Mixed-Initiative Optimization in Security Games: A Preliminary Report

Bo An, Manish Jain, Milind Tambe
Computer Science Department
University of Southern California
Los Angeles, CA 90089
{boa,manish.jain,tambe}@usc.edu

Christopher Kiekintveld
Department of Computer Science
University of Texas, El Paso
El Paso, TX 79968
cdkiekintveld@utep.edu

Abstract

Stackelberg games have been widely used to model patrolling or monitoring problems in security. In a Stackelberg security game, the defender commits to a strategy and the adversary makes its decision with knowledge of the leader's commitment. Algorithms for computing the defender's optimal strategy are used in deployed decision-support tools in use by the Los Angeles International Airport (LAX), the Federal Air Marshals Service, and the Transportation Security Administration (TSA). Those algorithms take into account various resource usage constraints defined by human users. However, those constraints may lead to poor (even infeasible) solutions due to users' insufficient information and bounded rationality. A mixed-initiative approach, in which human users and software assistants (agents) collaborate to make security decisions, is needed. Efficient human-agent interaction process leads to models with higher overall solution quality. This paper preliminarily analyzes the needs and challenges for such a mixed-initiative approach.

Introduction

The last five years have witnessed the successful application of game theory in reasoning about complex security problems. Stackelberg games have been widely used to model patrolling or monitoring problems in security. In a Stackelberg security game, the defender commits to a strategy and the adversary makes its decision with knowledge of the leader's commitment. Two systems applying Stackelberg game models to assist with randomized resource allocation decisions are currently in use by the Los Angeles International Airport (LAX) (Pita et al. 2008) and the Federal Air Marshals Service (FAMS) (Tsai et al. 2009). The United States Transportation Security Administration (TSA) is currently evaluating the application of a similar system called GUARDS for use in scheduling airport security operations. At the heart of these deployed applications are the algorithms (e.g., DOBSS (Paruchuri et al. 2007), ERASER-C (Kiekintveld et al. 2009), ASPEN (Jain et al. 2010a)) for computing the defender's optimal strategy given that the follower will behave strategically with the knowledge of the defender's strategy. Existence of various constraints makes it challenging to efficiently compute the defender's optimal strategy in Stackelberg security games. For both the deployed applications (Pita et al. 2008; Tsai et al. 2009), the defender has limited resources such as the available vehicle checkpoints, canine units and air marshals. In addition, when human users are faced with exceptional circumstances and/or extra knowledge, they add hard constraints such as forced checkpoints at some locations or requiring a minimum percentage of flights to be covered. Current approaches will simply compute the optimal solution to meet all the constraints (if possible). Unfortunately, these user defined constraints may lead to poor (or infeasible) solutions due to the users' bounded rationality and insufficient information about how constraints affect the solution quality. Significantly better solution quality can be obtained if some of these constraints can be relaxed. However, there may be infinitely many ways of relaxing constraints and the software assistant may not know which constraints can be relaxed and by how much, as well as the real-world consequences of relaxing some constraints. Therefore, it is promising to adopt a mixed-initiative approach in which human users and software assistants collaborate to make security decisions.

However, designing an efficient mixed-initiative optimization approach is not trivial and there are five major challenges. First, the scale of security games and constraints prevent us from using an exhaustive search algorithm to explore all constraint sets. Second, the user's incomplete information regarding the consequences of relaxing constraints requires preference elicitation support. Third, the decision making of shifting control between the user and the software assistant is challenging. Fourth, it is difficult to evaluate the performance of a mixed-initiative approach. Finally, it is a challenging problem to design good user interfaces for the software assistant to explain how constraints affect the solution quality. This paper preliminarily analyzes the needs and challenges for a mixed-initiative approach in security games.

Security Games with Constraints

In a Stackelberg security game there are two agents - the defender (security force) and an attacker — who act as the leader and the follower respectively. There are a set of targets, which the defender is trying to protect. Each target has a reward and penalty for both the defender and the attacker. Thus, some targets may be more valuable to the defender than others. To protect these targets the defender can allocate its limited resources to execute a security activity. Once a resource is allocated to a target, the target is marked as covered, otherwise it is marked as uncovered. In a Stackelberg security game, the defender first commits to a strategy, then the attacker decides its optimal attack with knowledge of the defender's strategy. Therefore, the defender's goal is to maximize its reward given that the attacker will attack with knowledge of the defensive strategy the defender has chosen. In most cases, the optimal strategy for the defender is a randomized strategy in which it chooses a mixed strategy over all its possible resource assignments. Randomized policies are unpredictable since even though the attacker may know the overall strategy, the attacker is unable to predict the exact resource assignment for any day.

