Security Games with Arbitrary Schedules: A Branch and Price Approach

Manish Jain, Erim Kardeş,
Christopher Kiekintveld, Milind Tambe
Computer Science Department
University of Southern California
Los Angeles, CA 90089
{manish.jain,kardes,kiekintv,tambe}@usc.edu

Fernando Ordóñez
Industrial and Systems Engineering
University of Southern California
Los Angeles, CA - 90089; and
Industrial Engineering Department
University of Chile (Santiago)
fordon@usc.edu

Abstract

Security games, and important class of Stackelberg games, are used in deployed decision-support tools in use by LAX police and the Federal Air Marshals Service. The algorithms used to solve these games find optimal randomized schedules to allocate security resources for infrastructure protection. Unfortunately, the state of the art algorithms either fail to scale or to provide a correct solution for large problems with arbitrary scheduling constraints. We introduce ASPEN, a branch-and-price approach that overcomes these limitations based on two key contributions: (i) A column-generation approach that exploits a novel network flow representation, avoiding a combinatorial explosion of schedule allocations; (ii) A branch-and-bound algorithm that generates bounds via a fast algorithm for solving security games with relaxed scheduling constraints. ASPEN is the first known method for efficiently solving massive security games with arbitrary schedules.

Introduction

Algorithms for attacker-defender Stackelberg games, resulting in randomized schedules for deploying limited security resources at airports, subways, ports, and other critical infrastructure have garnered significant research interest (Paruchuri et al. 2008; Kiekintveld et al. 2009). Indeed, two important deployed security applications are using such algorithms: ARMOR and IRIS. ARMOR has been in use for over two years by Los Angeles International Airport police to generate canine-patrol and vehicle-checkpoint schedules (Pita et al. 2009). IRIS was recently deployed by the Federal Air Marshals Service (FAMS) to create flight schedules for air marshals (Tsai et al. 2009). These applications use efficient algorithms that solve large-scale games (Paruchuri et al. 2008; Conitzer and Sandholm 2006; Basilico, Gatti, and Amigoni 2009), the latest being ERASER-C, the algorithm used in IRIS.

Unfortunately, current state of the art algorithms for Stackelberg games are inadequate for many applications. For example, US carriers fly over 27,000 domestic and 2,000 international flights daily, presenting a massive scheduling challenge for FAMS. IRIS addresses an important part of this space — the international sector — but only considers schedules with a single departure and return flight. The ERASER-C algorithm used in this application does not provide correct solutions for longer and more complex tours (which are common in the domestic sector). In fact, recent complexity results show that the problem of finding Stackelberg equibria with general scheduling constraints is NP-hard (Korzhyk, Conitzer, and Parr 2010), and can be solved in polynomial time only for restricted cases. This is due to the exponential explosion in the size of the defender's strategy space caused by large, arbitrary schedules.

Despite the discouraging complexity results, arbitrary schedules are an important element of security scheduling problems for FAMS, border security, and many other applications. We develop algorithms for SPARS (*Security Problems with ARbitrary Schedules*), drawing on techniques used to solve very large mixed-integer programs. Our main contribution is ASPEN (Accelerated SPars ENgine), which is based on the branch and price framework. The first novel feature of ASPEN is that it uses column generation to avoid representing the full (exponential) strategy space for the defender. To this end, we provide a novel decomposition of SPARS into a master problem and a network flow subproblem. Second, ASPEN uses a novel branch-and-bound method for searching the space of attacker strategies, achieving significant performance improvements by integrating branching criteria and bounds generated with ORIGAMI algorithm (Kiekintveld et al. 2009). We evaluate ASPEN empirically on SPARS, illustrating that this is the first known method for efficiently solving real-world-sized security games with arbitrary schedules.

SPARS

A security game (Kiekintveld et al. 2009) is a two-player game between a defender and an attacker. The attacker's pure strategy space $\mathcal{A}$ is the set of targets T that could be attacked, $T = \{t_1, t_2, \ldots, t_n\}$. The corresponding mixed strategy $\mathbf{a} = \langle a_i \rangle$ is a vector where a_i represents the probability of attacking t_i. The defender allocates resources of different types $\lambda \in \Lambda$ to protect targets, with the number of available resources given by $R = \{r_1, r_2, \ldots, r_{|\Lambda|}\}$. Each resource can be assigned to a *schedule* covering multiple targets, $s \subseteq T$, so the set of all legal schedules $S \subseteq \mathcal{P}(T)$. There is a set of legal schedules for each λ, $S_\lambda \subseteq S$.

