ISE 599: Security and Game Theory **Instructor: Milind Tambe**

Security Games and Combinatorial Optimization

By *Haifeng Xu* *Spring 2016*

In this note, we first describe the model of *zero-sum* security games, and show how the model and its computation relate to many standard combinatorial optimization problems. Then we show how the model and computation generalize to general-sum security games.

1 Zero-sum Security Games - A Unified View

The zero-sum security game is a two-player game played between a *defender* and an *attacker*. More precisely, a zero-sum security game consists of the follows:

- Two players: a *defender* and an *attacker*.
- n targets (e.g., physical facilities or critical locations) to be protected, denoted by set $[n]$.
- Set of *defender pure strategies*, denoted as $\mathcal{E}$. We will use $\mathbf{e}(\in \mathcal{E})$ to denote a generic pure strategy. Crucially, any $\mathbf{e}$ can be viewed as a *subset* of $[n]$, denoting the protected (a.k.a., *covered*) targets by this pure strategy. In particular, the defender has multiple security resources; One pure strategy is a feasible allocation of these resources – $\mathbf{e}$ is precisely the set of covered targets in this feasible resource allocation.

  Equivalently, one can view $\mathbf{e} \in \{0,1\}^n$ as a *binary* vector where the value-1 entries specify the covered targets. For convenience, we will use this representation throughout this note. A defender mixed strategy is a distribution $\mathbf{p} \in \Delta_{|\mathcal{E}|}$ over elements in $\mathcal{E}$. Here $\Delta_m = \{\mathbf{p} \in \mathbb{R}^m : \sum_{i=1}^m p_i = 1;\ p_i \geq 0\}$ is the m-dimensional *simplex*, consisting of all the (discrete) distributions over support of size m.
- *Attacker pure strategies* $i \in [n]$, meaning that the attacker attacks target i.[1] Like most of the literature, here we assume the attacker can only attack one target. We use $\mathbf{y} \in \Delta_n$ to denote an attacker mixed strategy where y_i is the probability of attacking target i.
- Payoffs: if the attacker attacks any target $i \in [n]$, the defender gets a reward r_i if i is covered or a cost c_i otherwise. As standard, we assume $r_i > c_i$, i.e., protecting a target is beneficial for the defender.

Note that, the defender aims at *maximizing* her utility while the attacker aims at *minimizing* the defender's utility. The key structure of a security game is the set $\mathcal{E}$ of defender pure strategies. Consider a simple example where the defender has $k(< n)$ security resources, each of which can be assigned to protect any target. In this case, any binary vector $\mathbf{e} \in \{0,1\}^n$ with at most k value-1 entries, denoting a subset of $[n]$ with size at most k, is a pure strategy (so $|\mathcal{E}| = \Omega(n^k)$ in this case). In practice, there are usually resource allocation constraints, thus not all such subsets correspond to *feasible* resource allocations. The key difference among different security game models lies at the structure of the set $\mathcal{E}$. More (realistic) examples will be provided in Section 2. We note that, the total number of pure strategies is usually extremely large, therefore computational efficiency in security games means time polynomial in n, while *not* in $|\mathcal{E}|$.

[1]In practice, the attacker can also choose to not attack. This can be incorporated into the current model by adding a dummy target. Therefore, we will not explicitly consider the case.

Marginal Probabilities. Given any defender mixed strategy $\mathbf{p} \in \Delta_{|\mathcal{E}|}$, the *marginal* coverage probability of target i is

$$x_i = \sum_{\mathbf{e}\in\mathcal{E}} p_e e_i \in [0,1]. \tag{1}$$

where e_i is the i'th entry of $\mathbf{e}$. Let $\mathbf{x} = (x_1, ..., x_n)^T$ denote the marginal probabilities of all targets induced by the mixed strategy $\mathbf{p}$. Notice that the marginal probabilities of a pure strategy $\mathbf{e}$ is precisely $\mathbf{e}$ itself,[2] and the convex hull of $\mathcal{E}$, defined as $\mathcal{P} = \{\mathbf{x} : \mathbf{x} = \sum_{\mathbf{e}\in\mathcal{E}} p_e \cdot \mathbf{e}, \forall \mathbf{p} \in \Delta_{|\mathcal{E}|}\}$, consists of all the feasible (i.e., implementable by a defender mixed strategy) marginals.

