One Size Does Not Fit All: A Game-Theoretic Approach for Dynamically and Effectively Screening for Threats[*]

Matthew Brown, Arunesh Sinha, Aaron Schlenker, Milind Tambe
University of Southern California
Los Angeles, CA 90089, USA
{mattheab, aruneshs, aschlenk, tambe}@usc.edu

Abstract

An effective way of preventing attacks in secure areas is to screen for threats (people, objects) before entry, e.g., screening of airport passengers. However, screening every entity at the same level may be both ineffective and undesirable. The challenge then is to find a dynamic approach for randomized screening, allowing for more effective use of limited screening resources, leading to improved security. We address this challenge with the following contributions: (1) a *threat screening game* (TSG) model for general screening domains; (2) an NP-hardness proof for computing the optimal strategy of TSGs; (3) a scheme for decomposing TSGs into subgames to improve scalability; (4) a novel algorithm that exploits a compact game representation to efficiently solve TSGs, providing the optimal solution under certain conditions; and (5) an empirical comparison of our proposed algorithm against the current state-of-the-art optimal approach for large-scale game-theoretic resource allocation problems.

Introduction

Screening people before allowing entry into a secure area is a standard practice throughout the world, e.g., screening resources are used to secure border crossings, sports stadiums, government buildings, etc. Of course, a majority of people will be familiar with airport passenger screening, where each passenger must pass through physical screening consisting of a combination of multiple types of resources (e.g. x-ray and walk-through metal detector) before boarding their flight. Given the significant projected future growth in aviation, agencies such as the Transportation Security Administration (TSA) in the United States are developing dynamic, risk-based screening approaches which optimize the use of resources so as to maintain a high level of security while handling increased passenger volume (AAAE 2014).

This paper considers domains where a screener inspects a set of screenees with the goal of preventing a potential adversary from taking an attack method through screening to attack within a secure area. For example, terrorists with non-metallic explosives may attempt to pass through airport screening undetected in order to attack a particular flight. The screener utilizes different types of screening resources that have: (i) different levels of effectiveness for detecting different attack methods; and (ii) different capacities in terms of the number of screenees that can be processed within a given time window. Effective screening may require assigning multiple screening resource types to inspect a screenee, but the screener may not be able to use the most effective screening resource type combination for every screenee. Hence, the screener may utilize available information to categorize screenees to help determine the appropriate level of scrutiny to apply during screening.

To address the challenge of how to optimally utilize limited screening resources so as to minimize the expected consequences of an attack, we introduce a formal *threat screening game* (TSG) model. TSGs are played between a screener and an adversary, where the screener commits to a screening strategy, assigning a randomized combination of screening resource types to each screenee category. The adversary is able to observe the screening strategy and best responds by selecting an attack method and a screenee category to pose as during screening. The objective of the screener is to minimize the worst-case expected consequences of an attack across all screenee categories and all attack methods.

The TSG model is inspired by research on security games (Jain et al. 2010a; An et al. 2011; Korzhyk, Conitzer, and Parr 2010; Pita et al. 2011; Shieh et al. 2014) and related variants, e.g. audit games (Blocki et al. 2013; 2015), adversarial patrolling games (Basilico, Gatti, and Amigoni 2009; Vorobeychik et al. 2014). Research on security games has provided models and algorithms for generating randomized allocations of limited security resources to protect against an adaptive adversary. However, there exist fundamental differences between threat screening domains and standard security game domains: (i) large numbers of non-adversarial screenees that must be processed and thus affect the screening of the adversary, and (ii) multiple screening resource types with varying efficacies and efficiencies which can combine to *work in teams*. The result is that such domains not only require a new game model (TSG), but also fundamentally new algorithms to handle scalability for TSGs.

Our approach to handle scalability in TSGs involves compactly representing the game, in the process advancing the state of the art in randomized allocation (Budish et al. 2013). In particular, TSGs do not satisfy the conditions identified

[*]Matthew Brown and Arunesh Sinha are both first authors of this paper.

in (Budish et al. 2013) that are required for a lossless compact allocation representation. We provide an efficiently computable compact representation for TSGs which is lossless under certain conditions, thereby providing a generalizable approach for handling scenarios in randomized allocation beyond those identified in the current state of the art.

