasonable and useful. Clustering and data-mining methods can help in formulating a representative security game in situations where there is sufficient information of events.

The rest of the chapter is organized as follows. Related work is discussed in Sect. 3.2. The Bayesian Stackelberg security game models, technical formulation, and solution algorithms are discussed in Sect. 3.3. The LAX and FAMS domains and the software assistants developed are described in Sect. 3.4. In Sect. 3.5, we illustrate how to use clustering methods to automatically build a Stackelberg security game for a network patrolling problem. We present our conclusions in Sect. 3.6. This chapter is based on our previous work (Paruchuri et al., 2007; Kiekintveld et al., 2009; Jain et al., 2010) and extends it by describing a data-driven process to build a Stackelberg security game.

## 3.2 Related Work

There are three main areas of related work that we review here: Optimization techniques for patrol planning that do not take the strategic behavior of adversaries into account, Stackelberg game models used in diverse security problems, and other game-theoretic models used for security.

The first area of related work applies optimization techniques that model a security domain but do not address the strategic aspects of the problem. These methods provide a randomization strategy for the defender, but they do not take into account the fact that the adversaries can observe the defender's actions and then adjust their behavior. Examples of such authors or approaches include Ruan et al. (2005) and Paruchuri et al. (2006), which are based on learning, Markov decision processes (MDPs) and partially observable Markov decision processes (POMDPs). As part of this work, the authors model the patrolling problem with locations and varying incident rates in each of the locations and solve for optimal routes using a MDP framework. Another example is the "Hypercube Queueing Model" (Larson, 1974) which is based on queueing theory and depicts the detailed spatial operation of urban police departments and emergency medical services. It has found application in police beat design, in allocation of patrolling time, etc. Such frameworks can address many of the problems we raise, including different target values and increasing uncertainty by using many possible patrol routes. However, they fail to account for the possibility that an intelligent attacker will observe and exploit patterns in the security policy. If a policy is based on the historical frequency of attacks, it is essentially a reactive policy and an intelligent attacker will always be one step ahead.

A second area of work uses Stackelberg games to model a variety of security domains. Bier (2007) give a strong endorsement of this type of modeling for security problems. Game-theoretic models have been applied in a variety of homeland security settings, such as protecting critical infrastructure (Brown et al., 2006; Pita et al., 2008; Nie et al., 2007). Wein (2009) apply Stackelberg games in the context of screening visitors entering the USA. In their work, they model the US Government as the leader who specifies the biometric identification strategy to maximize the detection probability using finger print matches, and the follower is the terrorist who can manipulate the image quality of the finger print. They have also been used for studying missile defense systems (Brown et al., 2005) and for studying the development of an adversary's weapon systems (Brown et al., 2005). A family of Stackelberg games known as inspection games is closely related to the security games we are interested in and includes models of arms inspections and border patrols (Avenhaus et al., 2002). Other recent work uses Stackelberg games to obtain randomized patrolling in a generic "police and robbers" scenario (Gatti, 2008) and perimeter patrols (Agmon et al., 2008).

Our work belongs to this line of research, focusing on Stackelberg games for patrol planning. Our work differs from the previous work mainly in the solution approach used and the domain constraints considered, which have arisen from our work deploying these systems in the real world. In addition to the ARMOR and IRIS systems that will be discussed in detail in this chapter, we are currently working on designing new game-theoretic scheduling assistants for other security agencies. For instance, GUARDS (Pita et al., 2011) is a system for scheduling activities being developed for the Transportation Security Administration. GUARDS is being evaluated at an undisclosed airport for potential nationwide deployment. Finally, PROTECT (An et al., 2011) is in use for scheduling the patrols of the United States Coast Guard in the port of Boston; it is currently being deployed in the port of New York and may be deployed at multiple other ports in the USA.

The third area of related work is the application of game-theoretic techniques that are not based on Stackelberg games to security applications. Security problems are increasingly studied using game-theoretic analysis, ranging from computer network security (Lye and Wing, 2005; Srivastava et al., 2005) to terrorism (Sandler and Arce, 2003). Babu et al. (2006) have worked on modeling passenger security system at US airports using linear programming approaches; however, their objective is to classify the passengers in various groups and then screen them based on the group to which they belong.

## 3.3 Methodology

A generic Stackelberg game has two players, a *leader* and a *follower*. These players need not represent individuals but could also be groups that cooperate to execute a joint strategy, such as a police force or terrorist organization. For the modeling of security, the leadership role is assumed by the police, and the role of follower by criminals. Decisions made by each player are where to protect and where to attack respectively. Thus, having police act first reflects the fact that patrols conducted by police officers are observable by criminals and, in the long run, the latter are able to estimate the probability of encountering police in a given sector. Thus, the decision of offenders is carried out once the likelihood of facing the police is observed.

The actions for the security forces represent the action of scheduling a patrol or security procedure to protect a set of targets, e.g., a checkpoint at the LAX airport or assigning federal air marshals to a flight. The actions for an adversary represent possible attacks at one of the targets being protected, e.g., a terminal at LAX or a certain flight.

### *3.3.1 Stackelberg Equilibrium*

In a Stackelberg game each player has a set of possible *pure strategies*, denoted $\sigma_d \in \Sigma_d$ and $\sigma_a \in \Sigma_a$. A *mixed strategy* allows a player to play a probability distribution over pure strategies, denoted $\delta_d \in \Delta_d$ and $\delta_a \in \Delta_a$. Payoffs for each player are defined over all possible joint pure-strategy outcomes: $\Omega_d : \Sigma_a \times \Sigma_d \to \mathscr{R}$ for the defender and similarly for each attacker. The payoff functions are extended to mixed strategies in the standard way by taking the expectation over pure-strategy outcomes. The follower can observe the leader's strategy and then act in a way to optimize its own payoffs. Formally, the attacker's strategy in a Stackelberg security game becomes a function that selects a strategy for each possible leader strategy: $F_{\mathrm{a}} : \Delta_{\mathrm{d}} \to \Delta_{\mathrm{a}}$.

The most common solution concept in game theory is a *Nash equilibrium*, which is a profile of strategies for each player in which no player can gain by unilaterally changing to another strategy (Osbourne and Rubinstein, 1994). Stackelberg equilibrium is a refinement of Nash equilibrium specific to Stackelberg games. It is a form of subgame perfect equilibrium in which it requires that each player select the best-response in any subgame of the original game (where subgames correspond to partial sequences of actions). The effect is to eliminate equilibrium profiles that are supported by non-credible threats off the equilibrium path. Subgame perfection is a natural requirement, but it does not guarantee a unique solution in cases where the follower is indifferent among a set of strategies. The literature contains two forms of Stackelberg equilibria that identify unique outcomes, 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. Unlike the weak form, strong Stackelberg equilibria are known to exist in all Stackelberg games (Basar and Olsder, 1995). A standard argument suggests that the leader is often able to induce the favorable strong form by selecting a strategy arbitrarily close to the equilibrium which causes the follower to strictly prefer the desired strategy (von Stengel and Zamir, 2004). We adopt strong Stackelberg equilibrium (SSE) as our solution concept in part for these reasons but also because it is the most commonly used in related literature (Osbourne and Rubinstein, 1994; Conitzer and Sandholm, 2006; Paruchuri et al., 2008).

**Definition 1** A set of strategies ($\delta_{\mathrm{d}}$, $F_{\mathrm{a}}$) form a *SSE* if they satisfy the following:

1. The leader plays a best-response:
   $\Omega_{\mathrm{d}}(\delta_{\mathrm{d}}, F_{\mathrm{a}}(\delta_{\mathrm{d}})) \geq \Omega_{\mathrm{d}}(\delta_{\mathrm{d}}', F_{\mathrm{a}}(\delta_{\mathrm{d}}'))\ \forall \delta_{\mathrm{d}}' \in \Delta_{\mathrm{d}}$.
2. The follower plays a best-response:
   $\Omega_{\mathrm{a}}(\delta_{\mathrm{d}}, F_{\mathrm{a}}(\delta_{\mathrm{d}})) \geq \Omega_{\mathrm{a}}(\delta_{\mathrm{d}}, \delta_{\mathrm{a}})\ \forall \delta_{\mathrm{d}} \in \Delta_{\mathrm{d}}, \delta_{\mathrm{a}} \in \Delta_{\mathrm{a}}$.
3. The follower breaks ties optimally for the leader:
   $\Omega_{\mathrm{d}}(\delta_{\mathrm{d}}, F_{\mathrm{d}}(\delta_{\mathrm{d}})) \geq \Omega_{\mathrm{d}}(\delta_{\mathrm{d}}, \delta_{\mathrm{a}})\ \forall \delta_{\mathrm{d}} \in \Delta_{\mathrm{d}}, \delta_{\mathrm{a}} \in \Delta_{\mathrm{a}}^{*}(\delta_{\mathrm{d}})$, where $\Delta_{\mathrm{a}}^{*}(\delta_{\mathrm{d}})$ is the set of follower best-responses, as above.