The defender's pure strategies are the set of *joint schedules* that assign each resource to at most one schedule. Additionally, we assume that a target may be covered by at most 1 resource in a joint schedule (though this can be generalized). A joint schedule $\mathbf{j}$ can be represented by the vector $\mathbf{P_j} = \langle P_{jt} \rangle \in \{0,1\}^n$ where P_{jt} represents whether or not target t is covered in joint schedule $\mathbf{j}$. The set of all feasible joint schedules is denoted by $\mathbf{J}$. We define a mapping M from $\mathbf{j}$ to $\mathbf{P_j}$ as: $M(\mathbf{j}) = \langle P_{jt} \rangle$, where $P_{jt} = 1$ if $t \in \bigcup_{s \in \mathbf{j}} s$; 0 otherwise. The defender's mixed strategy $\mathbf{x}$ specifies the probabilities of playing each $\mathbf{j} \in \mathbf{J}$, where each individual probability is denoted by x_j. Let $\mathbf{c} = \langle c_t \rangle$ be the vector of coverage probabilities corresponding to $\mathbf{x}$, where $c_t = \sum_{\mathbf{j} \in \mathbf{J}} P_{jt} x_j$ is the marginal probability of covering t.

Payoffs depend on the target attacked and whether or not a defender resource is covering the target. $U_d^c(t)$ denotes the defender's utility if t is attacked when it is covered by a resource of any type. If t is not covered, the defender gets $U_d^u(t)$. Likewise, the attacker's utilities are denoted by $U_a^c(t)$ and $U_a^u(t)$. We assume adding coverage to target t is strictly better for the defender and worse for the attacker: $U_d^c(t) > U_d^u(t)$ and $U_a^c(t) < U_a^u(t)$, however, not necessarily zero-sum. For a strategy profile $\langle \mathbf{c}, \mathbf{a} \rangle$, the expected utilities for the defender and attacker are given by:

$$U_d(\mathbf{c}, \mathbf{a}) = \sum_{t \in T} a_t \left(c_t U_d^c(t) + (1 - c_t) U_d^u(t) \right) \quad (1)$$

$$U_a(\mathbf{c}, \mathbf{a}) = \sum_{t \in T} a_t \left(c_t U_a^c(t) + (1 - c_t) U_a^u(t) \right) \quad (2)$$

We adopt a Stackelberg model in which the defender acts first and the attacker chooses a strategy after observing the defender's mixed strategy. Stackelberg games are common in security domains where attackers can surveil the defender strategy (Paruchuri et al. 2008). The standard solution concept is Strong Stackelberg Equilibrium (SSE) (Leitmann 1978; Breton, Alg, and Haurie 1988; von Stengel and Zamir 2004), in which the leader selects an optimal mixed strategy based on the assumption that the follower will choose an optimal response, breaking ties in favor of the leader.[1] There always exists an optimal pure-strategy response for the attacker, so we restrict our attention to this set in this paper.

Example: Consider a FAMS game with 5 targets (flights), $T = \{t_1, \ldots, t_5\}$, and three marshals of the same type, $r_1 = 3$. Let the set of feasible schedules be $S_1 = \{\{t_1, t_2\}, \{t_2, t_3\}, \{t_3, t_4\}, \{t_4, t_5\}, \{t_1, t_5\}\}$. The set of feasible joint schedules is shown below, where column $\mathbf{J}_1$ represents the joint schedule $\{\{t_1, t_2\}, \{t_3, t_4\}\}$.

$$\mathbf{P} = \begin{array}{c} \\ t_1: \\ t_2: \\ t_3: \\ t_4: \\ t_5: \end{array} \begin{array}{c} \begin{array}{ccccc} \mathbf{J}_1 & \mathbf{J}_2 & \mathbf{J}_3 & \mathbf{J}_4 & \mathbf{J}_5 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

[1] This tie-breaking rule is counter-intuitive, but the defender can make this response strictly optimal for the attacker by playing a strategy an infinitesimal ϵ away from the SSE strategy.

Each joint schedule in $\mathbf{J}$ assigns only 2 air marshals in this example, since no more than 1 FAM is allowed on any flight. Thus, the third air marshal will remain unused. Suppose all of the targets have identical payoffs $U_d^c(t) = 1$, $U_d^u(t) = -5$, $U_a^c(t) = -1$ and $U_a^u(t) = 5$. In this case, the optimal strategy for the defender randomizes uniformly across the joint schedules, $\mathbf{x} = \langle .2, .2, .2, .2, .2 \rangle$, resulting in coverage $\mathbf{c} = \langle .8, .8, .8, .8, .8 \rangle$. All pure strategies have equal payoffs for the attacker, given this coverage vector.