Player Utilities. Given a defender mixed strategy $\mathbf{p} \in \Delta_{|\mathcal{E}|}$, let $\mathbf{x} \in \mathbb{R}^n$ be the marginal probabilities induced by $\mathbf{p}$, as in Equation (1). Let $\mathbf{y} \in \Delta_n$ be any attacker mixed strategy. Then the defender's expected utility can be expressed in terms of $\mathbf{x}$, $\mathbf{y}$, as follows:

$$U(\mathbf{x},\mathbf{y}) \quad = \quad \textstyle\sum_{i=1}^n y_i \Big(r_i \cdot x_i + c_i \cdot [1 - x_i] \Big), \tag{2}$$

where $r_i \cdot x_i + c_i \cdot [1 - x_i]$ is precisely the defender's expected utility conditioned on that the attacker attacks target i. Note that, $U(\mathbf{x},\mathbf{y})$ has the form $\mathbf{x}^T A \mathbf{y} + \alpha \cdot \mathbf{y}$ for some *non-negative* diagonal matrix A, and is called *bilinear* in $\mathbf{x}$ and $\mathbf{y}$.

2 Security Games & Combinatorial Optimization

Interestingly, it turns out that security games have a very close connection to many combinatorial optimization problems. In particular, we consider the following combinatorial problem.

Problem 2.1 (Defender Best Response (**DBR**)). *For any* non-negative *weight vector* $\mathbf{w} \in \mathbb{R}^n_+$, *compute*

$$\mathbf{e}^* = \arg\max_{\mathbf{e}\in\mathcal{E}} [\mathbf{w} \cdot \mathbf{e}].$$

The DBR problem over $\mathcal{E}$ *is to compute* $\arg\max_{\mathbf{e}\in\mathcal{E}} [\mathbf{w} \cdot \mathbf{e}]$ *for any input* $\mathbf{w} \in \mathbb{R}^n_+$.

In other words, the DBR problem is to compute a defender pure strategy that maximizes the total weights it "collects". We claim that Problem 2.1 is precisely the defender's best response problem to an arbitrary attacker mixed strategy. To see this, given any attacker mixed strategy $\mathbf{y}$, we have $U(\mathbf{x},\mathbf{y}) = \sum_{i=1}^n x_i \Big(y_i [r_i - c_i] \Big) + \sum_{i=1}^n y_i c_i$ for any $\mathbf{x} \in \mathcal{P}$. Let $w_i = y_i [r_i - c_i] \geq 0$. Since the term $\sum_{i=1}^n y_i c_i$ is not affected by the defender strategy, the defender's best response to $\mathbf{y}$ is $\arg\max_{\mathbf{x}\in\mathcal{P}} \mathbf{x} \cdot \mathbf{w} = \arg\max_{\mathbf{e}\in\mathcal{E}} \mathbf{e} \cdot \mathbf{w}$. Oppositely, given any $\mathbf{w} \in \mathbb{R}^n_+$, it is easy to find an attacker mixed strategy $\mathbf{y} \in \Delta_n$ such that $y_i(r_i - c_i)$ is proportional to w_i for all $i \in [n]$, making Problem 2.1 equivalent to the defender's best response to $\mathbf{y}$. Notice that the DBR problem is a combinatorial optimization problem over the set system $\mathcal{E}$. The difference among various security game models essentially lies at the structure of $\mathcal{E}$. In the following context, we illustrate how some typical DBR problems relate to standard combinatorial problems.

Uniform Matroid. In simple security settings, the defender has a certain number of security resources, say k resources; each resource can be assigned to protect any (one) target, i.e., there are no allocation constraints. As a result, any subset of $[n]$ of size at most k is a defender pure strategy. In this case, $\mathcal{E}$ is a uniform matroid and the DBR problem is simply to find the largest k weights. The LAX airport security game system deployed in 2007 – one of the earliest applications of security games – is captured by this model [12].

[2]Here we assume security forces have perfect protection effectiveness. That is, once a target is covered, regardless by one or multiple resources, it is fully protected with probability 1. One can also generalize this to nonperfect effectiveness.

Bipartite Matching. A natural generalization of the uniform matroid case is that the resource allocation has constraints. In this case, the defender has k heterogeneous resources, and each resource can only be allocated to some particular targets associated with that resource. This naturally models several types of scheduling constraints in practice. For example, due to geographic constraints, policemen from a certain police station can only patrol the area around that station. Also, different types of security forces specialize in protecting different types of targets. The feasibility constraints can be modeled as edges of a bipartite graph with security resources on one side and targets on another side; A resource can be assigned to a target if and only if there exists an edge between them in the bipartite graph. Any defender pure strategy corresponds to a bipartite matching and the DBR problem is to compute the maximum weighted bipartite matching.