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, forcing commitment to a pure strategy would guarantee a loss. However, it has been shown that the ability to commit to a mixed strategy always weakly increases the leader's payoffs in equilibrium profiles of the game (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.

The Bayesian extension to the Stackelberg game allows for multiple types of players, with each type associated with its own payoff values. For the security games of interest in this chapter, we assume that there is only one leader type (e.g., only one police force), although there are multiple follower types (e.g., multiple adversary types trying to infiltrate security). The set of follower types is denoted by $\Gamma$. Each type $\gamma$ is represented by a different payoff matrix. The leader does not know the follower's type. The goal is to *find the optimal mixed strategy* for the leader to commit to, given that each follower type will know the mixed strategy of the leader when choosing its own strategy. Payoffs for each type are defined over all possible joint pure-strategy outcomes: $\Omega_d : \Sigma_a^\Gamma \times \Sigma_d \rightarrow \mathscr{R}$ for the defender and similarly for each attacker type. The leader's best response is now a weighted best response to the followers' responses, where the weights are based on the probability of occurrence of each type. The strategy of each attacker type $\gamma$ becomes: $F_{\mathrm{a}}^{\gamma} : \Delta_{\mathrm{d}} \rightarrow \Delta_{\mathrm{a}}^{\gamma}$, which still satisfies constraints 2 and 3 in Definition 1.

### 3.3.2 Security Game Representation

There are two major problems with using conventional methods to represent security games in normal form. First, many solution methods require the use of a Harsanyi transformation when dealing with Bayesian games (Harsanyi and Selten, 1972). The Harsanyi transformation converts a Bayesian game into a normal-form game, but the new game may be exponentially larger than the original Bayesian game. Our compact representation avoids this Harsanyi transformation, and instead we directly operate on the Bayesian game. Operating directly on the Bayesian representation is possible in our model because the evaluation of the leader strategy against a Harsanyi-transformed game matrix is equivalent to its evaluation against each of the game matrices for the individual follower types. (For more details, see the Appendix; a further detailed explanation appears in Paruchuri et al. (2008)). The second problem arises because the defender has many possible resources to schedule in the security policy. This can also lead to a combinatorial explosion in a standard normal-form representation. For example, if the leader has $m$ resources to defend $n$ entities, then normal-form representations model this problem as a single leader with $\binom{n}{m}$ rows, each row corresponding to a leader action of covering $m$ targets with security resources. However, in our compact representation, the game representation would only include $n$ rows, each row corresponding to whether the corresponding target was covered or not. Such a representation is equivalent to the normal form representation for the class of problems we address in this work (see Kiekintveld et al. 2009 for additional details). This compactness in

**Table 3.1** Example payoffs for an attack on a target

| | Covered | Uncovered |
|---|---|---|
| Defender | 5 | −20 |
| Attacker | −10 | 30 |

our representation is possible because the payoffs for the leader in these games simply depend on whether the attacked target was covered or not, and not on what other targets were covered (or not covered). The representation we use here avoids both of these potential problems, using methods similar to other compact representations for games (Koller and Milch, 2003; Jiang and Leyton-Brown, 2006).

We now introduce our compact representation for security games. 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 resources could be federal air marshals). Associated with each target are four payoffs defining the possible outcomes for an attack on the target, as shown in Table 3.1. There are two cases, depending on whether or not the target is covered by the defender. The defender's payoff for an uncovered attack when facing an adversary of type $\gamma$ is denoted $U_{\mathrm{d}}^{\gamma,\,u}(t)$, and for a covered attack $U_{\mathrm{d}}^{\gamma,\,c}(t)$. Similarly, $U_{\mathrm{a}}^{\gamma,\,u}(t)$ and $U_{\mathrm{a}}^{\gamma,\,c}(t)$ are the payoffs of the attacker.

A crucial feature of the model is that payoffs depend only on the target attacked, and whether or not it is covered by the defender. The payoffs do *not* depend on the remaining aspects of the schedule, such as whether any unattacked target is covered or which specific defense resource provides coverage. For example, if an adversary succeeds in attacking Terminal 1, the penalty for the defender is the same whether the defender was guarding Terminal 2 or 3. Therefore, from a payoff perspective, many resource allocations by the defender 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 $A^{\gamma}$ gives the probability of attacking a target by a follower of type $\gamma$. We restrict the attack vector for each follower type to attack a single target with probability 1. This is without loss of generality because a SSE solution still exists under this restriction (Paruchuri et al., 2008). Thus, the follower of type $\gamma$ can choose any pure strategy $\sigma_{\mathrm{a}}^{\gamma} \in \Sigma_{\mathrm{a}}^{\gamma}$, that is, attack any one target from the set of targets.

The payoff for a defender when a specific target $t$ is attacked by an adversary of type $\gamma$ is given by $U_{\mathrm{d}}^{\gamma}(t, C)$ and is defined in (3.1). Thus, the expectation of $U_{\mathrm{d}}^{\gamma}(t, C)$ over $t$ gives $U_{\mathrm{d}}^{\gamma}$, which is the defender's expected payoff given coverage vector $C$ when facing an adversary of type $\gamma$ whose attack vector is $A^{\gamma}$. $U_{\mathrm{d}}^{\gamma}$ is defined in (3.2). The same notation applies for each follower type, replacing "d" with "a." Thus, $U_{\mathrm{a}}^{\gamma}(t, C)$ gives the payoff to the attacker when a target $t$ is attacked by an adversary of type $\gamma$. We will see $U_{\mathrm{a}}^{\gamma}(t, C)$ and $U_{\mathrm{d}}^{\gamma}(t, C)$ used in the MILP discussed later. We also define the useful notion of the *attack set* in (3.3), $\Lambda^{\gamma}(C)$, which contains all targets that yield the maximum expected payoff for the attacker type $\gamma$ given coverage $C$. This *attack set* is used by the adversary to break ties when calculating a SSE. Moreover, in these security games, exactly one adversary is attacking in one instance of the game; however, the adversary could be of any type and the defender does not know the type of the adversary faced.

$$U_d^\gamma(t, C) = c_t U_d^{\gamma,c}(t) + (1 - c_t) U_d^{\gamma,u}(t) \tag{3.1}$$

$$U_d^\gamma(C, A^\gamma) = \sum\nolimits_{t \in T} a_t^\gamma \cdot (c_t \cdot U_d^{\gamma,c}(t) + (1 - c_t) U_d^{\gamma,u}(t)) \tag{3.2}$$

$$\Lambda^\gamma(C) = \{t : U_a^\gamma(t, C) \geq U_a^\gamma(t', C) \forall t' \in T\}. \tag{3.3}$$

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 when facing this adversary of type $\gamma$ with probability $p^\gamma$ is $\hat{U}_d^\gamma(C) = U_d^\gamma(t^*, C) \times p^\gamma$, and for the attacker $\hat{U}_a^\gamma(C) = U_a^\gamma(t^*, C)$.

### 3.3.3 Solution Method

We introduce the ERASER-C algorithm (Efficient Randomized Allocation of Security Resources with Constraints), which takes as input a security game in the compact form described in Sect. 3.3.2 and solves for an optimal coverage vector corresponding to a SSE strategy for the defender. We allow resources to be assigned to *schedules* covering multiple targets. The set of legal schedules $S = \{s_1, \ldots, s_l\}$ 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 $H : S \times T \rightarrow \{0, 1\}$, which evaluates to 1 if and only if $t$ is covered in $s$. The defender's strategy is now an assignment of resources to schedules, rather than targets. Another important notion is the presence of *resource types*, $\Omega = \{\omega_1, \ldots, \omega_v\}$, 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.[1]

The combination of schedules and resource types captures key elements of the security domains. For example, in FAMS, federal air marshals are resources, and flights are potential targets, with payoff values defined by risk analysis of the flight. Due to location and timing constraints, however, a marshal cannot be on all possible flights. For example, a marshal in New York cannot board flights flying out of Los Angeles. Legal schedules can be used to define the set of possible flights that a

[1] Our current implementation uses complete matrices to represent $H$ and $Ca$, but sparse representations could offer additional performance improvements.