ASPEN Solution Approach and Related Work

The ERASER-C mixed-integer linear program (Kiekintveld et al. 2009) is the most recent algorithm developed for larger and more complex Stackelberg security games.Whereas previous work has focused on patrolling arbitrary topologies using Stackelberg games (Basilico, Gatti, and Amigoni 2009; Paruchuri et al. 2008), it has typically focused on a single defender. In contrast, ASPEN and ERASER-C focus on games with large numbers of defenders of different types, handling the combinatorial explosion in the defender's joint schedules. Unfortunately, as the authors note, ERASER-C may fail to generate a correct solution in cases where arbitrary schedules with more than two flights (i.e., multi-city tours) are allowed in the input, or when the set of flights cannot be partitioned into distinct sets for departure and arrival flights. For instance, ERASER-C incorrectly outputs the coverage vector $\mathbf{c} = \langle 1, 1, 1, 1, 1 \rangle$ for the example above. ERASER-C avoids enumerating joint schedules to gain efficiency, but loses the ability to correctly model arbitrary schedules. Furthermore, ERASER-C only outputs a coverage vector $\mathbf{c}$ and not the distribution $\mathbf{x}$ over joint schedules $\mathbf{J}$ necessary to implement the coverage in practice.

SPARS problems can be formulated as mixed-integer programs in which adversary strategies are represented by integer variables $\mathbf{a}$ with $a_t = 1$ if target t is attacked and 0 otherwise. Two key computational challenges arise in this formulation. First, the space of possible strategies (joint schedules) for the defender suffers from combinatorial explosion: a FAMS problem with 100 flights, schedules with 3 flights, and 10 air marshals has up to 100,000 schedules and $\binom{100000}{10}$ joint schedules. Second, integer variables are a well-known challenge for optimization. Branch and Price (Barnhart et al. 1994) is a framework for solving very large optimization problems that combines branch and bound search with column generation to mitigate both of these problems. This method operates on joint schedules (and not marginal probabilities, like ERASER-C), so it is able to handle arbitrary scheduling constraints directly.

An example of branch and price for our problem is shown in Figure 1, with the root representing the original problem. Branches to the left (gray nodes) set exactly one variable t_i in $\mathbf{a}$ to 1 and the rest to zero, resulting in a linear program that gives a lower bound on the overall solution quality. Branches to the right fix variable t_i to zero, leaving the remaining variables unconstrained. An upper bound on solution quality computed for each white node can be used to terminate execution without exploring all of the possible integer assignments. Solving the linear programs in each gray

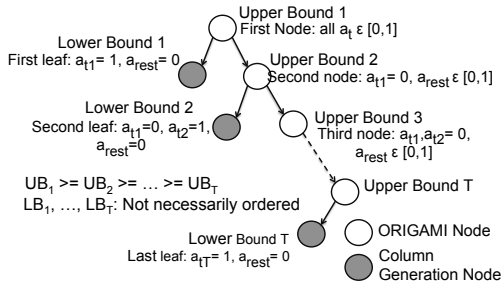


Figure 1: Working of Branch and Price

node normally requires enumerating all joint schedules for the defender. Column generation (i.e., pricing) is a technique that avoids this by iteratively solving a restricted *master problem*, which includes only a small subset of the variables, and a *slave problem*, that identifies new variables to include in the master problem to improve the solution.

Unfortunately, branch and price is not an "out of the box approach" and it has only recently begun to be applied in game-theoretic settings (Halvorson, Conitzer, and Parr 2009). We introduce a novel master-slave decomposition to facilitate column generation for SPARS, including a network flow formulation of the slave problem. We also show experimentally that conventional linear programming relaxations used for branch and bound perform poorly in this domain, and we replace them with novel techniques based on fast algorithms for security games without scheduling constraints.

ASPEN Column Generation

The linear programs at each leaf in Figure 1 are decomposed into into *master* and *slave* problems for column generation (see Algorithm 1). The master solves for the defender strategy $\mathbf{x}$, given a restricted set of columns (i.e., joint schedules) $\mathbf{P}$. The objective function for the slave is updated based on the solution of the master, and the slave is solved to identify the best new column to add to the master problem, using *reduced costs* (explained later). If no column can improve the solution the algorithm terminates.