While airport passenger screening is our motivating domain, the purpose of this paper is to introduce models, algorithms, and insights that are generally applicable to screening for a variety of both physical and non-physical threats. Our contributions include: (1) the general TSG model; (2) an NP-hardness proof for computing the optimal strategy of TSGs; (3) a scheme for decomposing TSGs into subgames to improve scalability; (4) a novel algorithm that exploits a compact game representation to efficiently solve TSGs, providing the optimal solution under certain conditions; and (5) an empirical comparison of the proposed algorithm against the current state-of-the-art approach for large-scale game-theoretic allocation problems. Finally, we evaluate the potential benefit of using a screening strategy obtained by solving a TSG over less dynamic screening strategies.[1]

Motivating Domain

While the TSG model has broad applicability, in this section we focus on one concrete domain. In the United States, the Transportation Security Administration (TSA) screens around 800 million airport passengers annually. The TSA utilizes a number of resources for screening passengers, e.g., x-ray machines, walk-through metal detectors, advanced imaging technology machines. Each passenger is required to go through a combination of screening resources before boarding their flight in order to minimize the threat of a terrorist passing through screening and attacking a flight.

The TSA's new DARMS (Dynamic Aviation Risk Management Solution) initiative fundamentally re-envisions aviation security at all airports nationwide (AAAE 2014). In our joint work with the TSA, we focus solely on the passenger screening component of DARMS. Whereas the TSA previously screened all passengers equally, recently they have implemented risk-based screening through programs such as TSA Pre✓® in which passengers can choose to submit to background checks in order to receive expedited screening. The idea being that fewer resources should be dedicated to screening lower risk passengers and more resources dedicated to screening higher risk passengers, improving both overall screening efficiency and efficacy simultaneously.

In DARMS, the TSA assigns each passenger a risk level based on available information such as flight history, frequent flyer membership, TSA Pre✓® status, etc. The TSA also assigns a value to each flight that measures its attractiveness as a target for terrorists based on gathered intelligence. The innovation in DARMS is that the *screening for each passenger is conditioned on both the passenger's risk level and flight*. Our goal is to exploit this additional flexibility by using the TSG model to compute the optimal dynamic screening strategy, given the available screening resources.

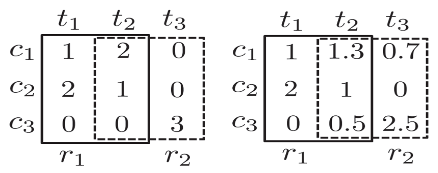
[1]All proofs in this paper can be found in the online Appendix at http://teamcore.usc.edu/papers/2016/DARMS.Appendix.pdf

Threat Screening Game Model

A *threat screening game* (TSG) is a Stackelberg game played between the screener (leader) and an adversary (follower) in the presence of a set of non-player screenees passing through a screening checkpoint operated by the screener. As a Stackelberg game, the screener commits to a randomized screening strategy, while the adversary is able to observe the screening strategy and select a best response. The complete specification of a TSG includes the following:

- *Time windows:* Screening problems feature a temporal dimension, as screenees do not arrive all at once, but rather over a period of time. We model this by slicing the game into a set of time windows W. We use superscript w for relevant variables to indicate the time window.
- *Screenee categories*: Screenees may have defining characteristics upon which their screening can be conditioned, e.g., risk level and flight in DARMS. Screenees with identical defining characteristics can be aggregated into a screenee category $c \in C$ as they are functionally indistinguishable within the context of the game. The total number of screenees in each category c is N_c and the number of screenees in c that arrive at the screening checkpoint during time window w is N_c^w. We assume a constant arrival rate for screenees within each screenee category and time window. All screenees in a category arriving in the same time window are screened equally in expectation according to the randomized screening strategy.
- *Adversary actions*: The actions for the adversary consist of selecting a time window $w \in W$ to go through screening, a screenee category $c \in C$ to pose as during screening, and an attack method $m \in M$. An example adversary action is w = *8:00AM-9:00AM*, c = {*HighRisk, Flight1*}, and m = *on-body non-metallic explosives*.
- *Adversary types*: There may be defining characteristics that an adversary cannot control, e.g., TSA-assigned risk level. Thus, there may be restrictions placed on the actions an adversary can select based on their defining characteristics. Therefore, we refer to an *adversary type* $\theta \in \Theta$, which restricts the adversary to select from screenee categories $C_\theta \subset C$. The adversary knows their own type, but the screener does not. The screener only knows a prior distribution $\mathbf{z}$ over the adversary types.
- *Resource types*: The set of screening resource types is R and all resources of type $r \in R$, e.g., all walk-through metal detectors, can be used to screen a combined total of at most L_r^w screenees during time window w.
- *Team types*: Screenees must be screened by one or more resources types, e.g., walk through metal detector *and* x-ray machine. Each unique combination of resources types constitutes a screening team type t. The set of all valid team types, denoted by T, is given a priori.
- *Team type effectiveness*: Team types vary in their ability to detect different attack methods. For team type t, E_m^t is the probability of detection against attack method m.