**Table 3.2** Notation table

| Symbol | Meaning |
|---|---|
| $d^\gamma$ | Reward of defender against adversary of type $\gamma$ |
| $k^\gamma$ | Reward of adversary type $\gamma$ |
| $p^\gamma$ | Probability of occurrence of adversary of type $\gamma$ |
| $\Gamma$ | Set of adversary types |
| $T$ | Set of targets |
| $A^\gamma$ | Attack vector for the adversary of type $\gamma$ |
| $a_t^\gamma$ | Probability of adversary of type $\gamma$ attacking target $t$ |
| $C$ | Coverage vector of the defender |
| $c_t$ | Probability of defender covering target $t$ |
| $h(s, \omega)$ | Probability of coverage of schedule $s$ by defender type $\omega$ |
| $x_s$ | Total coverage probability over schedule $s$ |
| $S$ | Set of valid schedules |
| $\Omega$ | Set of resource types |
| $Ca(s, \omega)$ | Capability: 1 if type $\omega$ can cover schedule $s$; 0 otherwise |
| $\mathcal{R}(\omega)$ | Number of available resources of type $\omega$ |
| $H(s, t)$ | Mapping: 1 if schedule $s$ covers target $t$; 0 otherwise |
| $M$ | Huge positive constant |
| $U_\mathrm{d}^\gamma(t, C)$ | Utility of the defender when facing adversary type $\gamma$ who attacks target $t$ when defender coverage is $C$ |
| $U_\mathrm{a}^\gamma(t, C)$ | Utility of the adversary of type $\gamma$ when target $t$ is attacked and defender coverage is $C$ |

federal air marshal could fly, given these constraints. Resource types are use to define the initial state (notably, location) of a marshal, which defines a subset of legal schedules that any given marshal could fly.

Adding scheduling and resource coverage constraints reduces the space of feasible coverage vectors. Consider an example with a single federal air marshal defending three flights. Suppose that 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.

The algorithm is a mixed-integer linear program (MILP) described in (3.4)–(3.11), with notation presented in Table 3.2. Constraints (3.5) and (3.12) force each adversary to select a pure strategy attacking a single target. The coverage vector $C$ is constrained by the number of available resources through (3.8) and the coverage in each target to be in the range [0, 1] by (3.13). The coverage of each schedule must sum to the contributions of the individual resource types, specified by constraint (3.6). The mapping between the coverage of schedules and coverage of targets is enforced in (3.7). Constraint (3.8) restricts the schedule so that only the available number of resources of each type are used. Constraint (3.9) enforces that no probability may be assigned infeasible schedules for each resource type. The defender's expected payoff is defined with constraint (3.10) when follower $\gamma$ attacks target $A^\gamma$. Since the objective maximizes $d^\gamma$, for any optimal solution $d^\gamma = U_\mathrm{d}^\gamma(C, A^\gamma)$. This also implies that $C$ is maximal, given $A^\gamma$ for any optimal solution, since $d^\gamma$ is maximized. In a similar way, constraint (3.11) forces the attacker to select a strategy in the attack set of $C$. If the attack vector specifies a target that is not maximal, this constraint is violated. Therefore, taken together, the objective and constraints (3.10)–(3.11) imply that $C$ and $A^\gamma$ are mutual best-responses for the defender and the adversary in any solution. Thus, the defender mixed strategy $C$ and the adversary attack vector $A^\gamma$ for each adversary type $\gamma$ form an SSE of the security Stackelberg game.

$$\max_{a,c,q,h,d,k} \sum_{\gamma\in\Gamma} d^\gamma p^\gamma \tag{3.4}$$

$$\sum_{t\in T} a_t^\gamma = 1 \quad \gamma \in \Gamma \tag{3.5}$$

$$\sum_{\omega\in\Omega} h_{s,\omega} = x_s \quad s \in S \tag{3.6}$$

$$\sum_{s\in S} x_s H(s,t) = c_t \quad t \in T \tag{3.7}$$

$$\sum_{s\in S} h_{s,\omega} Ca(s,\omega) \leq \mathscr{R}(\omega) \quad \omega \in \Omega \tag{3.8}$$

$$h_{s,\omega} \leq Ca(s,\omega) \quad s,\omega \in S \times \Omega \tag{3.9}$$

$$d^\gamma - U_d^\gamma(t,C) \leq (1-a_t^\gamma)\cdot M \quad t \in T, \gamma \in \Gamma \tag{3.10}$$

$$0 \leq k^\gamma - U_a^\gamma(t,C) \leq (1-a_t^\gamma)\cdot M \quad t \in T, \gamma \in \Gamma \tag{3.11}$$

$$a_t^\gamma \in \{0,1\} \quad t \in T, \gamma \in \Gamma \qquad (3.13)$$

$$c_t \in [0,1] \quad t \in T \qquad (3.13)$$

$$x_s \in [0,1] \quad s \in S \tag{3.14}$$

$$h_{s,\omega} \in [0,1] \quad s,\omega \in S \times \Omega \tag{3.15}$$

The payoff values $U_\mathrm{d}^\gamma(t, C)$ and $U_\mathrm{a}^\gamma(t, C)$ are calculated based on (3.1) and (3.2). The values of $U_\mathrm{d}^{\gamma,\, c}$ and $U_\mathrm{d}^{\gamma,\, u}$ used in these equations are the payoff values to the defender when a target is covered and uncovered, respectively. These values are provided by the domain experts, as described in Sect. 3.5. Similarly, the payoff values for the adversaries are also provided by the domain experts.

The values of other model parameters are calculated based on the user input and the game specification. Police officers and canines are the resources for ARMOR for checkpoint and ARMOR for canine, respectively. ARMOR does not differentiate between different resources (e.g., all canines are assumed to be equally capable), and hence there is exactly one resource type $\Omega$. The number of resources $\mathscr{R}$, i.e., checkpoints or canines, is directly input by the user in the system. In the case of ARMOR, the set of legal schedules is an assignment of a checkpoint to an inbound road and is automatically generated by the system since ARMOR is aware of the road map of the airport. The capability matrix $Ca$ in ARMOR consists of all ones since any resource could be assigned to any target. For example, any canine could be scheduled to any terminal.

Similarly, all the model parameters are defined based on user input and domain constraints in IRIS. The federal air marshals are the resources for IRIS. In IRIS, the different FAMS Offices form the different resource types. This information has already been supplied to IRIS by the domain experts. The numbers of resources of each type $\mathscr{R}$, that is the number of federal air marshals in each office, is directly input in IRIS by the end users. The set of legal schedules $S$ is provided as an input to the system by the FAMS in IRIS. Each schedule in IRIS is a sequence of flights that a federal air marshal can take to complete a tour. In IRIS, the capability matrix $Ca$ is defined based on resource types; for example, federal air marshals at the FAM office based in Los Angeles can only cover schedules flying out of Los Angeles, and hence only those schedules would have their capabilities set to 1. The mapping $M$ is also calculated by the systems based on the domain specifications. For example, in IRIS, if schedule $s$ is to take flight f1 followed by flight f2, then the row in $M$ corresponding to $s$ would have ones only for columns corresponding to f1 and f2.

Kiekintveld et al. (2009) have shown that the ERASER-C MILP corresponds to an SSE of the security game. The intuition behind the proof are two claims: (1) the coverage probability of the leader and the attack set of the follower are mutual best-responses by the construction of the MILP, and (2) the coverage probability of the leader gives the leader the optimal utility.

## 3.4 Software Systems Deployed at the LAX and FAMS Domains

Both LAX and FAMS are security scenarios in which there is a leader/follower dynamic between the security forces and terrorist adversaries. In both domains there are limited resources available to protect a very large space of possible targets, so it is not possible to provide complete coverage. Finally, the targets have diverse values and vulnerabilities in each domain. The domains, however, differ primarily due to size. In the LAX security domain there are eight terminals that must be protected, while the air marshals are responsible for protecting tens of thousands of commercial flights each day. This difference in size requires, in addition to scalable solution algorithms, different types of interfaces to have domain experts specify each game. Finally, while in the LAX domain all security resources can reach all targets, in the FAMS domain, the security resources must satisfy more complicated constraints (e.g., a given marshal cannot be assigned to two flights with overlapping time schedules).