Coverage Problem. In some domains, one security resource can cover several targets. One deployed real example is the scheduling problem of the federal air marshals, where one air marshal is scheduled to protect several flights, but constrained on that the arrival destination of any former flight should be the starting point of the next flight [15]. In other words, each security resource (i.e., air marshal) can protect a restricted subset of targets (i.e., flights). As a result, each pure strategy is the union of targets covered by each security resource. The DBR problem in this case is the maximum weighted coverage problem. Another natural example is to protect targets distributed on the plane and each security guard can cover a region of certain size. The DBR problem here is a 2-dimensional geometric maximum coverage problem.[3] Other examples include patrolling on a graph (e.g., street graph of a city or network systems) in which a patroller at a node can protect all the adjacent edges or a patroller on an edge can protect its two end nodes. The DBR problem here is the vertex or edge coverage problem.

Min-Cost Flow. Many security games are played out in both space and time, which is also referred as *spatio-temporal* security games. For example, the deployed security system in [6] helps to schedule the patrol boats of US Coast Guard to protect the (moving) Staten Island ferries during the day time. Conservation area patrolling for protecting wildlife is another example [7]. One common way to handle such settings is to discretize the space to build a 2-D – spatial and temporal dimension – grid, and patrol the discrete *(space, time)* points (see Figure 1). The constraint here is that starting from a position at time t, the positions that a security resource can possibly reach at time $t + 1$ are restricted due to various constraints like speed limit, terrain barriers, etc. For example, the move highlighted by red in Figure 1 is infeasible since the patroller can not move to the end within a small time period due to speed limit, while the blue-colored moves are feasible. This can be modeled by adding edges between time layers to indicate feasible moves. The patrolling schedule for each security resource corresponds to a path across all time layers, which specifies the target this resource covers at each time point (see the blue path in Figure 1). The DBR problem is, for any given non-negative weights at each *(space, time)* point, computing k paths for the k resources to maximize the total weights they cover. This can be solved by adding a super *source* and *sink* to the graph, and then computing a k-unit min-cost integer flow with negative costs.

Packing. Our last example is motivated by recent work to optimize the allocation of security resources for passenger screening for the Transportation Security Administration (TSA) in United States [3]. Consider an airport with n flights and flight i has m_i passengers. The TSA has several screening *tools*, e.g., x-ray, walk-through metal detector, chemicals, etc., and each screening tool has a capacity of the maximum number of passengers it can check. Opposite from the coverage case where each resource can protect several targets, here several tools are needed to screen one passenger. More precisely, each passenger is screened by a screening *team* which is a combination of several screening tools. By attacking flight i we mean the attacker becomes a passenger for that flight (i.e., bought its ticket), and brings attack equipments with him. By protecting the flight from a certain passenger we mean that passenger is screened, and identified as an attacker if he is, by a screening team.[4] The DBR problem is to, given non-negative weight w_i for all passengers in any flight i, allocate as many passengers as possible to teams for screening, subject to each screening tool's capacity constraint, so that the total weights of screened passengers are maximized. This is a very general packing problem. In fact, the reader

[3]For more information about geometric coverage, see the thesis [8] by Leeuwen and the references therein.

[4]In [3], each screening team has an effectiveness factor denoting the probability a team can identify an attacker. The setting here is slightly simplified with perfect effectiveness factor 1. Nevertheless, it still captures the core difficulty of the problem.

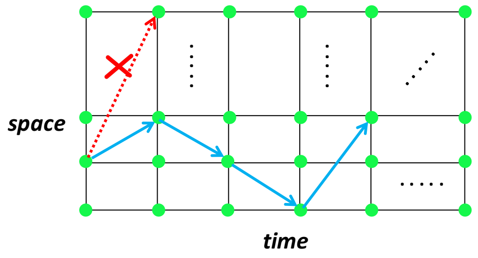

Figure 1: Feasible (blue) and Infeasible (red) Moves in Discretized Spatio-Temporal Games.

may easily check that when $w_i = 1$ and $m_i = 1$ for any flight i, the problem encodes a vertex packing (i.e., independent set) problem.