Pure strategy A pure strategy P for the screener can be represented by $|W|$ non-negative integer-valued matrices



(a) Pure Strategy (b) Marginal Strategy

Figure 1: TSG Strategies

P^w of size $|C| \times |T|$. The c, t entry in P^w is the number of screenees in c assigned to be screened by team type t during time window w, which we denote as $P^w_{c,t}$. Pure strategy P must assign every screenee to be screened by a team type while satisfying the resource type capacity constraints for each time window. The set of all valid pure strategies $\hat{P}$ is given by the assignments that satisfy the constraints:

$$\textstyle\sum_{t \in T} I^t_r \sum_{c \in C} P^w_{c,t} \leq L^w_r \quad \forall w, \forall r \tag{1}$$

$$\textstyle\sum_{t \in T} P^w_{c,t} = N^w_c \quad \forall w, \forall c \tag{2}$$

where I^t_r is an indicator function returning 1 if team type t contains resource type r and 0 otherwise. We assume $\hat{P} \neq \emptyset$, i.e., it is possible to assign every screenee to a team type.

Example TSG A pure strategy is shown in Figure 1(a) for an example TSG consisting of one time window $W = \{w_1\}$, three screenee categories $C = \{c_1, c_2, c_3\}$, two screening resource types $R = \{r_1, r_2\}$, and three screening team types $T = \{t_1, t_2, t_3\}$ with $t_1 = \{r_1\}$, $t_2 = \{r_1, r_2\}$, and $t_3 = \{r_2\}$. Each screenee category contains three screenees, i.e., $N^{w_1}_{c_1} = N^{w_1}_{c_2} = N^{w_1}_{c_3} = 3$ and thus by Equation 2 each row in the matrix must sum to three. Each resource type can screen at most six screenees, i.e., $L^{w_1}_{r_1} = L^{w_1}_{r_2} = 6$, and thus by Equation 1 the total assignments corresponding to r_1 (solid line box) and the total assignments corresponding to r_2 (dashed line box) must each be less than or equal to six.

Mixed Strategy Given a mixed strategy $\mathbf{q}$ over all valid pure strategies $\hat{P}$ (i.e., $\sum_{P \in \hat{P}} q_P = 1$, $0 \leq q_P \leq 1$), we can compute the expected (marginal) number of screenees in c assigned to be screened by team type t during time window w as $n^w_{c,t} = \sum_P q_P P^w_{c,t}$. The values of $n^w_{c,t}$ form the marginal strategy $\mathbf{n}$, with an example shown in Figure 1(b).

Utilities Since all screenees in category c are screened equally in expectation, we can interpret $n^w_{c,t}/N^w_c$ as the probability that a screenee in category c arriving during time window w will be screened by team type t. Then, the probability of detecting an adversary type in category c during time window w using attack method m is given by $x^w_{c,m} = \sum_t E^t_m n^w_{c,t}/N^w_c$. The payoffs for the screener are given in terms of whether adversary type θ chooses screenee category c and is either detected during screening, denoted as $U^d_{s,c}$, or is undetected during screening, denoted as $U^u_{s,c}$. Similarly, the payoffs for adversary type θ are given in terms of whether θ chooses screenee category c and is either detected during screening, denoted as $U^d_{\theta,c}$, or is undetected during screening, denoted as $U^u_{\theta,c}$. As the nature of our motivating domain is zero-sum, we assume a zero-sum game such that $U^d_{\theta,c} = -U^d_{s,c}$ and $U^u_{\theta,c} = -U^u_{s,c}$.

Given adversary type θ's choice of time window w, screenee category c, and attack method m, the screener's expected utility is given by $U_s = x^w_{c,m} U^d_{s,c} + (1 - x^w_{c,m}) U^u_{s,c}$ and the expected utility for adversary type θ is $U_\theta = -U_s$.

Computing TSG Equilibria

The optimal mixed strategy for a zero-sum TSG can be obtained by solving the linear program $MixedStrategyLP$

$$\max_{\mathbf{n},\mathbf{q},\mathbf{s},\mathbf{x}} \quad \textstyle\sum_{\theta \in \Theta} z_\theta s_\theta \tag{3}$$

$$s_\theta \leq x^w_{c,m} U^d_{s,c} + (1 - x^w_{c,m}) U^u_{s,c} \quad \forall w, \forall c, \forall m \tag{4}$$

$$x^w_{c,m} = \textstyle\sum_{t \in T} E^t_m \frac{n^w_{c,t}}{N^w_c} \quad \forall w, \forall c, \forall m \tag{5}$$

$$n^w_{c,t} = \textstyle\sum_{P \in \hat{P}} q_P P^w_{c,t} \quad \forall w, \forall c, \forall t \tag{6}$$

$$\textstyle\sum_{P \in \hat{P}} q_P = 1, \quad q_P \geq 0 \quad \forall P \tag{7}$$