In this section, we describe both security domains (LAX and FAMS) and discuss the architecture of the software systems developed for these domains. We begin with a description of the generic software architecture and then describe each domain and their specific software assistant. We finish this section with a list of lessons learned in doing these deployments.

### *3.4.1 Software Assistants*

We now describe in detail the system architecture for each of the two software assistants, focusing primarily on the ARMOR system but providing some discussion of IRIS as a point of comparison. We paid particular attention to organization acceptance during the development process. The end users of both ARMOR and IRIS are security officers, and the system must be simple enough for them to be comfortable using it on a regular basis. In particular, the systems are designed to hide as much of the complexity of the game-theoretic models as possible, while still allowing enough flexibility for the users to input important parameters that change regularly. This required considerable effort in both user interface design and identifying ways to simplify and reduce the inputs required by the system to specify a game model. In the case of IRIS, it was also very important to build in functionality to import data from other systems to ease the burden of data entry (e.g., importing flight information from existing databases). Finally, the schedules that the system produces must be presented in a format that is easy to understand, with tools that allow final modifications if necessary.

Both ARMOR and IRIS are stand-alone desktop applications. ARMOR was developed in the Microsoft.NET framework, while IRIS is a stand-alone Java application. Due to security concerns, both systems are run on machines that are not connected to any network. The underlying solution methods use the open source GLPK[2] toolkit to solve the necessary mixed-integer programs. The general structure of the two applications is shown in Fig. 3.1. The core architecture can be divided into three modules, which we describe in detail in the subsequent sections:

1. *Input*: Interface for the user to enter parameters and domain knowledge.
2. *Back-end*: Inputs are translated into a game model, which is passed to the Bayesian Stackelberg game solver and then to a final process that generates a specific sample schedule based on the computed probabilities.
3. *Display Module*: The final schedule is presented to the user, with options to modify the output if necessary.

[2] http://www.gnu.org/software/glpk/.

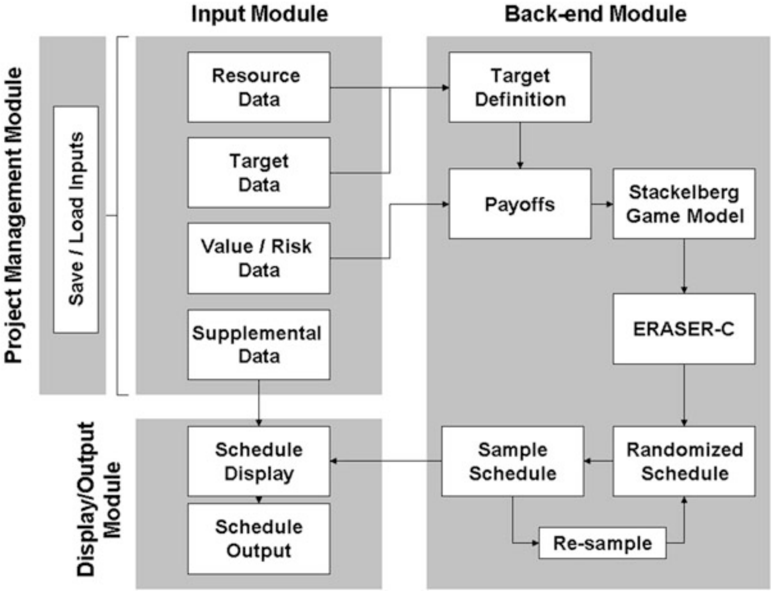


**Fig. 3.1** General structure for the security assistants

We rely on the users and domain experts to provide the knowledge required to specify the game model. While some elements of the model do not change over time, others change frequently. For these, we must provide the users a convenient way to enter the necessary values. The basic inputs that both ARMOR and IRIS require fall into four categories: (1) the number of available resources and their capabilities, (2) the set of targets, (3) payoff values for each target, and (4) supplemental data to improve the user experience (e.g., names and labels). Both applications allow users to save and reuse this information across multiple executions.

The balance of how much information is hard-coded and how much is entered by the user is quite different for ARMOR and IRIS. For example, in ARMOR the set of targets is hard-coded because the number of terminals at LAX changes very rarely. However, in IRIS the flight information may change every time the system is run, so this is part of the user input. Determining which parameters were necessary to expose to the user was a significant task, and required several iterations with the domain experts and end users to strike the right balance between the complexity of the inputs and the flexibility of the system to capture the necessary information.

The Back-end module is fairly common to the two applications. This model builds a specific instance of a Bayesian Stackelberg game, based on all of the data provided by domain experts and entered through the GUI by end users. Some of the necessary information is hard-coded in each system, while other inputs can be modified by the user during the scheduling process.

Once an explicit game model has been generated, it is passed as input to the ERASER-C mixed-integer program. This model is solved using the standard open source solver GLPK in these applications. ERASER-C returns an optimal mixed strategy for the defender—a probability distribution over the defender's actions—which represents a randomized policy for allocating the security resources of either LAX or FAMS. We sample the randomized schedule found to generate a specific schedule for the security forces. This sample schedule specifies exactly where and when each resource should be assigned to each target. If necessary, it is also possible to "resample" from the randomized schedule to get another specific schedule, though this capability is used rarely. Any specific constraints that the schedules must satisfy are taken into consideration when final schedules are sampled. These sampled schedules are then displayed for the user through the Display Module.

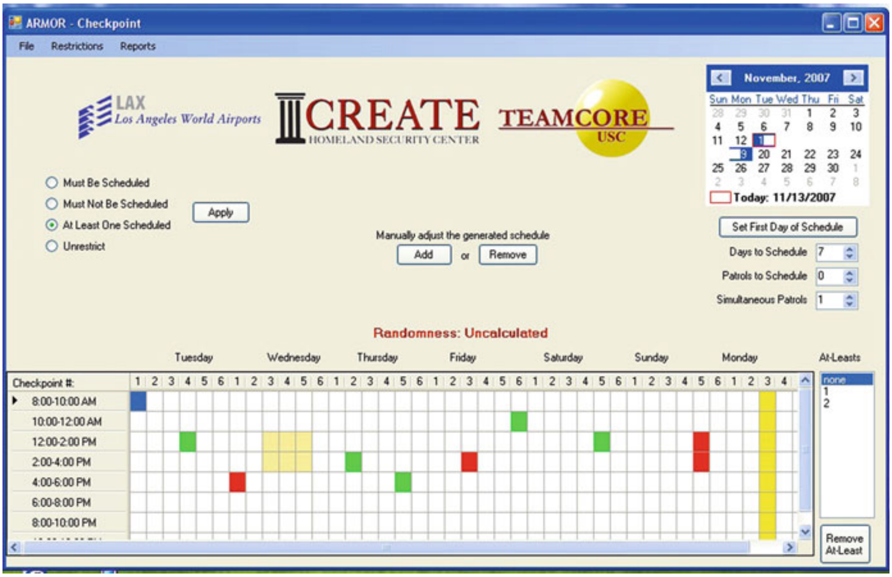
The output module presents the generated sampled schedule to the user. The user can then review the schedule and accept it as is, or add additional constraints and run the scheduling process again. Since the specifics of Input and Display Modules are domain dependent we describe both of them, first for LAX and then for FAMS.

### *3.4.2 LAX Domain: ARMOR*

LAX is the fifth busiest airport in the USA, the largest destination airport in the USA, and serves 60–70 million passengers per year (General description, 2007; Stevens et al., 2006). LAX is known to be a prime terrorist target on the west coast of the USA, with multiple arrests of plotters attempting to attack LAX (Stevens et al., 2006). To protect LAX, the airport police have designed a security system that utilizes multiple rings of protection. As is evident to anyone traveling through the airport, these rings include such things as vehicular checkpoints, police units patrolling the roads to the terminals, patrolling inside the terminals (with canines), and security screening and bag checks for passengers. Airport police use intelligent randomization within two of these rings: (1) placing vehicle checkpoints on inbound roads that service the LAX terminals, including both location and timing (a checkpoint is shown in Fig. 3.2), and (2) scheduling patrols for bomb-sniffing canine units at the different LAX terminals (as shown in Fig. 3.2). The numbers of available vehicle checkpoints and canine units are limited by resource constraints, so randomization is used as a method to increase the effectiveness of these resources while avoiding creating patterns in deployment.