Algorithm 1 Column generation

1. Initialize P
2. Solve Master Problem
3. Calculate reduced cost coefficients from solution
4. Update objective of slave problem with coefficients
5. Solve Slave Problem

if Optimal solution obtained **then**

6. Return $(\mathbf{x}, \mathbf{P})$

else

7. Extract new column and add to $\mathbf{P}$
8. Repeat from Step 2

Master Problem: The master problem (Equations 3 to 8) solves for the probability vector $\mathbf{x}$ that maximizes the defender reward (Table 1 describes the notation). This master problem operates directly on columns of $\mathbf{P}$, and the coverage vector $\mathbf{c}$ is computed from these columns as $\mathbf{Px}$. Constraints 4–6 enforce the SSE conditions that the players choose mutual best-responses, mirroring similar constraints in ERASER-C. The defender expected payoff (Equation 1) for target t is given by the t^{th} component of the column vector $\mathbf{DPx} + \mathbf{U}_d^u$ and denoted $(\mathbf{DPx} + \mathbf{U}_d^u)_t$. Similarly, the attacker payoff for target t is given by $(\mathbf{APx} + \mathbf{U}_a^u)_t$. Constraints 4 and 5 are active only for the single target t^* attacked ($a_{t^*} = 1$). This target must be a best-response, due to Constraint 6.

Table 1: Notation

Variable	Definition	Dimension
$\mathbf{P}$	Mapping between T and J	$\|T\| \times \|J\|$
$\mathbf{x}$	Probability Distribution over J	$\|J\| \times 1$
$\mathbf{a}$	Attack vector	$\|T\| \times 1$
d	Defender Reward	-
k	Adversary Reward	-
$\mathbf{d}$	Column vector of d	$\|T\| \times 1$
$\mathbf{k}$	Column vector of k	$\|T\| \times 1$
$\mathbf{D}$	Diag. matrix of $U_d^c(t) - U_d^u(t)$	$\|T\| \times \|T\|$
$\mathbf{A}$	Diag. matrix of $U_a^c(t) - U_a^u(t)$	$\|T\| \times \|T\|$
$\mathbf{U}_d^u$	Vector of values $U_d^u(t)$	$\|T\| \times 1$
$\mathbf{U}_a^u$	Vector of values $U_a^u(t)$	$\|T\| \times 1$
M	Huge Positive constant	-

$$\max \quad d \tag{3}$$

$$\text{s.t.} \quad \mathbf{d} - \quad \mathbf{DPx} - \mathbf{U}_d^u \leq (\mathbf{1} - \mathbf{a})M \tag{4}$$

$$\mathbf{k} - \quad \mathbf{APx} - \mathbf{U}_a^u \leq (\mathbf{1} - \mathbf{a})M \tag{5}$$

$$\mathbf{APx} + \mathbf{U}_a^u \leq \mathbf{k} \tag{6}$$

$$\sum_{j \in J} x_j = 1 \tag{7}$$

$$\mathbf{x}, \mathbf{a} \geq 0 \tag{8}$$

Slave Problem: The slave problem finds the best column to add to the current columns in $\mathbf{P}$. This is done using *reduced cost*, which captures the total change in the defender payoff if a candidate column is added to $\mathbf{P}$. The candidate column with minimum reduced cost improves the objective value the most (Bertsimas and Tsitsiklis 1994). The reduced cost $\bar{c}_j$ of variable x_j, associated with column $\mathbf{P_j}$, is given in Equation 9, where $\mathbf{w}, \mathbf{y}, \mathbf{z}$ and h are dual variables of master constraints 4, 5, 6 and 7 respectively. The dual variable measures the influence of the associated constraint on the objective, and can be calculated using standard techniques.

$$\bar{c}_j = \quad \mathbf{w}^T(\mathbf{DP_j}) + \mathbf{y}^T(\mathbf{AP_j}) - \mathbf{z}^T(\mathbf{AP_j}) - h \tag{9}$$

An inefficient approach would be to iterate through all of the columns and calculate each reduced cost to identify the best column to add. Instead, we formulate a minimum cost network flow (MCNF) problem that efficiently finds the optimal column. Feasible flows in the network map to feasible joint schedules in the SPARS problem, so the scheduling constraints are captured by this formulation. For a SPARS instance we construct the MCNF graph G as follows.

