IRIS - A Tool for Strategic Security Allocation in Transportation Networks

Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld,
Fernando Ordóñez, Milind Tambe
University of Southern California, Los Angeles, CA 90089
{jasontts, srathi, kiekintv, fordon, tambe} @usc.edu

ABSTRACT

Security is a concern of major importance to governments and companies throughout the world. With limited resources, complete coverage of potential points of attack is not possible. Deterministic allocation of available law enforcement agents introduces predictable vulnerabilities that can be exploited by adversaries. Strategic randomization is a game theoretic alternative that we implement in Intelligent Randomization In Scheduling (IRIS) system, a software scheduling assistant for the Federal Air Marshals (FAMs) that provide law enforcement aboard U.S. commercial flights.

In IRIS, we model the problem as a Stackelberg game, with FAMS as leaders that commit to a flight coverage schedule and terrorists as followers that attempt to attack a flight. The FAMS domain presents three challenges unique to transportation network security that we address in the implementation of IRIS. First, with tens of thousands of commercial flights per day, the size of the Stackelberg game we need to solve is tremendous. We use ERASER-C, the fastest known algorithm for solving this class of Stackelberg games. Second, creating the game itself becomes a challenge due to number of payoffs we must enter for these large games. To address this, we create an attribute-based preference elicitation system to determine reward values. Third, the complex scheduling constraints in transportation networks make it computationally prohibitive to model the game by explicitly modeling all combinations of valid schedules. Instead, we model the leader's strategy space by incorporating a representation of the underlying scheduling constraints.

The scheduling assistant has been delivered to the FAMS and is currently undergoing testing and review for possible incorporation into their scheduling practices. In this paper, we discuss the design choices and challenges encountered during the implementation of IRIS.

Categories and Subject Descriptors

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

General Terms

Security, Design, Theory

Cite as: IRIS – A Tool for Strategic Security Allocation in Transportation Networks, Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordóñez, Milind Tambe, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 37 – 44

Keywords

Security of Agent Systems, Game Theory, Bayesian and Stackelberg Games

1. INTRODUCTION

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. In 2001, the 9/11 attack on the World Trade Center in New York City via commercial airliners resulted in USD 27.2 billion of direct short term costs [11] as well as a government-reported 2,974 lives lost. The 2004 Madrid commuter train bombings resulted in 191 lives lost, 1755 wounded, and an estimated cost of 212 million Euros [5]. Finally, in the 2005 London subway and bus bombings, 52 innocent lives were lost, 700 others injured, and an estimated economic cost of 2 billion pounds [17].

In addition to security checkpoints prior to boarding the vehicle, patrols aboard the vehicles are another key defensive measure used by many organizations to provide law enforcement in these domains [4, 9]. In all of these networks, there are hundreds or thousands of vehicles to protect, making it difficult to create patrol schedules at all. Furthermore, since motivated aggressors will attempt to observe law enforcement patterns and try to exploit the schedule, law enforcement organizations have embraced the use of randomization in their scheduling practices. Also, in all of these networks, it is not possible to simply assign law enforcement personnel to vehicles in isolation without considering the route the vehicle takes as well as departure and arrival times. As a simple example, we cannot ask an officer to board and protect a 10:00AM flight from New York to Los Angeles as well as a 10:30AM flight from London to Chicago.

Thus, three primary challenges must be overcome by any method of randomizing law enforcement in transportation networks: (i) the runtime of solution methods are often prohibitive due to the scale of the problems; (ii) the number of inputs required under many solution methods is also often prohibitive due to the scale of the problem; (iii) the scheduling constraints of a transportation network must be obeyed in any schedule created.

One obvious method of randomization is a dice-roll approach where we protect each target with the same probability. However, in a realistic situation, some targets are valued more highly than others. If a security organization patrols every target with equal probability, an intelligent attacker will undoubtedly choose the target with the highest value, as this will have the highest expected payoff. Knowing this, the defenders should place more emphasis on protecting the high-value target to decrease the expected value that the attacker has for attacking it. Indeed, we could randomize based on the target values instead of uniformly. This strategy, which we show to be superior to a uniform random strategy, fails to account for how the attacker will update his strategy based on knowledge of the defender's strategy.

An alternative methodology would be to use a non-game theoretic approach to modeling and solving the problem, such as learning and MDP's as done by Ruan et al. [15]. 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. Such a framework could account for the differing values of targets in the transportation network problem and their use of multiple patrol routes to introduce unpredictability to insurgents seems to address the majority of issues we face. However, their strategy still fails to account for attackers updating their strategy after observing the defender's strategy.

This presents a problem for the security policy because it was generated based on a given frequency of attacker appearances in each of the locations. When the security policy is enacted, however, an intelligent attacker that observes the policy will adjust his behavior and begin to attack locations that were weighted with less security due to the relative infrequency of attacks there previously.