Remark 2.2. *We note that the examples above do not represent all the settings of security games. For example, there is also study on budget constraints for acquiring security resources (e.g., [1]), which induces the budgeted version of the above combinatorial problems (e.g., the uniform matroid case becomes a* Knapsack *problem). In fact, real domains are usually more complicated with various types of constraints, involving intersections of these combinatorial structures. Nevertheless, the combinatorial nature of these problems does not change and the DBR problem still has the same form as defined in Problem 2.1.*

3 Solving Zero-Sum Security Games is a Combinatorial Problem

We are interested in solving zero-sum security games *over* $\mathcal{E}$, by which we mean all zero-sum games with valid payoff structures, but a fixed set $\mathcal{E}$ of defender pure strategies. It is well-known that in zero-sum games, all the standard equilibrium concepts are payoff-equivalent to the well-known *minimax equilibrium*, and our goal is to efficiently compute the minimax equilibrium for security games over $\mathcal{E}$ in $poly(n)$ time. The following theorem shows that, computing the minimax equilibrium in zero-sum security games is computationally equivalent (in polynomial-time sense) to the DBR problem.

Theorem 3.1. *[16] There is a $poly(n)$ time algorithm to compute the minimax equilibrium for zero-sum security games over $\mathcal{E}$,* if and only if *there is a $poly(n)$ time algorithm to solve the DBR problem over $\mathcal{E}$.*

There has been a belief in the literature on the possibility of solving security games without going through the DBR problem. Indeed, various other techniques have been employed to tackle security games, e.g., generalized Birkhoff-von Neumann theorem [4], various techniques from convex and non-convex optimization as well as heuristics. Interestingly, the key message conveyed by Theorem 3.1 is that to solve security games, one *cannot avoid* solving, and also *only needs* to solve, the DBR problem, if polynomial time efficiency is concerned.

By convention, we will call an algorithm for solving the DBR problem a *DBR Oracle*.[5] The proof of Theorem 3.1 is theoretical and relatively involved. We refer the reader to [16] for a formal proof. However, we do note that, the theoretical reductions in [16] can be practically "implemented" using the techniques of column generation. Though the running time of the implementation is *not* guaranteed to be polynomial (which is why we quoted "implement"), it usually runs efficiently in practice and is widely used within the operation research

[5]This is also called *defender oracle* in some context.

community. In order to gain some intuitions about Theorem 3.1, in the following context, we will briefly illustrate one direction of the theorem, i.e., how to compute the minimax equilibrium using a DBR oracle.

The Minimax Equilibrium Formulation

It is well-known that the minimax equilibrium consists of the defender's maximin strategy and the attacker's minimax strategy. We start from computing the defender's maximin strategy. Let variable $\mathbf{p} \in \Delta_{|\mathcal{E}|}$ be the defender's mixed strategy and variable $\mathbf{x} \in [0,1]^n$ be the marginal coverage probability determined by $\mathbf{p}$ as defined in Equation (1). Assuming an adversarial attacker, the defender maximizes her worst-case utility, denoted by variable u. Then $\mathbf{p}$, $\mathbf{x}$ and u can be computed by the following linear program.

$$\begin{array}{lll} \text{maximize} & u & \\ \text{subject to} & x_i r_i + (1-x_i)c_i \geq u, & \text{for } i = 1,2,...,n. \\ & \sum_{\mathbf{e}\in\mathcal{E}} p_e \cdot \mathbf{e} = \mathbf{x} & \\ & \sum_{\mathbf{e}\in\mathcal{E}} p_e = 1 & \\ & p_e \geq 0, & \text{for } \mathbf{e} \in \mathcal{E}. \end{array} \tag{3}$$

Reducing Minimax Equilibrium to DBR

We show that LP (3) can be computed in $poly(n)$ time if the DBR problem over $\mathcal{E}$ admits a $poly(n)$ time algorithm. LP (3) has exponentially many variables but polynomially many constraints. We instead consider the following *dual program* of LP (3) with variables r and w_i, y_i for all $i \in [n]$.

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^n c_i y_i + r & \\ \text{subject to} & r - \mathbf{e}\cdot\mathbf{w} \geq 0, & \text{for } \mathbf{e} \in \mathcal{E}. \\ & (r_i - c_i)y_i = w_i, & \text{for } i = 1,2,...,n. \\ & \sum_{i=1}^n y_i = 1 & \\ & y_i \geq 0, & \text{for } i = 1,2,...,n. \end{array} \tag{4}$$