The eight different terminals at LAX have very different characteristics, leading to different assessments of the value/risk for each terminal. For example, international flights are concentrated at a few terminals, while terminals have varying physical size and passenger loads. Because uncertainty about the adversary was identified by airport police as a key problem, the model should take into account the different types of adversaries that may be encountered. For example, there may be both hard-line, well-funded international terrorists planning attacks as well as

**Fig. 3.2** Security checkpoints and canine patrols at LAX



○ Must Be Scheduled
○ Must Not Be Scheduled
● At Least One Scheduled
○ Unrestrict

Today: 11/13/2007

Days to Schedule 7

Patrols to Schedule 0

Simultaneous Patrols 1

Randomness: Uncalculated

**Fig. 3.3** ARMOR interface

amateur individuals. The payoff values for different attack scenarios should depend on the type of attacker and their capabilities.

The interface for the ARMOR checkpoints program is shown in Fig. 3.3 and provides options for the number of available resources, the number of scheduled days, the time slots to schedule, and the monthly calendar. A spreadsheet is used to display the proposed schedule and provide additional opportunities for the end users to modify the schedules in an iterative process. Three options are provided to change the possible scheduling actions: (a) number of checkpoints allowed during a particular time slot; (b) the time interval of each time slot; (c) the number of days to schedule over. Furthermore, three options are given to the user to enforce constraints onto the schedule: (a) forced checkpoint; (b) forbidden checkpoint; (c) at least one checkpoint. These constraints are intended to be used sparingly to accommodate situations where a user, faced with exceptional circumstances and extra knowledge, wishes to influence the output of the game. The user can impose these specific actions in the schedule using the spreadsheet interface. Each restriction is represented by a different color in the spreadsheet. The interface for the ARMOR Canine Patrols at LAX has similar features.

ARMOR generates a different game for each time slot on each day. The number of defender resources in the model is the number of canine units/checkpoints specified by the user. The number of targets is the number of terminals for the canines system, and the number of inbound roads for the checkpoints system. Generating the game matrix also requires values for the payoffs associated with each possible target. These payoff values depend on a variety of conditions, such as passenger loads, cost of the infrastructure, and publicity to the adversary. Domain experts provided us with formulae to automatically generate payoff values for all possible combinations of such conditions, which we encode in ARMOR. The system is also provided with estimates of the passenger load and other elements (the details of these formulae and tools cannot be discussed due to security concerns). For any given day, ARMOR is able to take the conditions for this day and select appropriate payoff values for the targets. As a result, it is not necessary for LAX police officers to enter these values by hand to generate each schedule, which is both time-consuming and error-prone. The system still retains a high degree of flexibility because values are precomputed and stored for a wide range of possible conditions.

The generated schedule of checkpoints and canines is presented to the user via a spreadsheet. Each row in the output spreadsheet corresponds to 1 h. Each column in the sheet corresponds to a terminal. Each entry in the sheet represents a schedule generated by ARMOR. The familiarity of the police officers with spreadsheets helped in the acceptance of the ARMOR schedules.

When ARMOR identifies that user constraints are causing unreasonably low likelihood of scheduling a checkpoint, it presents the schedule to the user with alerts. The user may then alter the schedule by modifying the forbidden/required checkpoints, or possibly by directly altering the schedule. Both possibilities are accommodated in ARMOR. If the user simply adds or removes constraints, ARMOR can create a new schedule. Once the schedule is finalized, it can be saved for actual use, thus completing the system cycle. This full process was designed to specifically meet the requirements at LAX for checkpoint and canine allocation.

### *3.4.3 FAMS Domain: IRIS*

The FAMS places undercover law enforcement personnel aboard flight soriginating in and departing from the USA to dissuade potential aggressors and prevent an attack should one occur (TSA, 2008). The exact methods used to evaluate the risks posed by individual flights is not made public by the service, but we can identify many factors that might influence such an evaluation. For example, flights have different numbers of passengers, and some fly over densely populated areas while others do not. International flights also serve different countries, which may pose different risks. Special events can also change the risks for particular flights at certain times (Federal Air Marshal Service, 2008).

The scale of the domain is massive. There are currently tens of thousands of commercial flights scheduled each day, and public estimates state that there are thousands of air marshals. Air marshals must be scheduled on tours of flights that obey various constraints (e.g., the time required to board, fly, and disembark). Simply finding schedules for the marshals that meet all of these constraints is a computational challenge. Our task is made more difficult by the need to find a randomized policy that meets these scheduling constraints, while also accounting for the different values of each flight.

The FAMS domain is considerably larger, and the information required to build a game model in this domain changes much more frequently. For these reasons the application is considerably more complex than ARMOR in terms of the user interface and the mechanisms required to input all of the necessary information. This additional complexity is necessary in this domain to accurately capture the situation and provide all of the functionality requested by the end users. However, it does place a greater burden on the users to learn the system, and scheduling is a more time-consuming process than in ARMOR. Again, finding the right level of complexity to expose to the users was an iterative process that involved many discussions with the users and domain experts.

In the FAMS domain, we require information about the available air marshals, their scheduling constraints, the possible flights, and information about the risks/values to associate with each flight. The data about resources include information about the number and location of air marshals, as well as the conditions that define legal flight schedules. Flight information includes various data about each flight, including flight number, carrier, origin, destination, aircraft type, etc. Finally, some information is collected to improve usability, even though it is not strictly necessary for the game-theoretic analysis. This includes naming schemes for airports and airlines and other information that allows the system to output schedules in a more usable format or to interface easily with other systems. IRIS also includes functionality to import data from existing databases with flight data and other information. This greatly reduces the amount of data entry necessary to create a schedule.

Specifying the payoff values for every possible flight was a particular challenge in this domain, since there are thousands of flights to consider. We use an attributed-based system to elicit these values, based on the Threat, Vulnerability, and Consequence (TVC) model for estimating terrorism risk (Willis et al., 2005). By eliciting values for attributes of flights rather than specific flights, we are able to dramatically reduce the number of entries required by the user. Each flight is then given an aggregate value based on these components; the specific calculations used to determine flight risk are sensitive information and cannot be revealed. The values of the attributes for each flight can be populated automatically from existing databases. To allow for specific intelligence or exceptional circumstances, the individual payoff values for any flight can also be directly edited by the end user. However, this is only rarely necessary and the majority of the analysis can be effectively automated.

This preference elicitation system of IRIS has substantially reduced the number of values that must be entered by the user. During a restricted test run on real data, the attribute-based approach called for a total of 114 values to input regardless of the number of flights. By contrast, there were 2,571 valid flights over a week, each requiring four payoff values, summing to 10,284 user-entered values without the attribute-based preference elicitation system. The attribute-based approach clearly requires far fewer inputs and remains constant as the number of flights increases, allowing for excellent scalability as we deal with larger and larger sets of flights. Equally importantly, attribute-based risk assessment is an intuitive and highly scalable method that can be used in any problem where people must distill numerous attributes of a situation into a single value for a large number of situations that share the same attributes.

The generated schedules are presented to the user via the application window. The schedule created is shown in the interface, and allows the users to view more detailed information about each target. The user is also able to output the schedule to a file which can then be used to analyze the schedule in more detail. The sample assignment of federal air marshals to flight schedules is exactly a schedule that could be used by the FAMS. At this point, the scheduling assistant allows the expert using the system to create numerous sample schedules based on the same optimal mixed strategy or to change the assignment of federal air marshals to flight schedules by hand to create a final schedule that meets the needs of the FAMS. Of course, the user can also adjust any of the parameters entered and resolve the game completely. The output of IRIS is in the same format as the other systems used by the FAMS officers. It has not been presented here for simplicity and because of security concerns.

### *3.4.4 Lessons Learned*

The design and deployment of ARMOR and IRIS have posed numerous challenges. We outline some key lessons learned during the design and deployment of these tools. First, there is a critical need for randomization in security operations. Security officials are aware that requiring humans to generate randomized schedules is unsatisfactory because, as psychological studies have often shown (Wagenaar, 1972; Treisman and Faulkner, 1987), humans have difficulty in randomizing, and they can also fall into predictable patterns. Instead, game-theoretic randomization that appropriately weighs the costs and benefits of different actions and randomizes with appropriate weights leads to improved results. Security officials were therefore extremely enthusiastic in their reception of our research and eager to apply it to their practices.