To alleviate this problem, another method would be to have a human create randomized schedules and weight his coverage of the targets based on their value as well as beliefs about attacker response behavior. However, studies have shown that humans are poor randomizers and can fall into predictable patterns [20]. Furthermore, in large domains like transportation networks, the man-hours of labor that must be spent to perform a randomization that factors varying target values and attacker behavior by hand would likely be prohibitively large.

Another possibility would be to use a game-theoretic method. The ARMOR program that is currently deployed at Los Angeles International Airport is one possible tool [14]. ARMOR randomizes checkpoints and canine patrols using a Bayesian Stackelberg game model and solves for the optimal randomized strategy for the defender. When we model the security game as a Bayesian Stackelberg game, we are able to input different payoff values for each of the targets, which allows us to account for varying target values in the real domain. The use of a set distribution of coverage probabilities prevents human predictability from compromising the system. Finally, the Stackelberg framework inherently accounts for the attacker's response to the defender's policy.

However, while the approach provides a starting point for addressing the transportation network problem, the three challenges unique to this domain remain. The DOBSS algorithm at the heart of ARMOR does not scale well and would not be able to handle the size of transportation network problems in a reasonable timeframe [10]. Scale also presents an issue for inputing values. In the ARMOR system, domain experts individually specified payoff values for each of the targets directly. In a transportation network, this could come to tens of thousands of values versus the 10s of values that we had at LAX. Also, the method of game modeling we used in ARMOR will not realistically be able to handle the hard scheduling constraints that we must account for. We outline the primary differences between the LAX domain and the FAMS domain in more detail in Section 2.

We provide three key contributions in the implementation of IRIS that revolve around the application of new representations and algorithms to the transportation network domain that allow us to generate randomized schedules in reasonable timeframes. First, we use a new, more efficient, MILP for modeling the Stackelberg games to overcome runtime issues that arise due to scale. Second, we introduce an attribute-based preference elicitation system for risk/reward assessment of potential targets that allows us to avoid requiring user inputs for every individual target [7]. Finally, we model the game with defender actions that incorporate the scheduling constrants, which allows us to accurately model and resolve the scheduling constraints issue without combinatorially exploding the actionspace.

Our work is an application of recent advances in multi-agent research in security/patrolling problems and optimal equilibrium algorithms for solving games [10] to a major security problem in the real world. Patrolling problems in general have received much attention in multi-agent research due to the variety of applicable domains such as robot patrol and border patrolling of large areas [4, 20]. Game theoretic advances, specifically in solving Stackelberg games [8, 16], have ultimately led to successful work in security policy randomization using a multi-agent systems framework [12, 13] and the recent deployment of the ARMOR system at LAX. As we develop more sophisticated techniques for solving these games, we will be able to address larger and more difficult problems in the real world that we could not previously handle. The creation of IRIS for the FAMS is an example of new advances in multi-agent research allowing us to solve a real-world problem that could not realistically be solved with previous techniques.

2. FEDERAL AIR MARSHAL SERVICE

The Federal Air Marshal Service (FAMS) domain is a particular instance of a transportation network security problem. The FAMS places undercover law enforcement personnel aboard flights originating in and departing from the United States to dissuade potential aggressors and prevent an attack should one occur [2]. Strategic randomization based on game-theoretic principles provides a possible method for creating a coverage schedule that avoids the pitfalls of a deterministic strategy.

Variation in target risk and value is extremely apparent in the FAMS domain. While many flights are overbooked, there are some flights with no passengers that are simply flying to a destination for a subsequent flight. Similarly, while some flights fly over densely populated areas, others do not. Also, while a particular subset of flights might be low risk at one point in time, they could become high risk at another time due to a special event [1]. We must somehow account for these variations between flights in our randomization of law enforcement forces if we hope to utilize resources effectively. We model this as a Stackelberg game, as in ARMOR [14].

However, the three problems unique to transportation networks remain and cannot be handled by the ARMOR system. Whereas ARMOR handles 10 terminals at the LA airport, the FAMS must protect tens of thousands of commercial flights per day. As shown in Kiekintveld et al. [10], the DOBSS algorithm at the heart of ARMOR cannot handle problems of this magnitude. Also, in ARMOR, domain experts have to enter four payoff values for each of the 10 targets in the domain. Asking the same thing of the FAMS would require tens of thousands of values to be entered by hand, likely introducing user-error and a potentially prohibitive time burden for each scheduling cycle. Finally, at LAX, we randomize fixed checkpoint locations and canine patrols that only need to be given a set of times and locations to be. The only constraints on resources is the number of checkpoints and canines available to the system. In the FAMS domain, we have hard scheduling constraints due to the fact that they are patrolling moving vehicles and not fixed locations. For example, asking law enforcement personnel to cover a flight from El Paso to New York might not be possible without someone first flying to El Paso from somewhere else if there are simply no personnel that normally reside in El Paso. In generating a schedule for coverage, we need to be mindful of such constraints.