Stackelberg security games have been applied in several contexts to assist security forces. The ARMOR system (Assistant for Randomized Monitoring over Routes) deployed at the Los Angeles International Airport (LAX) is used by airport police to randomize the placement of checkpoints on roads entering the airport, and the routes of canine unit patrols in the airport terminals (Pita et al. 2008). The IRIS system (Intelligent Randomization in Scheduling) is designed to randomize flight schedules for the Federal Air Marshals Service (FAMS) (Tsai et al. 2009). The security game models at the heart of ARMOR and IRIS are created based on the data provided by human users including LAX police and federal air marshals.

Human users also specify a variety of constraints about resource usage. There are two types of constraints: non-negotiable constraints and negotiable constraints. Non-negotiable constraints must be obeyed at all times and cannot be modified. One example of non-negotiable constraints is a scheduling constraint. For instance, air marshals in the IRIS application must be scheduled on tours of flights that obey temporal constraints (e.g., the time required to board, fly, and disembark). The Federal Air Marshals Service (FAMS) also requires more complex reasoning about logistical constraints in how resources are scheduled. For example, a given marshal cannot be assigned to two flights with overlapping time schedules and an air marshal that is current in city A can only leave on flights that are outbound from this area. In addition, the schedules that the air marshals can actually fly are subject to other policy constraints.

Negotiable constraints can be modified at run-time by human users and we only concern about negotiable constraints in this paper. There are different types of negotiable constraints. The defender always has resource constraints. In the ARMOR system, the numbers of available vehicle checkpoints and canine units are limited. In the FAMS domain, there are limited number of marshals. In addition, human users may place constraints on the defender's actions to affect the output of the game when they are faced with exceptional circumstances and extra knowledge. For instance, in the ARMOR system there could be forced checkpoints (e.g., when the Governor is flying) and forbidden checkpoints. In the FAMS domain, there could be global constraints such as 30% of flights from city A should be covered or all flights from city A to another city B should be covered.

It is a challenging problem to efficiently compute the defender's optimal strategy to meet all of these constraints and the literature provides a number of solutions such as DOBSS (Paruchuri et al. 2007), ERASER-C (Kiekintveld et al. 2009), and ASPEN (Jain et al. 2010a). However, those (negotiable) constraints defined by humans may lead to poor (even infeasible) solutions due to users' bounded rationality and insufficient information about how constraints affect the solution quality. For instance, if there are a small number of marshals in city A, an optimization algorithm cannot find a solution to satisfy the constraint that 30% of flights from city A should be covered. We may get significantly better solution quality if we slightly relax some constraints. For instance, the user in the ARMOR domain may alter the schedule by altering the forbidden/required checkpoints. Therefore, it is promising to adopt a mixed-initiative approach in which human users and software assistants collaborate to make better security decisions.

Mixed-Initiative Optimization in Security Games

Mixed-initiative interaction refers to flexible interaction strategy in which humans and software agents/assistants collaborate to achieve their objectives. The basic issues involved in a mixed-initiative approach include how to divide the responsibility between the human and the software agent, how to adjust control between the human and the software agent, and how to exchange knowledge and information between the human and the software agent (Horvitz 1999; Tecuci, Boicu, and Cox 2007). Many of the challenges associated with mixed-initiative interaction have been studied in recent years within the research area of adjustable autonomy (Scerri, Pynadath, and Tambe 2002; Schurr et al. 2006; Schurr, Marecki, and Tambe 2009).

The problem of mixed-initiative optimization in security games is the collaboration between human users and software assistants to make trade-offs between maximizing the solution quality and minimizing the cost associated with relaxing constraints. Consider the following simple scenario of the FAMS domain in which we need to decide patrol schedules (i.e., assign FAMS to flights) for the flights from city A to cities B, C, and D. There are three initial negotiable constraints regarding resource usage: 1) There are 20 marshals in city A; 2) 30% of flights from city A to cities B, C, and D should be covered; and 3) all the flights from city A to city B should be covered. With this initial constraints, there may be no feasible schedule if there are too many flights from city A. Then the software assistant and the human user interact with each other to relax some constraints to improve the quality of the schedule. There are many ways to relax constraints, e.g., having more resources at city A, decreasing the coverage percentage, or allowing some flights to city B not being covered. While the solution quality of the schedule may increase if some constraints are relaxed, the human user also suffers from a cost by relaxing constraints. For instance, it is costly to have more air marshals.