We note that the optimal $\{y_i\}_{i\in[n]}$ of LP (4) is precisely the attacker's minimax strategy. Let polytope

$$\mathcal{Q} = \{(r,\mathbf{w},\mathbf{y}) : \, r \in \mathbb{R}, \, \mathbf{w},\mathbf{y} \in \mathbb{R}^n \text{ are feasible for LP (4) } \} \tag{5}$$

be all the feasible solutions for LP (4). The key observation here is that, given any $(r,\mathbf{w},\mathbf{y})$, we can check whether $(r,\mathbf{w},\mathbf{y})$ is in $\mathcal{Q}$ or not, and if not, we can find a hyperplane to separate the given $(r,\mathbf{w},\mathbf{y})$ from $\mathcal{Q}$.[6] To see this, given any r and w_i, y_i for $i \in [n]$, we can explicitly check the last three constraints in LP (4) in $poly(n)$ time. To check whether they satisfy the first constraint, we run the DBR oracle which outputs a pure strategy $\mathbf{e}^*$. If $r < \mathbf{e}^* \cdot \mathbf{w}$, then the first constraint is violated at $\mathbf{e} = \mathbf{e}^*$, which also gives a hyperplane $r - \mathbf{e}^* \cdot \mathbf{w} = 0$ that separates $(r,\mathbf{w},\mathbf{y})$ from $\mathcal{Q}$. Otherwise, $r \geq \mathbf{e}^* \cdot \mathbf{w} = \max_{\mathbf{e}\in\mathcal{E}} \geq \mathbf{e}\cdot\mathbf{w}$ for any $\mathbf{e} \in \mathcal{E}$. Therefore, the first constraint is satisfied. It turns out that such ability of separation will *not only* allow us to theoretically solve both LP (4) and (3) in polynomial time, *but also* gives a practical implementation using column generation to solve both LPs.

4 General-sum Security Games – Stackelberg and Nash

We now briefly mention how the results in Section 3 generalize to general-sum security games. A general-sum security game basically has the same structure as the zero-sum version, except that the attacker now has his own

[6]For any polytope $\mathcal{P} \subseteq \mathbb{R}^n$, a hyperplane $\mathbf{a}\cdot\mathbf{x} = b$ separates $\mathbf{x}_0 \in \mathbb{R}^n$ from $\mathcal{P}$, if $\mathbf{a}\cdot\mathbf{x}_0 < b$ and $\mathbf{a}\cdot\mathbf{x} \geq b$ for any $\mathbf{x} \in \mathcal{P}$. Notice that such separation naturally implies $\mathbf{x}_0 \notin \mathcal{P}$.

utilities on each target depending on its protection status, and the attacker seeks to maximize his own expected utility. Particularly, characterization of the defender's pure strategy set $\mathcal{E}$ and the DBR problem keep the same. The main solution concept adopted in literature is the *Strong Stackelberg Equilibrium* (**SSE**) – the defender is the leader. This is motivated by the consideration that the attacker usually does surveillance before committing an attack, thus is able to observe the empirical distribution of the defender's patrolling strategy. In this case, our goal is to compute the optimal mixed strategy for the defender to commit to.

However, in many cases the attacker does little surveillance. In fact, sometimes even the attacker intends to do surveillance, he cannot observe the defender's strategies due to limited attacker resources and, sometimes, confidentiality of the defender's resource allocation (e.g., plainclothes police). In these settings, the defender cannot commit to a strategy,[7] thus *Nash Equilibrium* (**NE**) serves as a more appropriate solution concept. Simultaneous-move security game models are particularly common for modeling interactions with terrorism, partially due to the fact that the defender's actions are usually confidential in such settings (see, e.g., [9, 13, 14, 2]). In networked information systems, the interaction between the defender (system protector) and attacker (malware) is usually modeled as a simultaneous-move security game as well since malwares do not analyze system's history behaviors, and the goal is to compute some particular (e.g., best or worst) NE [11, 10].

It turns out that similar results as the zero-sum cases hold, as stated in the following two theorems. Furthermore, computing these equilibria can be implemented by column generation techniques as well. For Nash equilibrium, we can not only compute one NE (NEs are not unique in games), but also compute the best or worst NE (in terms of the defender utility). Note that for general normal-form 2-player games, it is NP-hard to maximize a player's Nash equilibrium utility [5].

Theorem 4.1. *[16] There is a $poly(n)$ time algorithm to compute the Strong Stackelberg equilibrium for security games over $\mathcal{E}$,* if and only if *there is a $poly(n)$ time algorithm to solve the DBR problem over $\mathcal{E}$.*

Theorem 4.2. *[16] There is a $poly(n)$ time algorithm to compute the best and worst (for the defender) Nash equilibrium for security games over $\mathcal{E}$,* if and only if *there is a $poly(n)$ time algorithm to solve the DBR problem over $\mathcal{E}$. Here, by "best/worst" we mean the NE that maximizes/minimizes the defender's utility.*

5 Take-Aways

First, the complexity of a security game is fully captured by the set system $\mathcal{E}$, as summarized in the following corollary of Theorem 3.1, 4.1 and 4.2. Therefor, it should only be a modeling choice, while not the computational concern, about which equilibrium concept to pick in real applications.