Thus, the FAMS domain presents all three challenges that must be overcome in a practical implementation of a randomized scheduling assistant for a transportation network: (i) runtime issues due to scale; (ii) preference elicitation issues due to scale; (iii) scheduling constraints.

3. BACKGROUND

The primary contribution of this work is in the application of game-theoretic methods to the FAMS domain and overcoming challenges encountered during implementation. Here, we lay the groundwork for our decision to use Stackelberg games in this effort. In Section 5 we will go on to detail how we map the FAMS domain into a Stackelberg game and the challenges encountered during the process.

A Stackelberg game allows us to create a randomized schedule while taking into account varying values of the targets and the fact that adversaries act with prior knowledge of a security force's policies. Since these are the basic challenges that we must overcome in solving security games, Stackelberg games form the basis of our approach. First, we describe Stackelberg games and the subtleties introduced by the leader/follower paradigm. Next, we outline their application to the security domain specifically. Finally, we give a brief overview of the ERASER-C algorithm that we use to solve these games.

3.1 Stackelberg Games

While Stackelberg games were first introduced to study duopoly competition [18], they have since been recognized as a general model of leadership and commitment. In Stackelberg games, there are two players: a leader and a follower. The leader commits to a strategy first, after which the follower makes his strategy choice. While this may appear completely detrimental for the leader, since the follower can act with full knowledge of the leader's strategy, the leader actually has a signficant advantage as well. To see this, consider a game with the payoff table as shown in Table 1. If we played this as a simultaneous game, the only pure-strategy Nash equilibrium is when the row player plays A and the column player plays C, giving the row player a payoff of 2. However, if the row player was the leader and the column player the follower in a Stackelberg game, the leader can commit to a uniform mixed strategy of playing each A and B half the time. This will force the optimizing follower to choose action D, giving the leader an expected payoff of 3.5.

The fact that the leader is able to force the follower into a specific subset of the actionspace potentially gives the leader an advantage in Stackelberg games. Indeed, it has been shown that being able to commit to a randomized mixed strategy always weakly increases the leader's payoff in equilibrium profiles of the game [19].

	C	D
A	2,1	4,0
B	1,0	3,2

Table 1: Payoff table for an example normal form game.

3.2 Stackelberg Security Games

We now discuss how we map security games onto this framework. In a security game, there exist two players; an attacker and a defender that may have multiple types. As stated before, a defender must perpetually defend the site in question, whereas the attacker is able to observe the defender's strategy and attack when success seems most likely. This fits neatly into the description of a Stackelberg game if we map the attacker to the follower's role and the defender to the leader's role [3, 6].

We can define the actions for the defender as the set of targets that he chooses to defend. So, as shown in Table 2, action '1' might actually represent a decision to protect sites 1, 2, and 3. We refer to this as a coverage set. For the attacker, the actions represent which site to attack. Payoffs can be calculated based on the action choices. If the attacker attacks a target that is in the defender's coverage set, then the attacker receives a negative payoff and the defender receives a positive payoff where the magnitudes depend on the value of the target. If the attacker attacks a target outside the defender's coverage set, then the attacker receives a positive payoff and the defender receives a negative payoff. This is the approach used in the ARMOR program at LAX [14]. ARMOR used the DOBSS algorithm [12] to solve the resulting game for an optimal mixed strategy for the defender, which would assign each coverage set a probability of being used.

3.3 ERASER / ERASER-C

We briefly describe the ERASER (Efficient Randomized Allocation of SEcurity Resources) and ERASER-C (Constrained) methods for modeling these Stackelberg security games. Like DOBSS, ERASER is a mixed-integer linear program. However, the additional insight that ERASER exploits is the fact that the payoffs in these games do not depend on the coverage set being implemented, but simply on whether or not the attacked target is in it. For example, while we might have different types of defenders, the particular type of defender that protects a target does not impact the payoff achieved. Also, whether or not an unattacked target is covered has no impact on the payoff either. This means that from a payoff perspective, many coverage sets are actually identical and we can represent the actions in terms of targets instead of coverage sets.

In the previous representation which DOBSS operates on, if we had 3 guards that could defend any of 10 sites, we would have $\binom{10}{3} = 120$ actions. In the new representation, we simply have 10 actions, one for each target, and a probability distribution across the targets that indicates how we distribute the 3 guards across them. In Tables 2 and 3, we show the actionspaces for the example outlined with DOBSS and then with ERASER.

Action	Targets	Prob.
1	1,2,3	p_1
2	1,2,4	p_2
3	1,2,5	p_3
...	...	...
120	8,9,10	p_{120}

Table 2: DOBSS.

Action	Targets	Prob.
1	1	p_1
2	2	p_2
3	3	p_3
...	...	...
10	10	p_{10}

Table 3: ERASER.