Let the set of constraints provided by the human user be $\mathcal{C}$.

The problem of computing the defender's optimal strategy in a security game can be formulated as follows.

$$\begin{aligned} \max_{x} \quad & f(x) \\ \text{s.t.} \quad & \mathcal{C}, \Re \end{aligned}$$

where x is defender's policy, which consists of a vector of the defender's pure strategies, $f(x)$ is the defender's expected utility given its strategy x and the attacker's best response, and $\Re$ is the set of constraints enforcing feasibility of both agents' strategy and optimality of the attacker's strategy (which could be for multiple attacker types) given the defender's strategy x.

Let $x^*(\mathcal{C})$ be the defender's optimal strategy given the set of constraints $\mathcal{C}$. Note that $x^*(\mathcal{C})$ could be infeasible in case that no strategy x can satisfy all the constraints $\mathcal{C}$. We can define the defender's maximum utility $f(\mathcal{C})$ with constraints $\mathcal{C}$ as follows.

$$f(\mathcal{C}) = \begin{cases} f\big(x^*(\mathcal{C})\big) & \text{if } x^*(\mathcal{C}) \text{ is feasible} \\ -\infty & \text{otherwise} \end{cases}$$

The human user may suffer from a cost while relaxing a constraint (e.g., having more air marshals). Thus, the human user has preference over different sets of constraints depending on factors such as the cost of relaxing each constraint and the relative importance of different constraints. For the convenience of analysis, we represent preferences with a cost function and let $c(\mathcal{C})$ be the human user's cost of having a constraint set $\mathcal{C}$. Then the optimization objective of both the software assistant and the human user is as follows.

$$\max_{\mathcal{C}} \quad f(\mathcal{C}) - c(\mathcal{C})$$

We can treat $f(\mathcal{C}) - c(\mathcal{C})$ as the *overall* utility of having constraints $\mathcal{C}$. Given the mixed-initiative optimization objective, the software assistant and the human user will collaborate to find the optimal constraint set that maximizes the overall utility. In addition, we assume that the mixed-initiative interaction has to terminate within a deadline T. There are different interaction models depending on how to decide the set of constraints to explore:

- The software assistant determines different constraint sets to explore and reports to the human user the optimal utility for each constraint set. The human user will make the decision about the final constraint set considering the human user's cost of relaxing constraints. This simple process could involve asking the human user for clarification or additional information.
- The human user always decides which constraint set to explore and the software assistant simply reports the solution quality of its optimal strategy after solving the security game.
- The software assistant and the human user together decide the next constraint set to explore. This process is more complex which involves both the human user and the software assistant's reasoning about the most promising direction of relaxing constraints.

The Challenges

Past work in mixed-initiative planning and scheduling, and the related area of adjustable autonomy have pointed out some of the exciting research challenges. Among these challenges a key challenge is that of addressing the uncertainty in decision making faced by the planning/scheduling tool or agent, and the cost/rewards of making correct/incorrect decisions and the cost of bothering or interrupting a user. While a decision-theoretic framework suggests itself as a solution given such costs and uncertainties, decision-making deadlines adds another layer of complexity. To solve this problem, researchers have focused on frameworks such as MDPs, continuous-time MDPs, dec-MDPs and others, to provide an agent a policy to follow when interacting with human users.

Our work faces many of these challenges, although we may not face complex task structure, complex communication interface, and multiple users/agents faced by others (Myers et al. 2007; Ferguson and Allen 2007; Scerri, Pynadath, and Tambe 2002; Rich and Sidner 2007). However, a key differentiating factor is that underlying our research is a game-theoretic solver. The major challenges of designing a mixed-initiative approach for security games include:

- **Scale of security games and constraints**: There may be a large number of constraints in the domain of security games and for some constraints, there may be infinitely many ways to relax a constraint such as the constraint about flight coverage in the domain of FAMS. Therefore, it is impractical to try all the constraint sets. In addition, even the fast algorithm for solving security games may need more than one hour due to scale of security games. For instance, US commercial airlines fly 27,000 domestic flights and over 2000 international flight each day. The scale of the security games together with real-time requirements preclude us from using an exhaustive search algorithm to try different constraint sets since we need to solve the security game for each constraint set. It is more reasonable for the software assistant to determine the next constraint set based on inferring its interaction history with the human user.
- **Knowledge acquisition**: It is possible that the human user does not have perfect knowledge of the cost of constraints. If the software assistant reports the optimal utilities for different constraint sets, the human user needs to reason about the consequence of adopting different constraint sets and the benefit of having a higher utility at the cost of relaxing some constraints. It may be difficult for the human user to tell whether one constraint is more important (and in what degree) than another constraint. It is even more difficult for the human user to directly measure the exact difference between having a 20% flight coverage and setting the flight coverage to a 15%. Therefore, preference elicitation support is necessary for the mixed-initiative approach.
- **Shift of control**: Recall that there are three interaction models depending on who is deciding the next constraint set to explore. While the human user may have superior

decision-making expertise, it often has bounded rationality and the software assistant may provide good suggestions based on its inference/learning capability. Deciding who decides the constraint set depends on many factors such us the human user's uncertainty, the software assistant's knowledge, and the potential cost of switching control.

- **Evaluation**: It is also difficult to evaluate the performance of a mixed-initiative approach. In addition to common difficulties in evaluating security systems such as security concerns in making evaluations of security policies publicly available and ethical concerns in not providing the best security possible to a control group (Jain et al. 2010b), mixed-initiative interaction brings some new difficulties. For instance, the performance of a mixed-initiative interaction model depends on the type of the human user and it is very costly and time consuming to evaluate different user types. We also need to compare the overall system's performance of the mixed-initiative approach versus fully automated optimization and alternative mixed-initiative approaches.
- **Explanation**: Traditionally, the software assistant never has to explain to the user how the problem formulation maps to the domain of the user and how it is solved. In a mixed-initiative model, a software assistant may need to explain the reason why there is no feasible solution to assist the human user to decide how to relax constraints. It is a challenging problem to design good user interfaces to facilitate the human-agent interaction and to explain how constraints affect the solution quality.

Conclusion

This position paper argues for the development of a mixed-initiative approach for solving security games, in which human users and software assistants collaborate to make security decisions. Research on methods for supporting mixed-initiative interaction in security games will undoubtedly lead to new applications of mixed-initiative interaction.

Future research will focus on different components of a mixed-initiative approach including sensitivity analysis for understanding how different constraints affect the solution quality, inference/learning for discovering directions of relaxing constraints, search for finding constraint sets to explore, preference elicitation for finding the human user's preference of different constraint sets, and interface design for explaining the game theoretic solver's performance. Our approach will be tested on real data from deployed applications.

References

Ferguson, G., and Allen, J. 2007. Mixed-initiative systems for collaborative problem solving. *AI Magazine* 28(2):23–32.

Horvitz, E. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 159–166.

Jain, M.; Kardes, E.; Kiekintveld, C.; Ordonez, F.; and Tambe, M. 2010a. Security games with arbitrary schedules: A branch and price approach. In *AAAI*.

Jain, M.; Tsai, J.; Pita, J.; Kiekintveld, C.; Rathi, S.; Tambe, M.; and Ordonez, F. 2010b. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshals Service. *Interfaces* 40:267–290.

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.

Myers, K.; Berry, P.; Blythe, J.; Conley, K.; Gervasio, M.; McGuinness, D. L.; Morley, D.; Pfeffer, A.; Pollack, M.; and Tambe, M. 2007. An intelligent personal assistant for task and time management. *AI Magazine* 28(2):47–61.

Paruchuri, P.; Pearce, J. P.; Tambe, M.; Ordonez, F.; and Kraus, S. 2007. An efficient heuristic approach for security against multiple adversaries. In *AAMAS*.

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

Rich, C., and Sidner, C. L. 2007. Diamondhelp: A generic collaborative task guidance system. *AI Magazine* 28(2):33–46.

Scerri, P.; Pynadath, D. V.; and Tambe, M. 2002. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research* 17(1):171–228.

Schurr, N.; Patil, P.; Pighin, F.; and Tambe, M. 2006. Using multiagent teams to improve the training of incident commanders. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 1490–1497.

Schurr, N.; Marecki, J.; and Tambe, M. 2009. Improving adjustable autonomy strategies for time-critical domains. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, 353–360.

Tecuci, G.; Boicu, M.; and Cox, M. T. 2007. Seven aspects of mixed-initiative reasoning: An introduction to this special issue on mixed-initiative assistants. *AI Magazine* 28(2):11–18.

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