Second, organizational acceptance is a key issue. In creating solutions for people, we must be cognizant of how difficult it will be for a user to adopt our solution. Each deviation from existing methodology is a step away from the familiar that we must convince the user to accept. Instead of asking people to make numerous and sometimes unnecessary changes, minimizing these differences and complexities can help pave the way toward a successful implementation. For example, tweaking the GUI to achieve a look and feel that the user is familiar and comfortable with can help the user understand the system faster and better. Similarly, because infrastructural changes are often costly and/or time-consuming, ease of incorporating our work into their daily routine is essential. For example, using inputs and creating outputs that were in the same format as existing protocols minimized the additional work that our assistant would create for the security officers and lead to easier acceptance of the system.

Third, it is important to provide the users with operational flexibility. When initially generating schedules for canine patrols, we created a very detailed schedule, micro-managing the patrols. This did not get as positive a reception from the officers. Instead, an abstract schedule that afforded the officers some flexibility to respond to situations on the ground was better received.

## 3.5 A Generic Network Security Problem

As noted above, implementing a Stackelberg security game model to plan patrols is a difficult process that to date has been undertaken with substantial effort in collaboration with the security providers. In many situations, however, there is enough information about the security process that a data-driven process could be used to assist security providers in defining the actions and payoffs of the security game. In this section we illustrate recent work that aims to automatically build a Stackelberg security game for the problem of patrolling a street network to prevent crime. The proposed approach uses data-mining tools on a database of past reported crime and events to identify the locations to be patrolled, the times at which the game changes, and the types of adversaries faced. The idea is to exploit temporal and spatial patterns of crime on the area to be patrolled to determine the priorities on how to use the limited security resources.

We consider the street network depicted in Fig. 3.4 which corresponds to a centric commercial, turistic, and economic district in Santiago, Chile. This is a busy part of the city usually with large crowds on the street and that historically concentrates a high number of crimes, for the most part theft or minor aggressions. This type of crime in particular can be deterred or reduced with appropriate patrolling by police. To represent the problem of deciding where to patrol as a Stackelberg security game, security providers need to identify the specific points on this street network that concentrate crime and determine the payoffs defenders and attackers would receive if crimes at these locations are committed or are prevented. In this security game, police patrols on foot would go to the points selected

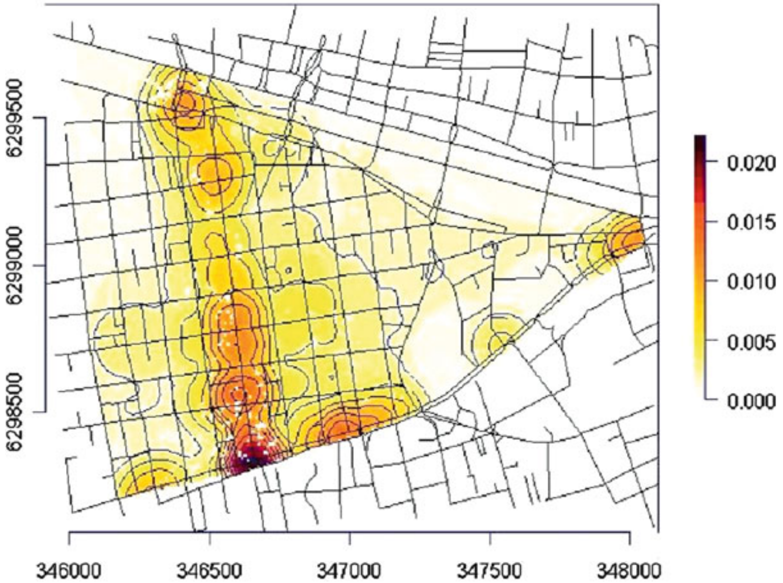


**Fig. 3.4** Patrolling area with a density plot of reported crimes in the period 12/15/2002–12/14/2004

following the random optimal mixed strategy that maximizes the defender's utility. Different types of criminals would, knowing the optimal mixed strategy of the police patrols, then decide where to attack on the network, if at all. We assume that both police and criminals appear at the point selected, without interacting in other parts of the network. In addition to the description of the street network, we obtained from the Chilean national police force information about reported crimes in the area and the police reports for a 2-year period. Each reported crime has a location, a date and time, and a description of the crime (classification of crime [robbery, theft, etc.], amount stolen, level of violence, etc.). The police reports include information about the available resources in each shift, which helps estimate the police resources used for preventive patrolling.

### *3.5.1 Building a Data-Driven Security Game*

This information is then processed in an automated data-driven procedure to build a security game in five steps: (1) Define the amount of data that will be relevant to calibrate the security game, (2) Determine locations to patrol, (3) Identify attacker types from data, (4) Determine times to patrol, (5) Determine payoffs for leader and followers.

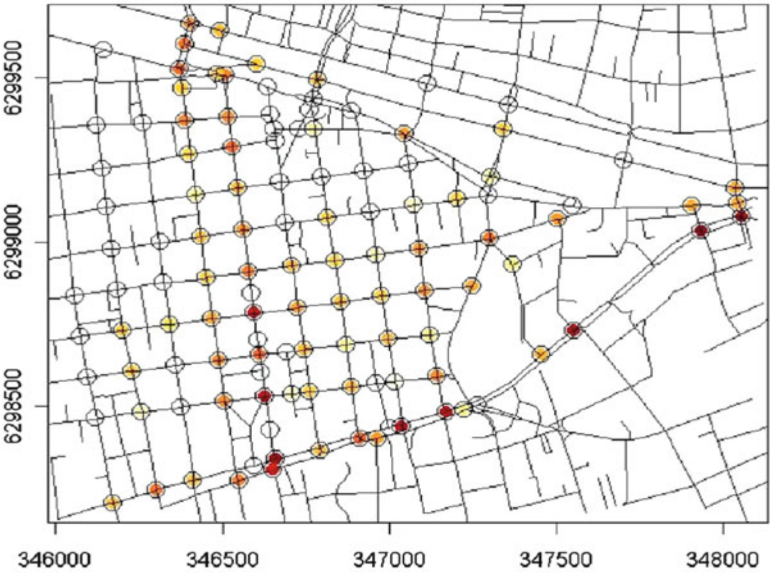


**Fig. 3.5** Optimal mixed strategy on locations selected. Node color corresponds to probability of coverage of the node, with a *darker color* indicating a higher probability of coverage

*Step 1*: In determining which data to use to build a representative security game, we must strike a balance between selecting too much data and too little. Sufficient past data should be included so that significant but perhaps rare patterns of crime are taken into consideration. However, if too much data are taken into account, we run the risk of representing crime patterns that no longer exist. You must rely on expert opinion to estimate how representative past data are of the current security situation leading to an estimate of how much of the past information to use in identifying the locations of the patrols, the types of adversaries, and the utilities for each. In the results we show below, we used a time window of 2 years of data (from December 15, 2002 until December 14, 2004) to build a week long game (for the week of December 15–22, 2004).

*Step 2*: We used an off-the-shelf clustering software to identify the locations to be patrolled from the density plot of reported crime displayed in Fig. 3.5. These locations are selected anywhere on the road network in a way that summarizes the geographical distribution of crimes without requiring a massive number of locations. We used the software DBSCAN (density-based spatial clustering of applications with noise) (Ester et al., 1996). This is a density segmentation tool which also removes the noise in the data and automatically selects the number of segments to consider. In the results we obtained, DBSCAN identifies 119 locations to protect in which there are at least 10 crimes within a radius of 20 m. These points represent 89.23% of the reported crimes. We note that a number of good clustering algorithms can help in identifying a set of locations that are representative of the spatial crime distribution.

*Step 3*: We follow the knowledge discovery in databases (KDD) scheme (Fayyad et al., 1996) to process the database of reported crimes and identify different types of attackers. The KDD approach is a generic scheme that outlines a series of procedures to, among other things, create a target data set, remove data noise and outliers, handle missing data, identify useful features in the data, etc. Each of the processes can be implemented with any of a number of existing tools. For the selection of attributes, we chose a wrapper technique that automatically selects the attributes that help segmentation (Dy and Brodley, 2004). To identify the clusters of crimes we use a $k$-means clustering model. We found that this model was superior to alternative clustering models we tried ($X$-means, expectation maximization) for this problem, both in runtime and the quality of solutions found, which are more easily interpretable. The number of reported crimes in each cluster informs us of the frequency of different types of crime and thus the likelihood of facing each. The crimes in the 2-year database were classified into 9 significant clusters that were characterized by 24 significant attributes.