A source node source_λ with supply r_λ is created for each defender type $\lambda \in \Lambda$. A single sink node has demand $\sum_{\lambda \in \Lambda} r_\lambda$. Targets in schedule s for resource λ are represented by pairs of nodes $(a_{s_\lambda,t}, b_{s_\lambda,t})$ with a connecting link (so each target corresponds to many nodes). For every schedule $s_\lambda \in S_\lambda$ we add a path from the source to the sink: $\langle \text{source}_\lambda, a_{s,t_{i_1}}, b_{s,t_{i_1}}, a_{s,t_{i_2}}, \ldots, b_{s,t_{i_L}}, \text{sink} \rangle$. The capacities on all links are set to 1, and the default costs to 0. A dummy flow with infinite capacity is added to represent the possibility that some resources are unassigned. The number of resources assigned to t in a column $\mathbf{P_j}$ is computed as: $\text{assigned}(t) = \sum_{s \in S} \text{flow}[\text{link}(a_{s,t}, b_{s,t})]$. Constraints are added to G so $\text{assigned}(t) \leq 1$ for all targets t.

A partial graph G for our earlier example is shown in Figure 2, showing paths for 3 of the 5 schedules. The paths correspond to schedules $\{t_1, t_2\}$, $\{t_2, t_3\}$ and $\{t_1, t_5\}$. The supply and demand are both 3, corresponding to the number of available FAMS. Double-bordered boxes mark the flows used to compute $\text{assigned}(t_1)$ and $\text{assigned}(t_2)$. Every joint schedule corresponds to a feasible flow in G. For example, the joint schedule $\{\{t_2, t_3\}, \{t_1, t_5\}\}$ has a flow of 1 unit each through the paths corresponding to schedules $\{t_2, t_3\}$ and $\{t_1, t_5\}$, and a flow of 1 through the dummy. Similarly, any feasible flow through the graph G corresponds to a feasible joint schedule, since all resource constraints are satisfied.

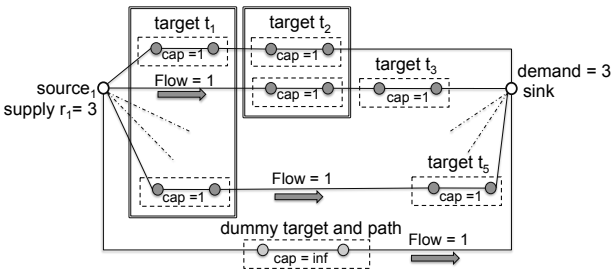


Figure 2: Example Network Graph

It remains to define link costs such that the cost of a flow is the reduced cost for the joint schedule. We decompose $\bar{c}_j$ into a sum of cost coefficients per target, $\hat{c}_t$, so that $\hat{c}_t$ can be placed on links $(a_{s,t}, b_{s,t})$ for all targets t. $\hat{c}_t$ is defined as $w_t.D_t + y_t.A_t - z_t.A_t$ where w_t, y_t and z_t are t^{th} components of $\mathbf{w}, \mathbf{y}$ and $\mathbf{z}$. D_t is equal to $U_d^c(t) - U_d^u(t)$ and $A_t = U_a^c(t) - U_a^u(t)$. The overall objective given below for the MCNF problem sums the contributions of the reduced cost from each individual flow and subtracts the dual variable h. If this is non-negative, no column can improve the master solution, otherwise the optimal column (identified by the flow) is added to the master and the process iterates.

$$\min_{\text{flow}} \sum_{(a_{s,t}, b_{s,t})} \hat{\text{c}}_\text{t}.\text{flow}[(a_{s,t}, b_{s,t})] - h$$

Improving Branching and Bounds

ASPEN uses branch and bound to search over the space of possible attacker strategies. A standard technique in branch and price is to use LP relaxation, i.e. allow the integer variables to take on arbitrary values, to give an optimistic bound on the objective value of the original MIP for each internal node. Unfortunately, our experimental results show that this generic method is ineffective in our domain. We introduce ORIGAMI-S, a novel branch and bound heuristic for SPARS based on ORIGAMI (Kiekintveld et al. 2009), which is an efficient solution method for security games without scheduling constraints and heterogeneous resources. We use ORIGAMI-S to solve a relaxed version of SPARS, and integrate this in ASPEN to give bounds and select branches.