Corollary 5.1. *For any set system $\mathcal{E}$, the following four problems reduce to* each other *in polynomial time: (1) Combinatorial optimization over $\mathcal{E}$ for non-negative linear objectives; (2) Solving zero-sum security games over $\mathcal{E}$; (3) Computing Strong Stackelberg equilibrium for security games over $\mathcal{E}$; (4) Computing best/worst (for the defender) Nash equilibrium for security games over $\mathcal{E}$.*

Second, Corollary 5.1 should guide our algorithm design in concrete applications. By examining the DBR problem, we can get a rough sense about how difficult the presented security game is, and what kind of algorithms can we hope for.

Finally, Corollary 5.1 provides a more convenient way for us to theoretically prove the computational complexity of security games, since the complexity of DBR problem is much easier to analyze. In fact, Corollary 5.1 simultaneously implies the computational complexity for solving various types of security games. For example, when $\mathcal{E}$ is any matroid set system or when the optimization over $\mathcal{E}$ has a min-cost flow formulation, the equilibrium can be computed efficiently. On the other hand, when the optimization over $\mathcal{E}$ is a coverage problem or packing problem, the equilibrium computation for these security games is NP-hard in general.

[7]Technically, the defender can still commit to play some strategy. However, the attacker cannot observe or verify the defender's strategy, making the commitment not effective.

References

[1] Sayan Bhattacharya, Vincent Conitzer, and Kamesh Munagala. Approximation algorithm for security games with costly resources. In *Internet and Network Economics*, pages 13–24. Springer, 2011.

[2] Vicki Bier, Santiago Oliveros, and Larry Samuelson. Choosing what to protect: Strategic defensive allocation against an unknown attacker. *Journal of Public Economic Theory*, 9(4):563–587, 2007.

[3] Matthew Brown, Arunesh Sinha, Aaron Schlenker, and Milind Tambe. One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats. In *AAAI conference on Artificial Intelligence (AAAI)*, 2016.

[4] Eric Budish, Yeon-Koo Che, Fuhito Kojima, and Paul Milgrom. Designing random allocation mechanisms: Theory and applications. *The American Economic Review*, 103(2):585–623, 2013.

[5] Vincent Conitzer and Tuomas Sandholm. New complexity results about Nash equilibria. *Games and Economic Behavior*, 63(2):621–641, 2008.

[6] Fei Fang, Albert Xin Jiang, and Milind Tambe. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 957–964. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[7] Fei Fang, Thanh H. Nguyen, Rob Pickles, Wai Y. Lam, Gopalasamy R. Clements, Bo An, Amandeep Singh, Milind Tambe, and Andrew Lemieux. Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *Proceedings of the Twenty-Eighth Innovative Applications of Artificial Intelligence Conference*, 2016.

[8] Van EJ Leeuwen et al. *Optimization and approximation on systems of geometric objects*. 2009.

[9] John A Major. Advanced techniques for modeling terrorism risk. *The Journal of Risk Finance*, 4(1):15–24, 2002.

[10] Marios Mavronicolas, Loizos Michael, Vicky Papadopoulou, Anna Philippou, and Paul Spirakis. The price of defense. In *Mathematical Foundations of Computer Science 2006*, pages 717–728. Springer, 2006.

[11] Marios Mavronicolas, Vicky Papadopoulou, Anna Philippou, and Paul Spirakis. A graph-theoretic network security game. In *Internet and Network Economics*, pages 969–978. Springer, 2005.

[12] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed ARMOR protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[13] Todd Sandler et al. Terrorism & game theory. *Simulation & Gaming*, 34(3):319–337, 2003.

[14] Todd Sandler et al. Counterterrorism a game-theoretic analysis. *Journal of conflict resolution*, 49(2):183–200, 2005.

[15] Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordonez, and Milind Tambe. IRIS - a tool for strategic security allocation in transportation networks. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems - Industry Track*, 2009.

[16] Haifeng Xu. The mysteries of security games: Equilibrium computation becomes combinatorial algorithm design. *http://arxiv.org/abs/1603.02377*, 2015.