The first line of Table 2 indicates that, in the DOBSS representation, the first action has the three guards protecting targets 1, 2, and 3 and we use the first action with a probability of p_1. The first line of Table 3 indicates that, in the ERASER representation, the first action has a guard protecting target 1 with a probability of p_1. As can be seen, this compact representation idea combinatorially shrinks the actionspace of the game, providing tremendous speed increases in solving these games. Although ERASER requires some postprocessing to obtain a solution in the same form as the DOBSS game, the postprocessing is compartively cheap.

ERASER-C takes this idea into a domain with scheduling constraints. Suppose we have 20 targets and 3 guards and the requirement that targets must be guarded as pairs in the set $S = \{(1,2),(3,4),...,(19,20)\}$. In DOBSS, we would need to generate actions that represented all combinations of all valid schedules instead of combinations of all valid targets as in the previous example. In ERASER-C, we perform a similar aggregation to ERASER and also include a mapping of schedules to targets. We then define a defender's strategy to be an assignment of resources to schedules instead of an assignment of resources to individual targets as in the previous case. In Tables 4 and 5, we show the actionspaces for the example outlined with DOBSS and then with ERASER-C.

Action	Targets	Probability
1	(1,2),(3,4),(5,6)	p_1
2	(1,2),(3,4),(7,8)	p_2
3	(1,2),(3,4),(9,10)	p_3
...	...	...
120	(15,16),(17,18),(19,20)	p_{120}

Table 4: DOBSS.

Action	Targets	Probability
1	(1,2)	p_1
2	(3,4)	p_2
3	(5,6)	p_3
...	...	...
10	(19,20)	p_{10}

Table 5: ERASER-C.

This exponentially more compact representation allows ERASER-C to solve this class of security games with scheduling constraints exponentially faster than DOBSS and is discussed in more detail in Kiekintveld et al. [10]. This makes it possible for us to solve these large transportation network games in reasonable timeframes, which we evaluate in Section 7.1.

4. SYSTEM ARCHITECTURE

The IRIS system consists of an input module, a back-end module, a display/output module, and a project management module. Figure 1 shows a generic diagram of the system, with grey boxes representing modules and white boxes within them representing the components of the modules. We now describe the modules as well as the particular instantiations of these modules in the FAMS domain.

The input module is composed of four classes of inputs that are required by the system in order to generate a representative Stackelberg game and create an optimal schedule. The first input is the resource data. In the FAMS domain, this is the coverage ability and number of FAMs, which we have chosen to model as locations, where each location has a specific number of FAMs who can cover some subset of flights. The second input is the target data. In this case, this is the flight data and includes all relevant information about flights that the user considers during scheduling, such as flight number, carrier, origin, destination, aircraft type, etc.. The third input is data required for risk assessment which will be used by our attribute-based risk analysis engine that we will discuss in Section 5.1. Figures such as number of passengers, flight path, city of origin, and city of destination are examples of relevant risk data in the FAMS domain. Finally, we allow inputs for data that can be used in the GUI to aid the user's navigation and understanding of the system's output. This might include alternative naming schemes for airports and airlines, for example. A sample input screen is shown in Figure 2.

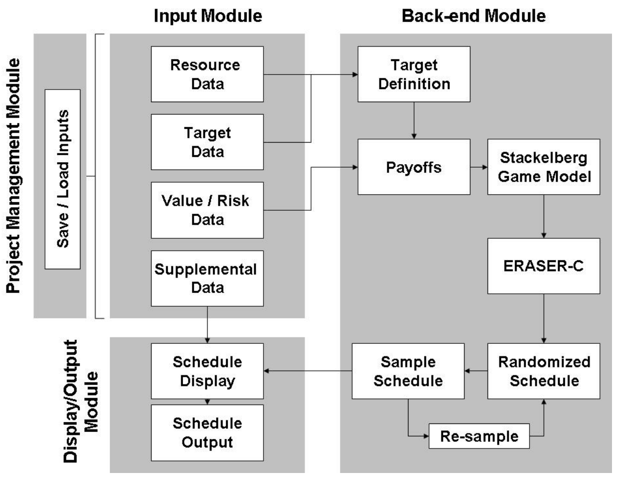
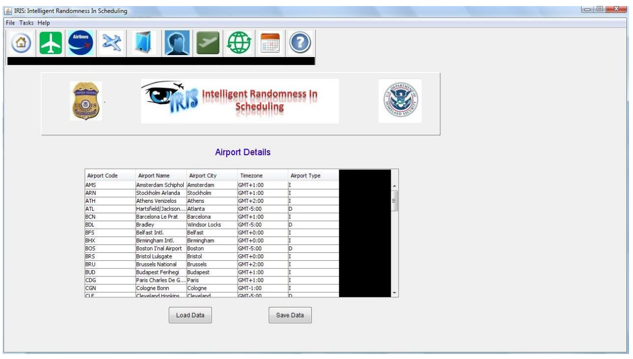


Figure 1: Generic diagram of IRIS system.