$$\min \quad k \tag{10}$$

$$\mathbf{U}_a(\mathbf{c}) = \quad \mathbf{A}\mathbf{c} + \mathbf{U}_a^u \tag{11}$$

$$\mathbf{0} \leq \mathbf{k} - \mathbf{U}_a(\mathbf{c}) \leq \quad (\mathbf{1} - \mathbf{q}) \cdot M \tag{12}$$

$$c_t = \quad \textstyle\sum_{s \in S} \tilde{c}_{t,s} \qquad \forall t \in T \tag{13}$$

$$\sum_{s \in S_\lambda} \tilde{c}_{T_\lambda(s),s} \leq \quad r_\lambda \qquad \forall \lambda \in \Lambda \tag{14}$$

$$\sum_{t \in T} c_t \leq \quad L \cdot \textstyle\sum_{\lambda \in \Lambda} r_\lambda \tag{15}$$

$$\mathbf{c} \leq \quad \mathbf{q} \tag{16}$$

$$\mathbf{q} \in \{0,1\}, \quad \mathbf{c}, c_{t,s} \in [0,1] \ \ \forall t \in T, s \in S \tag{17}$$

The ORIGAMI-S model is given in Equations 10–17. It minimizes the attacker's maximum payoff (Equations 10–12). The vector $\mathbf{q}$ represents the *attack set*, and is 1 for every target that gives the attacker maximal expected payoff (Equation 12). The remaining nontrivial constraints restrict the coverage probabilities. ORIGAMI-S defines a set of probabilities $\tilde{c}_{t,s}$ that represent the coverage of each target t in each schedule $s \in S_\lambda$. The total coverage c_t of target t is the sum of coverage on t across individual schedules (Equation 13). We define a set T_λ which contains one target from each schedule $s \in S_\lambda$. The total coverage assigned by resource type λ is upper-bounded by r_λ (Equation 14), analogous to the constraint that the total flow from a source in a network flow graph cannot be greater than the available supply. Total coverage is also bounded by multiplying the number of resources by the *maximum* size of any schedule (L) in Equation 15. The defender can never benefit by assigning coverage to nodes outside of the attack set, so these are constrained to 0 (Equation 16).

ORIGAMI-S is solved once at the beginning of ASPEN, and targets in the attack set are sorted by expected defender reward. The maximum value is an initial upper bound on the defender reward. The first leaf node that ASPEN evaluates corresponds to this maximum valued target (i.e, setting its attack value to 1), and a solution is found using column generation. This solution is a lower bound of the optimal solution, and the algorithm stops if this lower bound meets the ORIGAMI-S upper bound. Otherwise, a new upper bound from the ORIGAMI-S solution is obtained by choosing the second-highest defender payoff from targets in the attack set, and ASPEN evaluates the corresponding leaf node. This process continues until the upper bound is met, or the available nodes in the search tree are exhausted.

Theorem 1 *The defender payoff, computed by ORIGAMI-S, is an upper bound on the defender's payoff for the corresponding SPARS problem. For any target not in the attack*

set of ORIGAMI-S, the restricted SPARS problem in which this target is attacked is infeasible.

Proof Sketch: ORIGAMI and ORIGAMI-S both minimize the maximum attacker payoff over a set of feasible coverage vectors. If there are no scheduling constraints, this also maximizes the defender's policy (Kiekintveld et al. 2009). Briefly, the size of the attack set in the solution is maximized, and the coverage probability on each target in the attack set is also maximal. Both of these weakly improve the defender's payoff because adding coverage to a target is strictly better for the defender and worse for the adversary.

ORIGAMI-S makes optimistic assumptions about the coverage probability the defender can allocate by taking the maximum that could be achieved by *any* legal joint schedule and allowing it to be distributed arbitrarily across the targets, ignoring the scheduling constraints. To see this, consider the marginal probabilities $\mathbf{c}^*$ of any legal defender strategy for SPARS. There is at least one feasible coverage strategy for ORIGAMI that gives the same payoff for the defender. Constraints 13 and 17 are satisfied by $\mathbf{c}^*$, because they are also constraints of SPARS. Each variable $\tilde{c}_{T_\lambda(s),s}$ in the set defined for Constraint 14 belongs to a single schedule associated with resource type λ, and at most r_λ of these can be selected in any feasible joint schedule, so this constraint must also hold for $\mathbf{c}^*$. Constraint 15 must be satisfied because it assumes that each available resource covers the largest possible schedule, so it generally allows excess coverage probability to be assigned. Finally, constraint 16 may be violated by $\mathbf{c}^*$ for some target t. However, the coverage vector with coverage identical to $\mathbf{c}^*$ for all targets in the ORIGAMI-S attack set and 0 coverage outside the attack set has identical payoffs (since these targets are never attacked).

Experimental Results

Comparison on FAMS domain: We compare the runtime performance of ASPEN, branch and price without the ORIGAMI-S heuristic (BnP) and ERASER-C. For this experiment we generate random instances of FAMS problems (Kiekintveld et al. 2009) with schedules of size two, with one departure flight and one arrival flight drawn from disjoint sets (for correct operation of ERASER-C). We vary the number of targets, defender resources, and schedules.

