ARMOR Security for Los Angeles International Airport

James Pita, Manish Jain, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western,
***Praveen Paruchuri, **Sarit Kraus**
University of Southern California, Los Angeles, CA 90089
*Intelligent Automation, Inc., RockVille, MD 20855
**Bar-llan University, Ramat-Gan 52900, Israel
Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742

Abstract

Security at major locations of economic or political importance is a key concern around the world, particularly given the threat of terrorism. Limited security resources prevent full security coverage at all times, which allows adversaries to observe and exploit patterns in selective patrolling or monitoring, e.g. they can plan an attack avoiding existing patrols. Hence, randomized patrolling or monitoring is important, but randomization must provide distinct weights to different actions based on their complex costs and benefits. To this end, this demonstration showcases a promising transition of the latest in multi-agent algorithms into a deployed application. In particular, it exhibits a software assistant agent called ARMOR (Assistant for Randomized Monitoring over Routes) that casts this patrolling/monitoring problem as a Bayesian Stackelberg game, allowing the agent to appropriately weigh the different actions in randomization, as well as uncertainty over adversary types. ARMOR combines two key features: (i) It uses the fastest known solver for Bayesian Stackelberg games called DOBSS, where the dominant mixed strategies enable randomization; (ii) Its mixed-initiative based interface allows users to occasionally adjust or override the automated schedule based on their local constraints. ARMOR has been successfully deployed since August 2007 at the Los Angeles International Airport (LAX) to randomize checkpoints on the roadways entering the airport and canine patrol routes within the airport terminals.

Introduction

Protecting national infrastructure is a challenging task for police and security agencies around the world; a challenge that is exacerbated by the threat of terrorism. Such protection of important locations includes tasks such as monitoring all entrances or inbound roads and checking inbound traffic. 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. Randomizing schedules for patrolling, checking, or monitoring is thus an important tool in the police arsenal to avoid the vulnerability that comes with predictability.

This demonstration focuses on a deployed software assistant agent that can aid police or other security agencies in randomizing their security schedules. We face at least three key challenges in building such a software assistant. First, the assistant must provide quality guarantees in randomization by appropriately weighing the costs and benefits of the different options available. Second, the assistant must address the uncertainty in information that security forces have about the adversary. Third, the assistant must enable a mixed-initiative interaction with potential users rather than dictating a schedule; the assistant may be unaware of users' real-world constraints and hence users must be able to shape the schedule development.

We have addressed these challenges in a software assistant agent called ARMOR (Assistant for Randomized Monitoring over Routes). Based on game-theoretic principles, ARMOR combines three key features to address each of the challenges outlined above. We build on these game theoretic foundations to reason about two agents — the police force and their adversary — in providing a method of randomization. In particular, the main contribution of our software is mapping the problem of security scheduling as a Bayesian Stackelberg game (Conitzer & Sandholm 2006) and solving it via the fastest optimal algorithm for such games called DOBSS (Decomponsed Optimal Bayesian Stackelberg Solver) (Paruchuri *et al.* 2008), addressing the first two challenges. While a Bayesian game allows us to address uncertainty over adversary types, by optimally solving such Bayesian Stackelberg games (which yield optimal randomized strategies as solutions), ARMOR provides quality guarantees on the schedules generated. The third challenge is addressed by ARMOR's use of a mixed-initiative based interface, where users are allowed to graphically enter different constraints to shape the schedule generated. ARMOR is thus a collaborative assistant that iterates over generated schedules rather than a rigid one-shot scheduler.

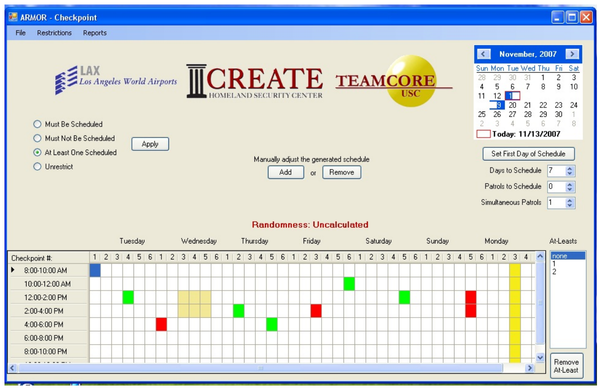
ARMOR has been deployed by Los Angeles Airport (LAX) police for the past six months to randomize scheduling of checkpoints (to check vehicles inbound to LAX) and canine patrols; pictures of these checkpoints and canine patrols are shown in Figure 1(a) and Figure 1(b).