IRIS: Intelligent Randomness In Scheduling
File Tasks Help
Intelligent Randomness In Scheduling
Airport Details
Load Data
Save Data

Airport Code	Airport Name	Airport City	Timezone	Airport Type
AMS	Amsterdam Schiphol	Amsterdam	GMT+1:00	I
ARN	Stockholm Arlanda	Stockholm	GMT+1:00	I
ATH	Athens Venizelos	Athens	GMT+2:00	I
ATL	Hartsfield/Jackson...	Atlanta	GMT-5:00	D
BCN	Barcelona Le Prat	Barcelona	GMT+1:00	I
BDL	Bradley	Windsor Locks	GMT-5:00	D
BFS	Belfast Intl.	Belfast	GMT+0:00	I
BHX	Birmingham Intl.	Birmingham	GMT+0:00	I
BOS	Boston I'nal Airport	Boston	GMT-5:00	D
BRS	Bristol Lulsgate	Bristol	GMT+0:00	I
BRU	Brussels National	Brussels	GMT+2:00	I
BUD	Budapest Ferihegi	Budapest	GMT+1:00	I
CDG	Paris Charles De G...	Paris	GMT+1:00	I
CGN	Cologne Bonn	Cologne	GMT-1:00	I
CLE	Cleveland Hopkins	Cleveland	GMT-5:00	D

Figure 2: Example input screen: Airport details.

The back-end module has six primary components. First, as we will describe later in Section 5, we have a preprocessing engine that uses the flight and resource information to create the set of all valid flight schedules which serve as the defender's targets in our system. Along with the flight data, this forms the target definition for the game. Second, we have a payoff generation process that combines the target definition information with the risk data. Third, we translate the payoff information into a Stackelberg game, represented as an MILP. Fourth, we include a generic MILP solver to solve the MILP created via the ERASER-C model. Fifth, we produce a randomized schedule of probability weights for each target based on the solution created by ERASER-C. Finally, we use the randomized schedule to create actual sample schedules that can be implemented by the FAMS.

This information is presented to the user via the display/output module. The schedule created is shown in the interface with pop-up windows as the user's mouse moves over the targets, showing more detailed information about each target. The user is also able to output the schedule to a file which he can then use to analyze the schedule in more detail as desired. The sample assignment of FAMs 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 simply change the assignment of FAMs 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 re-solve the game completely.

The final module, the project manager, is a project-based system where the input files can be stored as a 'state-of-the-world', which can then be loaded again for future use. The user can then specify a default project for general operational use. If additional security risks are known to exist for the upcoming scheduling cycle, the user can alter the default project and save the new settings in case a similar situation arises again in the future. If this occurs, the user simply needs to load the previously saved project and all the data will be loaded into the system automatically. This also ensures proper record-keeping of the entire process of schedule-generation for accountability purposes.

5. MAJOR CHALLENGES

Now we describe the process of mapping the FAMS domain into a Stackelberg security game and the challenges encountered and addressed during the process.

5.1 Describing the game

Although we have decided to cast the FAMS domain as a Stackelberg game, we still have not determined what to use as targets or their payoffs. For this, we must consult with the experts in the domain such as the current scheduler and/or field personnel to understand what risks and potential damages they see in their domain. In addition to understanding the current scheduling practices, the key questions to answer in this phase will be used to determine the structure of the game, such as target definition, resource capabilities and constraints, and risk evaluation for the targets. The answers to this set of questions will allow us to determine the data required as inputs to the IRIS system.

The first two items are much more straightforward to understand. The difficulty that remains with defining the game is efficiently gathering the risk data. While we could simply ask the user to enter each of these risk numbers for every single flight, the attribute-based nature of the problem lends itself to a much easier method. Here we introduce an attribute-based preference elicitation system for aggregating the risk data associated with targets.

We create a preference elicitation system based on the Threat, Vulnerability and Consequence (TVC) model for estimating terrorism risk [21]. In TVC, the risk of terrorism is broken down into more easily estimated components and a formula is created to combine them into an aggregate risk value. Thus, instead of asking the user to calculate and input each flight's overall risk values themselves, we can simply ask the user to input the risk/value numbers for each of the components of risk that he would use to make the calculation. From here we can create a vector of attributes for each flight by automatically pulling in values that pertain to a flight. Using each of these vectors, we can specify a combination formula to aggregate the information into payoff values for use in our Stackelberg game model. Since a given flight's payoff value may be based on a large number of attributes, we also allow the user to bypass this system and directly edit the payoff value if required. This allows the user to make quick corrections or changes if special situations arise for individual flights.

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 4 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.

5.2 Solving the game

We now describe in more detail the considerations involved in addressing the scheduling constraint challenge mentioned in Section 2 that we use ERASER-C to resolve.

Recall that we have a number of hard scheduling constraints inherent in the FAMS domain such as not being able to schedule a FAM on a pair of flights from City A to City B and back if the return flight leaves before the arrival flight lands. These hard scheduling constraints in the FAMS domain make the naïve formulation of modeling every combination of possible schedules difficult to implement. With tens of thousands of flights per day to consider, the number of combinations of possible schedules explodes prohibitively quickly as is discussed in detail in Kiekintveld et al. [10].