*Step 4*: Since the security conditions change during the day and the Stackelberg security game describes static conditions, we separate the day into different time intervals (or blocks) in which the security conditions remain almost constant. The different types of crime identified in Step 3 include three different time blocks which are found to be significant. Intersecting these times with the police patrolling shifts gives us a total of seven time intervals, or blocks, where the likelihood and composition of different types of crime and patrolling resources are kept about constant.

| Block | From | To | Block | From | To | Block | From | To |
|---|---|---|---|---|---|---|---|---|
| S1 | 0:00 | 6:59 | S2 | 7:00 | 9:59 | S3 | 10:00 | 14:59 |
| S4 | 15:00 | 17:59 | S5 | 18:00 | 19:59 | S6 | 20:00 | 21:59 |
| S7 | 22:00 | 23:59 | | | | | | |

In blocks S2 and S3 there are 23 patrolling units available, in blocks S4, S5, and S6 there are 24 patrolling units, and in blocks S1 and S7 there are nine patrolling units. Here, one patrolling unit corresponds to a pair of policemen on foot.

We determine the probability of facing each type of adversary by the frequency with which each of the nine types of crimes occur. To make this frequency more dependent on recent events, the past event data are scaled with an exponential decay function. Table 3.3 shows these frequencies for each of the nine types of crimes over the seven time blocks found.

**Table 3.3** Probability of facing each follower in the different time blocks

| Cluster | S1 | S2 | S3 | S4 | S5 | S6 | S7 | Total |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.234 | 0.516 | 0.624 | 0 | 0.603 | 0.562 | 0.395 | 1,815 |
| 1 | 0.078 | 0.057 | 0.048 | 0.142 | 0.049 | 0.079 | 0.097 | 679 |
| 2 | 0 | 0 | 0 | 0.470 | 0 | 0 | 0 | 545 |
| 3 | 0.032 | 0.018 | 0.018 | 0 | 0.012 | 0.027 | 0.050 | 369 |
| 4 | 0 | 0 | 0 | 0.260 | 0 | 0 | 0 | 405 |
| 5 | 0.253 | 0.091 | 0.063 | 0.079 | 0.066 | 0.093 | 0.150 | 808 |
| 6 | 0.023 | 0.027 | 0.022 | 0.048 | 0.033 | 0.016 | 0.024 | 419 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0.223 | 0.285 | 575 |
| 8 | 0.381 | 0.291 | 0.225 | 0 | 0.238 | 0 | 0 | 1,110 |
| Total | 727 | 457 | 1,892 | 1,217 | 939 | 881 | 612 | |

**Table 3.4** Expected payoff for each type of criminal, in US dollars, if attack is successful (average utility) and if attack is unsuccessful (average cost, using a 40 % discount rate while in prison)

| Cluster | Average utility | Prison time | Average cost |
|---|---|---|---|
| 0 | 182 | 61 | 638 |
| 1 | 209 | 1,752 | 731 |
| 2 | 136 | 63 | 476 |
| 3 | 451 | 1,746 | 1,579 |
| 4 | 175 | 1,747 | 614 |
| 5 | 217 | 1,686 | 761 |
| 6 | 139 | 74 | 486 |
| 7 | 138 | 1,757 | 485 |
| 8 | 218 | 1,739 | 764 |

*Step 5*: In this work we determine the payoffs for the attacker as a valuation of the monetary payoff of being successful or getting caught for each type of crime. In the case of the police, we estimate that the payoff for catching a criminal is zero (for all types) while the penalty for a successful crime equals the expected amount earned by that type of criminal. We first determine from the information on the database the average expected reward for the criminal in a successful attack. To determine the penalty of an unsuccessful attack, we estimated the expected number of days in jail for that type of crime and evaluated the amount of forgone earnings for the criminal for not being able to commit crimes during that period. We note that there are a number of alternative models that can be incorporated here, in particular models of risk aversion that better represent human behavior in adversarial environments, such as prospect theory (Kahneman and Tvesky, 1979) or quantal response (McKelvey and Palfrey, 1995). Table 3.4 presents the values of payoffs for the Stackelberg game for each of the nine types of adversaries.

**Table 3.5** Defender's expected utility in different time blocks

| Block of time | S1 | S2 | S3 | S4 | S5 | S6 | S7 |
|---|---|---|---|---|---|---|---|
| Stackelberg (mixed) | – 1.87 | – 0.31 | – 0.29 | – 0.24 | – 0.21 | – 0.22 | – 1.63 |
| Stackelberg (pure) | – 8.87 | – 3.90 | – 3.62 | – 3.27 | – 3.53 | – 3.48 | – 8.30 |
| Maximin (mixed) | – 5.40 | – 2.42 | – 2.21 | – 1.94 | – 2.18 | – 2.18 | – 5.10 |
| Maximin (pure) | – 8.87 | – 3.95 | – 3.67 | – 3.27 | – 3.53 | – 3.48 | – 8.30 |

### *3.5.2 Additional Considerations in a Data-Driven Security Game*

The procedure above helps security providers build a Stackelberg security game to determine efficient patrols in an urban street network. This game can then be formulated as the mixed integer programs described in Sect. 3.3 and solved to optimality. A solution for this problem is depicted in Fig. 3.5. The color at each node corresponds to the amount of coverage in the optimal mixed strategy for a certain time block. To implement this solution, the police should sample from this distribution to decide which locations to patrol each day in every time block.

The game developed can also be used to evaluate the current practice and the proposed patrol plan. Currently police direct their preventive patrols to the locations where the highest concentration of crime is expected to occur, based on recent past activity (2 weeks). We assume that the highest concentration of crime are the locations where the game predicts the highest payoff for the adversary, therefore directing the patrols to the maximum payoff locations leading to a minimax strategy. Table 3.5 presents the defender's expected profits in each time block under each of four different strategies: the optimal mixed and pure strategies of the Stackelberg game and Minimax. We note that the utility for the leader is always better in the Stackelberg game (mixed).

The set of tools described here hope to complement the experience and intuition of law enforcement. There is much information that is difficult to include in decisions on how to patrol. This is the case in part because of the amount of data and in part because the data are not being collected or are biased. We note that a better description of the security problem can be obtained, and thus a better security game formulated, by incorporating additional sources of information, such as surveys of victimization and physical description of places. We believe this is an interesting avenue of future research to create robust systems that would be more easily deployable in diverse settings.

## 3.6 Conclusions

Monitoring and patrolling are key components of law enforcement in security domains. In generating schedules for these patrols, it is important to account for varying weights of the targets being protected as well as the fact that potential attackers can often observe the procedures being used. This chapter describes scheduling assistants for the LAX police, ARMOR, and the FAMS, IRIS, which provide game-theoretic solutions to this problem. The two systems assist the security forces in generating randomized patrols while ensuring that differences in importance of different targets are preserved. A critical observation in the deployment of these scheduling assistants is the difficulty faced in reducing a complex security domain to a Stackelberg game model. To address this difficulty we present a data-mining-based model to assist security personnel in defining the Stackelberg security game from historic data.

ARMOR and IRIS make use of algorithmic advances in multi-agent systems research to solve the class of massive security games with complex constraints that were not previously solvable in realistic time-frames. Thus, although our applications were designed to be deployed at LAX and FAMS, they provide a general framework for solving patrolling scheduling problems in other domains as well.

Our approach of using Stackelberg games to model real-world security problems is applicable in a wide range of domains that share the following attributes: (a) there are intelligent players, (b) one player's strategy is observable by the other player, (c) player's have varying preferences among targets, and (d) it is not possible to provide full coverage of all targets. Some examples of similar security situations include security in computer networks, checkpoints at subway stations, security inspections at ports, and monitoring of other mediums of public transport.

Ultimately the security providers (Police, Air Marshals) are the judge of the usefulness of these Stackelberg security game models. As in any model it is critical to allow for expert knowledge to inform the system and provide feedback on the quality of solutions. With this in mind the development of the interface of these deployed systems has been an important aspect of this work. This research and these applications have been effective in helping in the security officers with scheduling and patrolling concerns. Thus, ARMOR and IRIS represent successful transitions of game-theoretic advances to applications that have been in use and effective in the real world. There are a number of additional improvements to these systems that could be done in the future to facilitate deployment to different domains. Some lines of future research include methods to incorporate qualitative information (estimates of unreported crime, fear of crime, etc.) to construct the Stackelberg games; coordination of different security resources; and considering attackers who deviate from rational behavior (due to differences in information or human bias).