ERASER-C outputs a coverage vector $\mathbf{c}$, so to obtain a probability distribution $\mathbf{x}$ over joint schedules $\mathbf{J}$ we need an additional method. Instead, we modify our column generation approach, using a revised master problem with the objective of minimizing the difference $\mathbf{Px}-\mathbf{c}$. The slave problem remains unchanged. In our results, we present runtimes for ERASER-C that include the generation of a joint distribution in order to provide a fair comparison with ASPEN.

All experiments were based on 15 sample games, and problem instances that took longer than 30 minutes to run were terminated. Results varying the number of defender resources are shown in Figure 3(a). The y-axis shows the runtime in seconds on the *logarithmic scale*. The x-axis shows the number of resources. ASPEN is the fastest of the three algorithms. The effectiveness of the ORIGAMI-S bounds and branching are clear in the comparison with standard BnP method. Since ASPEN solves a far more general set of security games (SPARS), we would not expect it to be competitive with ERASER-C in its specialized domain. However, ASPEN was 6 times faster than ERASER-C in some instances. This improvement over ERASER-C was an unexpected trend, and can be attributed to the number of columns generated by the two approaches (Table 2). We observe similar results in the second and third data sets presented in Figures 3(b) and 3(c).

Table 2: Number of columns: 200 targets, 600 schedules

Resources	ASPEN	ERASER-C	BnP (max. 30 mins)
10	126	204	1532
20	214	308	1679
30	263	314	1976
40	227	508	1510
50	327	426	1393

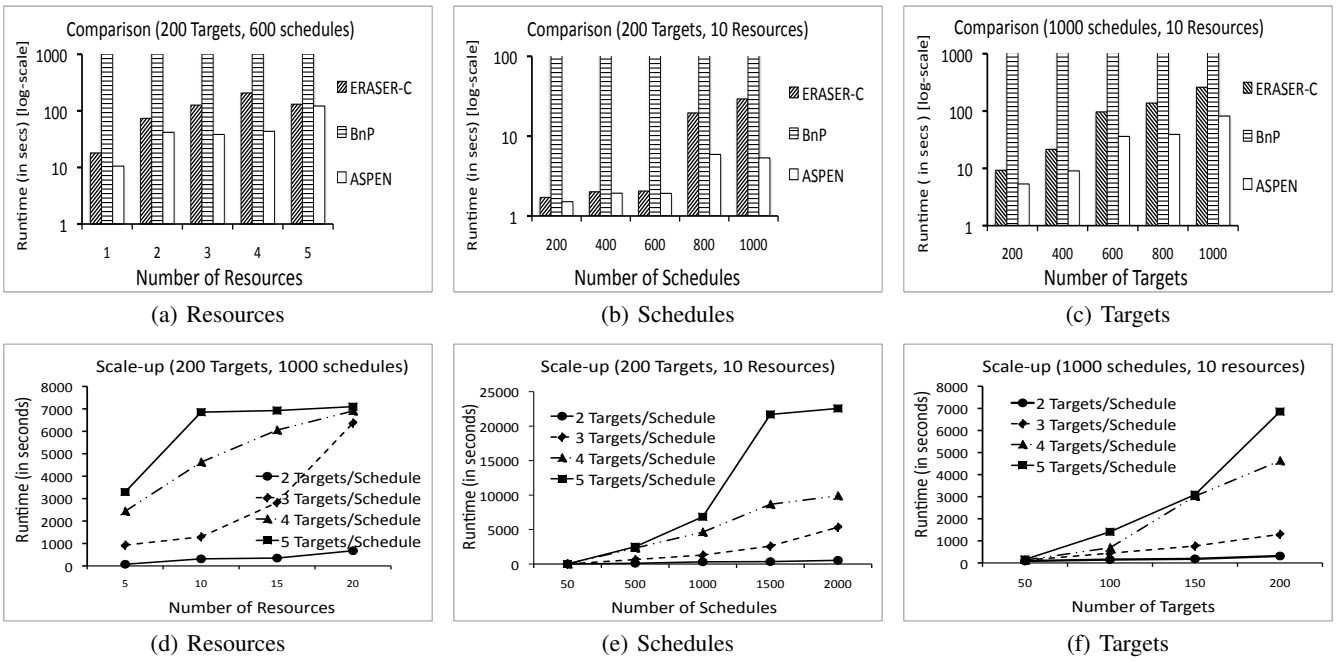
Table 3: Number of columns: 200 targets, 1000 schedules