One reasonable approach to this issue would be to model the problem with each 'target' representing a possible 'schedule' of flights and simply preprocessing the flights into valid schedules before creating the game model. Thus, instead of a possible coverage being 'Flight 22 from City A to City B', it might be 'Flight 22 from City A to City B and Flight 12 from City B to City A', where each flight schedule is a realistic sequence of flights that a FAM could take that would also bring him home at the end of the schedule. The defender would now simply need to determine an optimal coverage strategy over the universe of flight schedules. The attacker would do the same. However, this is no longer a true representation of the domain. In reality, a potential attacker can actually choose to attack individual flights, not only a sequence of flights, and restricting him to only attacking flight sequences changes the problem we are solving.

An example that illustrates this problem can be constructed as follows. Suppose there are four flights, A,B,C, and D, where flights A and B are medium value, C is extremely high value, and D has extremely low value. After constructing flight schedules, we find that flights must be paired as A/B and C/D. Suppose that in our valuation, the two medium-value flights come to the same overall value assessment as a high-value paired with a low-value. Thus, our strategy would be to defend both pairs with equal probability, since an attacker evaluating the options would see the same two equal-value options and be indifferent between the two.

However, an attacker might actually view flights individually because they may not have scheduling constraints. With defenders spread evenly across all four flights, the attacker is clearly best off attacking the highly-valued flight C. This should alter our strategy so that we place more emphasis on the C/D flight pair than we did previously to lower their expected payoff. Thus, the model of flight schedules for attackers and defenders does not properly represent the game and can lead to suboptimal strategies.

Instead, we model the scheduling constraints explicitly by preprocessing the flights into valid flight schedules and incorporating these in a compact way as described in Section 3.3. This alleviates the problem of an exploding actionspace and allows us to solve much larger games in reasonable timeframes. Thus, we are able to simultaneously overcome the complexity introduced by scheduling constraints and solve these games much more quickly than the DOBSS method could have.

6. ORGANIZATIONAL ACCEPTANCE

In real-world deployment of cutting-edge research, overcoming organizational doubt and resistance is a critical and easily overlooked aspect of the project. For example, if a new methodology has too steep of a learning curve or is too dramatic a change from the current methodology, it is very difficult for people to adopt. If the new program requires significant infrastructure change in order to incorporate it into other routines, it may simply not be cost-effective to do so. Also, if people have trouble convincing themselves that the new solution performs at least as well as the existing solution, we cannot reasonably expect them to use it. We now discuss three key points that contributed to our successful collaboration with the FAMS: adhering to current practices, ease of incorporation, and error-checking.

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 towards a successful implementation. For example, we spent months fine-tuning IRIS's interface until we achieved a look and feel that they were familiar and comfortable with. Changes included such things as what information to display in the schedule, how to output the finalized schedule, how to format the flight number and carrier name, terminology to use, colors to use, logos, etc. While it is true that these changes were all cosmetic and as researchers we may be tempted to dismiss them, their importance cannot be stressed enough in real-world implementation.

Similarly, because infrastructure changes are often costly and time-consuming, ease of incorporating our work into their daily routine is essential. In our case, the FAMS had a system from which they could extract flight information as for their current schedule generation practices and then a set of tools for analyzing the schedules that were created. While the specific format of inputs/ouputs is not of theoretical interest, we recognized that incorporating IRIS into their scheduling practices will probably require a significant amount of work already and asking them to change other processes on our behalf would be very inconsiderate. Allowing IRIS to use inputs and create outputs that were in the same format as existing protocols minimized the additional work that our assistant would create for them.

Another aspect of organizational acceptance is making it easy to error-check your implementation. The existing practices have a proven track record and users will inevitably be wary of new methods until they have convinced themselves of its quality. Instead of attempting to convince the users that the solution is correct, we found that giving them tools to check, compare, and alter the solution's output served just as well. For example, in IRIS, we output schedules in a manner that easily allows for external reporting tools to be run on them. This makes it that much simpler to convince users of the 'randomness' of the schedules and that we are not modeling their problem incorrectly. Also, due to the size of the problem, error-checking helps identify errors in the process that could lead to incorrect outputs. Instead of asking people to unreasonably assume correctness of our solution, we simply include tools for them to check the results directly.

While not strictly a part of most academic research, organizational acceptance is a very real problem that must be tackled with every application of theory into practice. Recognizing and working through these situations is crucial and oftentimes proves to be quite simple to achieve.

7. EXPERIMENTAL RESULTS

Two primary concerns might be raised regarding the IRIS system: (i) time required to develop a randomized schedule; (ii) evidence supporting the quality of the schedules generated. To address this, we conduct experiments exploring these two issues. All of our experiments were run on a machine with dual Xeon 3.2Ghz processors and 2GB of RAM, running RHEL 3. We use CPLEX 9.0.0 with default parameter settings to solve the MILPs.