System Architecture

There are two separate versions of ARMOR, ARMOR-checkpoint and ARMOR-canine. While in the following we focus on ARMOR-checkpoint for illustration, both these versions use the same underlying architecture with different inputs. This architecture consists of a front-end and

(a) LAX Checkpoint (b) Canine Patrol

Figure 1: LAX Security



Must Be Scheduled
Must Not Be Scheduled
[✓ At Least One Scheduled]
Unrestrict

Days to Schedule 7
Patrols to Schedule 0
Simultaneous Patrols 1

Figure 2: ARMOR Interface

a back-end, integrating four key components: (i) a front-end interface for user interaction; (ii) a method for creating Bayesian Stackelberg game matrices; (iii) an implementation of DOBSS; (iv) a method for producing suggested schedules for the user. They also contain two major forms of external input. First, they allow for direct user input into the system through the interface. Second, they allow for file input of relevant information for checkpoints or canines, such as traffic/passenger volume by time of day, which can greatly affect the security measures taken and the values of certain actions.

The ARMOR interface, seen in Figure 2, consists of a file menu, options for local constraints, options to alter the action space, a monthly calendar and a main spreadsheet to view any day(s) from the calendar. Together these components create a working interface that meets all the key requirements set forth by LAX officers for checkpoint and canine deployment at LAX.

The base of the interface is designed around six possible adjustable options. The first three options alter the action space and are as follows: (i) number of checkpoints allowed during a particular timeslot; (ii) time interval of each timeslot; (iii) number of days to schedule over. For each given timeslot, the system constructs a new game. Given knowledge of the total number of inbound roads, the number of checkpoints allowed during that timeslot determines the available actions for the LAX police, whereas the action space of the adversary is determined by the number of inbound roads. Thus, the system can set up the foundation for the Bayesian Stackelberg game by providing all the actions possible in the game. Once the action space has been generated, it can be sent to the back-end to be set up as a Bayesian Stackelberg game, solved, and returned as a suggested schedule, which is displayed to the user via the spreadsheet. The third option determines how many iterations of the game will be played (as it determines the number of days to schedule over).

Given the submitted user information, the system must create a meaningful Bayesian Stackelberg game matrix. Based on pre-specified rewards we can provide the rewards for the LAX police and the adversaries to generate a game matrix for each adversary type. After the final game matrices are constructed for each adversary type, they are sent to the DOBSS implementation, which chooses the optimal mixed strategy over the current action space.

To demonstrate the process, assume there are three possible inbound roads or checkpoint locations (A, B, C), one possible timeslot to schedule over, and two checkpoints available for scheduling. Given this scenario, the unique combinations possible include scheduling checkpoints A and B, A and C, and B and C, over the given time frame. We will assume that checkpoints A and B are highly valuable while C, although not completely invaluable, has a very low value. Based on this information, a likely mixed strategy generated by DOBSS would be to assign a high probability to choosing action A and B, say seventy percent, and a low probability to both the other actions, say fifteen percent each. Whatever the mixed strategy actually comes out to be, it is the optimal strategy a user could take to maximize security based on the given information. This mixed strategy is then stored and used for the actual schedule generation.

The next three options serve to impose local constraints in the generated schedule: (i) forced checkpoint; (ii) forbidden checkpoint; (iii) 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 modify the output of the game. The user may impose these restrictions by forcing specific actions in the schedule. In particular, the "forced checkpoint" option schedules a checkpoint at a specific time on a specific day. The "forbidden checkpoint" option designates a specific time on a specific day when a checkpoint should not be scheduled. Finally, the "at least one checkpoint" option designates a set of time slots and ensures that a checkpoint is scheduled in at least one of the slots.

Acknowledgements: This research is supported by the United States Department of Homeland Security through Center for Risk and Economic Analysis of Terrorism Events (CREATE).

References

Conitzer, V., and Sandholm, T. 2006. Computing the Optimal Strategy to Commit to. In *Proceeding of the ACM Conference on Electronic Commerce (ACM-EC)*.

Paruchuri, P.; Pearce, J. P.; Marecki, J.; Tambe, M.; Ordóñez, F.; and Kraus, S. 2008. Playing Games for Security: An Efficient Exact Algorithm for Solving Bayesian Stackelberg Games. In *AAMAS*.