Resources	3 Tar. / sch.	4 Tar. / sch.	5 Tar. / sch.
5	456	518	658
10	510	733	941
15	649	920	1092
20	937	1114	1124

ASPEN on Large SPARS Instances: We also evaluate the performance of ASPEN on arbitrary scheduling problems as the size of the problem is varied to include very large instances. No comparisons could be made because ERASER-C does not handle arbitrary schedules and the only correct algorithms known, DOBSS (Paruchuri et al. 2008) and BnP, do not scale to these problem sizes. We vary the number of resources, schedules, and targets as before. In addition, we vary the number of targets per schedule for each of the three cases to test more complex scheduling problems. Figure 3(d) shows the runtime results with 1000 feasible schedules and 200 targets, averaged over 10 samples. The x-axis shows the number of resources, and the y-axis shows the runtime in seconds. Each line represents a different number of schedules per target. The number of joint schedules in these instances can be as large as 10^{23} ($\binom{1000}{10} \approx 2.6\times10^{23}$). Interestingly, the runtime does not increase much when the number of resources is increased from 10 to 20 when there are 5 targets per schedules. Column 4 of Table 3 illustrates that the key reason for constant runtime is that the average number of generated columns remains similar. Similarly, for a fixed number of resources, we observe an increase in runtime for more complex schedules that corresponds with an increase in the number of columns generated. The other two experiments, Figure 3(e) and 3(f) also show similar trends.

Conclusions

We present a branch and price method, ASPEN, for solving large-scale Stackelberg security games with arbitrary constraints. ASPEN incorporates several novel contributions, including a decomposition of SPARS to enable column generation and the integration of ORIGAMI-S to substantially



(a) Resources (b) Schedules (c) Targets

(d) Resources (e) Schedules (f) Targets

Figure 3: Runtime Results

speed up the branch and bound search. Experimental results show that ASPEN is competitive with ERASER-C for the restricted class of games where ERASER-C is applicable. More importantly, ASPEN solves far more general instances of scheduling problems where ERASER-C and other existing techniques fail. ASPEN is also substantially faster than a standard implementation of branch and price for this domain. This work contributes to a very new area of work that applies techniques used in large-scale optimization to game-theoretic problems—an exciting new avenue with the potential to greatly expand the reach of game theory.

Acknowledgements

We would also like to thank Jason Tsai and the reviewers for their comments and suggestions. 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. F. Ordóñez would also like to acknowledge the support of Fondecyt, through Grant No. 1090630.

References

Barnhart, C.; Johnson, E.; Nemhauser, G.; Savelsbergh, M.; and Vance, P. 1994. Branch and price: Column generation for solving huge integer programs. In *Operations Research*, volume 46, 316–329.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, 500–503.

Bertsimas, D., and Tsitsiklis, J. N. 1994. *Introduction to Linear Optimization*. Athena Scientific.

Breton, M.; Alg, A.; and Haurie, A. 1988. Sequential Stackelberg equilibria in two-person games. *Optimization Theory and Applications* 59(1):71–97.

Conitzer, V., and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *ACM EC-06*, 82–90.

Halvorson, E.; Conitzer, V.; and Parr, R. 2009. Multi-step multi-sensor hider-seeker games. In *IJCAI*, 336–341.

Kiekintveld, C.; Jain, M.; Tsai, J.; Pita, J.; Tambe, M.; and Ordonez, F. 2009. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, 689–696.

Korzhyk, D.; Conitzer, V.; and Parr, R. 2010. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, *to appear*.

Leitmann, G. 1978. On generalized Stackelberg strategies. *Optimization Theory and Applications* 26(4):637–643.

Paruchuri, P.; Pearce, J. P.; Marecki, J.; Tambe, M.; Ordonez, F.; and Kraus, S. 2008. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *AAMAS-08*, 895–902.

Pita, J.; Bellamane, H.; Jain, M.; Kiekintveld, C.; Tsai, J.; Ordez, F.; and Tambe, M. 2009. Security applications: Lessons of real-world deployment. In *SIGECOM Issue 8.2*.

Tsai, J.; Rathi, S.; Kiekintveld, C.; Ordez, F.; and Tambe, M. 2009. IRIS a tool for strategic security allocation in transportation networks. In *AAMAS-09 (Industry Track)*, 37–44.

von Stengel, B., and Zamir, S. 2004. Leadership with commitment to mixed strategies. Technical Report LSE-CDAM-2004-01, CDAM Research Report.