7.1 Runtime Analysis

In creating the system, practical application was the primary goal. If the system was not fast enough for feasible use, then it could not be useful to the FAMS. Thus, we set an initial goal of creating a week's schedule for all flights between the United states and a Region A within a reasonable timeframe and include tools to manage the effects of scheduling for successive periods to allow for schedule-creation for longer periods. In Figure 3, we show results of creating randomized schedules for subsets of the FAMS problem.

The experiments were run using one week of real flight data for subregions of Region A and three separate sets of hypothetical FAMS home city data that vary the number of FAMs available to explore its impact upon runtime. Region A and the countries within it are actual places for which we have used real flight data, but due to security concerns we have anonymized them here. Region A is composed of five larger countries which we have designated 1-5 as well as a few small countries which are only included in the full region tests. We created random values for the other inputs and held them constant throughout the experiments.

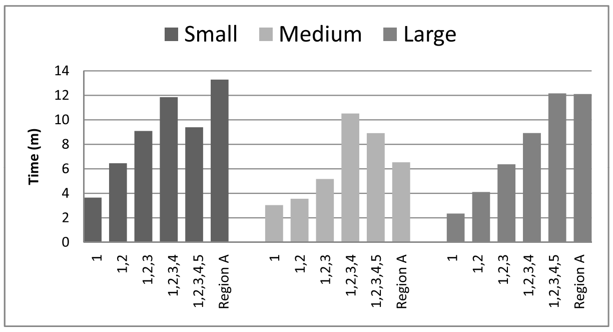


Figure 3: Runtimes for scheduling one week subregions of region A.

Region	Flights	Flight Schedules
1	873	1,181
1,2	1,147	1,403
1,2,3	1,528	1,660
1,2,3,4	1,845	1,975
1,2,3,4,5	2,033	2,114
Region A	2,571	2,416

Table 6: Game sizes for experiments run.

In Figure 3, the y-axis represents the time required for generating a schedule in minutes and the x-axis shows which countries are included in the game being run. The x-axis is grouped in three separate sets which represent, from left to right, a small number of FAMs, a medium number of FAMs, and a large number of FAMs distributed through the same set of home cities. For example, the first bar on the left represents an average runtime of 3.65 minutes to create a schedule for all flights to Country 1 within a one week period over 20 trials. In light grey, the first bar on the left represents the exact same experiment, but run with a medium number of FAMs. In Table 6, we show the exact size of the game for each of the subregions. As can be seen, all tested game instances could be completed within 15 minutes.

7.2 Evaluation

In evaluating our method, we compare the schedules generated using IRIS against a uniform random policy as well as a naïve weighting policy. In the uniform random policy, each location covers all of its reachable flight schedules with equal probability regardless of payoff. In the naïve weighting policy, each location randomizes across the flight schedules based solely on the payoff attainable by the attacker, where probabilities are dictated by the relative payoffs. For example, if the universe of flight schedules included 11 flights, where 10 flights had payoff of 10 and 1 flight had a payoff of 100 and the defender had 2 resources available, the naïve weighting policy would assign a probability of 100 percent to the high value flight and 10 percent each to the low value flights. As the benchmark of quality, we calculate the highest expected payoff attainable by the defender assuming that the attacker chooses the target with the highest payoff for him.

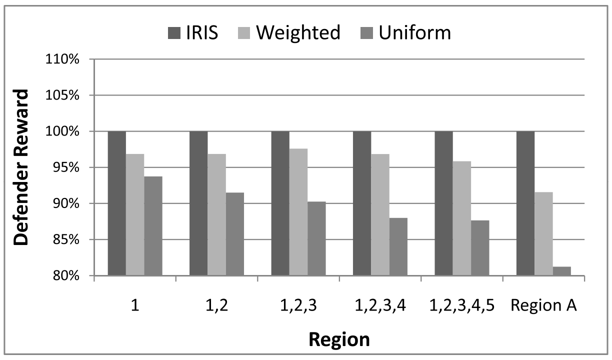


Figure 4: Maximum expected defender reward by strategy.

In Figure 4, the y-axis represents the normalized payoff return for each of the three strategies, with all payoffs normalized to the maximum expected defender's payoff achievable under the strategy generated by IRIS. Across the x-axis, each of the regions are labeled as before. Thus, the rightmost group of bars, from left to right, represents the maximum expected defender's payoff achievable under IRIS as 100 percent, under the weighted randomization strategy as 9 percent worse than IRIS's, and under the uniform randomization strategy as 19 percent worse than IRIS. As can be seen, IRIS's solution is superior to the other two strategies in every region tested.