## References

Agmon N, Sadov V, Kaminka GA, Kraus S (2008) The impact of adversarial knowledge on adversarial planning in perimeter patrol. In: AAMAS

An B, Pita J, Shieh E, Tambe M, Kiekintveld C, Marecki J (2011) GUARDS and PROTECT: next generation applications of security games. ACM SIGecom Exchanges 10(1):31–34

Avenhaus R, von Stengel B, Zamir S (2002) Inspection games. In: Aumann RJ, Hart S (eds) Handbook of game theory, vol 3. North-Holland, Amsterdam, pp 1947–1987 (Chap. 51)

Babu L, Lin L, Batta R (2006) Passenger grouping under constant threat probability in an airport security system. Eur J Oper Res 168:633–644

Bard JF (1999) Practical bilevel optimization: algorithms and applications. Nonconvex optimization and its applications, vol 30. Springer, Berlin

Basar T, Olsder GJ (1995) Dynamic noncooperative game theory, 2nd edn. Academic, San Diego

Bier VM (2007) Choosing what to protect. Risk Anal 27(3):607–620

Blanco M, Valino A, Heijs J, Baumert T, Gomez JG (2007) The economic cost of March 11: measuring the direct economic cost of the terrorist attack on March 11, 2004 in Madrid. Terror Polit Viol 19(4):489–509

Breton M, Alg A, Haurie A (1988) Sequential stackelberg equilibria in two-person games. Optim Theor Appl 59(1):71–97

Brown G, Carlyle M, Kline J, Wood K (2005) A two-sided optimization for theater ballistic missile defense. Oper Res 53:263–275

Brown G, Carlyle M, Royset J, Wood K (2005) On the complexity of delaying an adversary's project. In: Golden B, Raghavan S, Wasil E (eds) The next wave in computing, optimization and decision technologies. Springer, Berlin, pp 3–17

Brown G, Carlyle M, Salmeron J, Wood K (2006) Defending critical infrastructure. Interfaces, 36 (6):530–544

Conitzer V, Sandholm T (2006) Computing the optimal strategy to commit to. In: ACM EC-06, pp 82–90

Dy JG, Brodley CE (2004) Feature selection for unsupervised learning. J Mach Learn Res 5:845–889

Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial database with noise. Technical report, Institute for Computer Science, University of Munich

Fayyad U, Piatetsky-Shapiro G, Smyth P (1996) From data mining to knowledge discovery: an overview. AI Mag 17(3):37–54

Federal Air Marshal Service (2008). http://en.wikipedia.org/wiki/Federal_Air_Marshal_Service

Fudenberg D, Tirole J (1991) Game theory. MIT, Cambridge

Gatti N (2008) Game theoretical insights in strategic patrolling: model and algorithm in normal-form. In: Ghallab M, Spyropoulos CD, Pakotakis N, Avouris N (eds) ECAI. IOS Press, Amsterdam, pp 403–407

General description: Just the facts (2007). http://www.lawa.org/lax/justTheFact.cfm

Harsanyi JC, Selten R (1972) A generalized Nash solution for two-person bargaining games with incomplete information. Manag Sci 18(5):80–106

Jain M, Tsai J, Pita J, Kiekintveld C, Rathi S, Ordóñez F, Tambe M (2010) Software assistants for patrol planning at LAX and federal air Marshals service. Interfaces 40(4):267–290

Jiang A, Leyton-Brown K (2006) A polynomial-time algorithm for action-graph games. Artif Intell 679–684

Kahneman D, Tvesky A (1979) Prospect theory: an analysis of decision under risk. Econometrica 47(2):263–292

Kiekintveld C, Jain M, Tsai J, Pita J, Tambe M, Ordóñez F (2009) Computing optimal randomized resource allocations for massive security games. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems. Budapest, Hungary, May 10–15, 1:689–696

Koller D, Milch B (2003) Multi-agent influence diagrams for representing and solving games. Games Econ Behav 45(1):181–221

Larson R (1974) A hypercube queueing modeling for facility location and redistricting in urban emergency services. J Comput Oper Res 1:67–95

Leitmann G (1978) On generalized stackelberg strategies. J Optim Theor Appl 26(4):637–643

Looney R (2002) Economic costs to the United States stemming from the 9/11 attacks. Strateg Insights 1(6)

Lye K-w, Wing JM (2005) Game strategies in network security. Int J Inf Secur 4(1–2):71–86

McKelvey RD, Palfrey TR (1995) Quantal response equilibria for normal form games. Games Econ Behav 10:6–38

Nie X, Batta R, Drury CG, Lin L (2007) Optimal placement of suicide bomber detectors. Mil Oper Res 12:65–78

Osbourne MJ, Rubinstein A (1994) A course in game theory. MIT, Cambridge

Paruchuri P, Tambe M, Ordonez F, Kraus S (2006) Security in multiagent systems by policy randomization. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-06). Hakodate, Japan, May 8–12, 273–280

Paruchuri P, Pearce JP, Tambe M, Ordóñez F, Kraus S (2007) An efficient heuristic approach for security against multiple adversaries. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007). Honolulu, Hawaii, May 14–18

Paruchuri P, Pearce JP, Marecki J, Tambe M, Ordóñez F, Kraus S (2008) Playing games with security: an efficient exact algorithm for bayesian stackelberg games. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems. Estoril, Portugal, May 12–16

Pita J, Jain M, Western C, Portway C, Tambe M, Ordóñez F, Kraus S, Parachuri P (2008) Deployed ARMOR protection: The application of a game-theoretic model for security at the Los Angeles international airport. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems. Estoril, Portugal, May 12–16

Pita J, Tambe M, Kiekintveld C, Cullen S, Steigerwald E (2011) GUARDS – game theoretic security allocation on a national scale. In: Proceedings of the 10th International conference on autonomous agents and multiagent systems. Taipei, Taiwan, May 2–6, 1:37–44

Ruan S, Meirina C, Yu F, Pattipati KR, Popp RL (2005) Patrolling in a stochastic environment. In: 10th international command and control research and tech. symposium, June 13–16

Sandler T, Arce DG (2003) Terrorism and game theory. Simul Gaming 34(3):319–337

Srivastava V, Neel J, MacKenzie AB, Menon R, Dasilva LA, Hicks JE, Reed JH, Gilles RP (2005) Using game theory to analyze wireless ad hoc networks. IEEE Commun Surv Tutuor 7(4)

Stevens D, Hamilton T, Schaffer M, Dunham-Scott D, Medby JJ, Chan EW, Gibson J, Eisman M, Mesic R, Kelley CT, Kim J, LaTourrette T, Riley KJ (2006) Implementing security improvement options at Los Angeles international airport. http://www.rand.org/pubs/documented_briefings/2006/RAND_DB499-1.pdf

Thornton P (2005) London bombings: Economic cost of attacks estimated at 2bn. July http://www.independent.co.uk/news/business/news/economic-cost-of-attacks-estimated-at-1632bn-499281.html

Treisman M, Faulkner A (1987) Generation of random sequences by human subjects: Cognitive operations or psychological process? J Exp Psychol 116(4):337–355

TSA: Federal Air Marshals (2008). http://www.tsa.gov/lawenforcement/programs/fams.shtm.

Tsai J, Rathi S, Kiekintveld C, Ordóñez F, Tambe M (2009) IRIS - A tool for strategic security application in transportation networks. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, Budapest, Hungary, May 10–15

von Stackelberg H (1934) Marktform und Gleichgewicht. Springer, Vienna

von Stengel B, Zamir S (2004) Leadership with commitment to mixed strategies. Technical report LSE-CDAM-2004-01, CDAM research report

Wagenaar WA (1972) Generation of random sequences by human subjects: A critical survey of literature. Psychol Bull 77(1):65–72

Wein LM (2009) Homeland security: From mathematical models to policy implementation: the 2008 Philip McCord Morse lecture. Oper Res 57(4):801–811

Willis H, Morral A, Kelly T, Medby J (2005) Estimating terrorism risk. RAND Corporation. Santa Monica. http://www.rand.org/pubs/monographs/2005/RAND MG388.pdf