8. SUMMARY

Onboard patrols are a key component of law enforcement in transportation networks. In generating schedules for these patrols, it is important to account for varying weights of the vehicles being protected as well as the fact that potential attackers can often observe the procedures being used. This paper describes a scheduling assistant for the FAMS, IRIS, which provides a game-theoretic solution to this problem. Although similar in spirit to the ARMOR system deployed at LAX, the inherent challenges of transportation networks evident in the FAMS domain necessitate major additions and advances to the existing methodologies.

In particularly, IRIS combines three key advancements: (i) it uses the fastest known solver for this class of security games, ERASER-C, that exploits symmetries in the payoff structure; (ii) it models the problem with action definitions for defenders and attackers that allow us to efficiently handle the scheduling constraints inherent in the domain; (iii) it includes an attribute-based preference elicitation system for calculating risk values for targets to alleviate the need for users to enter risk values for each target individually.

IRIS makes 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 timeframes. Thus, although our work on IRIS thus far has been restricted to the FAMS, it provides a general framework for solving patrolling scheduling problems in other transportation networks as well.

One example of another transportation network that resembles the FAMS domain would be the New York City subway system. The NYC subway includes tens of subway routes, thousands of subway cars, and a daily weekday ridership of a few million. Although there are only a few subway routes relative to the number of commercial flights in the FAMS domain, once we consider that a patrolling officer can start and end his patrol route at any two subway stops and must also choose which subway cars to protect while aboard the train, the number of potential patrol 'actions' explodes extremely quickly. Similarly to the FAMS domain, we also have scheduling constraints on what sequence of subway routes a law enforcement officer can patrol. Other transportation networks that share similar challenges include bus systems and commuter rails as well.

A working version of IRIS is currently undergoing review by the FAMS as of December 2008 and we are working to expand our methods to other applicable domains in the near future.

9. ACKNOWLEDGEMENTS

IRIS's development has only been so successful due to the exceptional collaboration with the United States Federal Air Marshal Service. 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. However, any opinions, finidngs, and conclusions or recommendations in this document are those of the authors and do not necessarily reflect views of the United States Department of Homeland Security.

10. REFERENCES

[1] Federal Air Marshal Service. *http://en.wikipedia.org/wiki/Federal_Air_Marshal_Service*, 2008.

[2] TSA: Federal Air Marshals. *http://www.tsa.gov/lawenforcement/programs/fams.shtm*,

2008.

[3] R. Avenhaus, B. von Stengel, and S. Zamir. Inspection games. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory*, volume 3, chapter 51, pages 1947–1987. North-Holland, Amsterdam, 2002.

[4] N. Billante. The Beat Goes On: Policing for Crime Prevention. *http://www.cis.org.au/IssueAnalysis/ia38/ia38.htm*, 2003.

[5] M. Blanco, A. Valino, J. Heijs, T. Baumert, and J. G. Gomez. The Economic Cost of March 11: Measuring the direct economic cost of the terrorist attack on March 11, 2004 in Madrid. *Terrorism and Political Violence*, 19(4):489–509, 2007.

[6] G. Brown, M. Carlyle, J. Salmeron, and K. Wood. Defending Critical Infrastructure. *Interfaces*, 36(6):530–544, 2006.

[7] L. Chen and P. Pu. Survey of preference elicitation methods. Technical report, Ecole Politechnique Federale de Lausanne, 2004.

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

[9] D. Kenney. *Police and Policing: Contemporary Issues*. Praeger, 1989.

[10] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, M. Tambe, and F. Ordonez. Computing Optimal Randomized Resource Allocations for Massive Security Games. In *AAMAS-09*, 2009.

[11] R. Looney. Economic Costs to the United States Stemming From the 9/11 Attacks. *Strategic Insights*, 1(6), August 2002.

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

[13] P. Paruchuri, J. P. Pearce, M. Tambe, F. Ordonez, and S. Kraus. An Efficient Heuristic Approach for Security Against Multiple Adversaries. In *AAMAS-07*, 2007.

[14] J. Pita, M. Jain, C. Western, C. Portway, M. Tambe, F. Ordonez, S. Kraus, and P. Parachuri. Depoloyed ARMOR protection: The application of a game-theoretic model for security at the Los Angeles International Airport. In *AAMAS-08 (Industry Track)*, 2008.

[15] S. Ruan, C. Meirina, F. Yu, K. R. Pattipati, and R. L. Popp. Patrolling in a Stochastic Environment. In *10th Intl. Command and Control Research and Tech. Symp.*, 2005.

[16] T. Sandholm, A. Gilpin, and V. Conitzer. Mixed-integer programming methods for finding Nash equilibria. In *AAAI-05*, pages 495–501, 2005.

[17] P. Thornton. London Bombings: Economic cost of attacks estimated at 2bn, July 2005.

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

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

[20] W. A. Wagenaar. Generation of Random Sequences by Human Subjects: A Critical Survey of Literature. 1972.

[21] H. Willis, A. Morral, T. Kelly, and J. Medby. *Estimating Terrorism Risk*. RAND Corporation, 2005.