Equation 3 is the objective function which maximizes the summation of the screener's worst case expected utility against adversary type θ, s_θ, multiplied by the probability of encountering θ, z_θ, over all adversary types $\theta \in \Theta$. Equation 4 constrains s_θ to be worst case screener utility over all $c \in C_\theta$, $w \in W$, and $m \in M$ that adversary type θ could choose. Equation 5 calculates the detection probabilities $\mathbf{x}$ from the marginal strategy $\mathbf{n}$. Equation 6 calculates the marginal strategy $\mathbf{n}$ from the mixed strategy $\mathbf{q}$. Equations 7 ensures $\mathbf{q}$ is a valid probability distribution over $\hat{P}$.

Decomposition A potential way to improve computational efficiency when solving for the screener strategy is to decompose the TSG into sub-games G^w for each time window. Since the TSG is sliced into $|W|$ time windows, the overall screener mixed strategy can be computed by solving G^w (assuming adversary types conduct one attack in each time window) separately and then combining the mixed strategies from each sub-game. The optimization problem $MixedStrategyLP(w)$ is similar to $MixedStrategyLP$, except with only the constraints for time window w. This decomposition scheme produces the optimal solution for the type of zero-sum TSGs we consider in this paper.

Theorem 1. *Decomposition produces the optimal mixed strategy for a TSG when there exists a real number j such that $j > 0$ and all sub-games satisfy $U^u_{s,c} = -jU^u_{\theta,c}$ and $U^d_{s,c} = -jU^d_{\theta,c}$, $\forall c \in C$.*

However, decomposition is not guaranteed to produce the optimal screening strategy for general-sum TSGs. A counterexample showing the potential suboptimality of decomposition for general-sum TSGs is located in the Appendix.

Complexity Despite decomposing a TSG into sub-games, solving G^w can be computationally difficult as the TSG inputs, and in particular the sets C and T, increase in size.

Theorem 2. *$MixedStrategyLP(w)$ is NP-Hard to solve.*

The challenge is that the number of valid pure strategies $|\hat{P}|$ is exponential in the inputs to the TSG. Thus, for any realistically-sized TSG, it would be impossible to even properly formulate $MixedStrategyLP(w)$ in practice as $\hat{P}$ would not fit into memory. Previous work for handling large pure strategy spaces in Stackelberg games can be categorized into two broad approaches: (1) *marginal-based approaches* (Kiekintveld et al. 2009; Letchford and Conitzer 2013) which exploit compact game representations, and (2) *column generation* (Jain et al. 2010b; Yang et al. 2013) which exploits small support set size in the mixed strategy. Our empirical results show that our novel marginal-based approach significantly outperforms column generation with respect to both runtime and solution quality. Hence, we adopt marginal-based approaches for solving TSGs.

Marginals $MixedStrategyLP(w)$ explicitly enumerates the pure strategies in $\hat{P}$ to compute the marginal strategy in Equation 6. An alternative approach is to bypass the enumeration of pure strategies and the modeling of the mixed strategy $\mathbf{q}$ entirely, instead directly optimizing the marginal assignments in $\mathbf{n}$. The optimal screener marginal strategy for a zero-sum TSG can be computed by solving the linear program $MarginalStrategyLP(w)$ which is obtained by Equations 3-5 from $MixedStrategyLP(w)$ and then:

$$\sum_{t\in T} I_r^t \sum_{c\in C} n_{c,t}^w \leq L_r^w \quad \forall r \tag{8}$$

$$\sum_{t\in T} n_{c,t}^w = N_c^w, \quad n_{c,t}^w \geq 0 \quad \forall c,t \tag{9}$$

Previously, the constraints from Equations 1 and 2 were implicitly encoded in $MixedStrategyLP(w)$ by considering an explicit enumeration of only valid pure strategies. Thus, any mixed strategy would respect Equations 1 and 2 by definition. $MarginalStrategyLP(w)$ does not reason over pure strategies and thus needs Equations 8 and 9 to explicitly enforce the resource type capacity constraints and screenee category assignment constraints in the marginal strategy.

Algorithmic Approach

Despite satisfying Equations 8 and 9, the optimal marginal strategy $\mathbf{n}^*$ from $MarginalStrategyLP(w)$ may not correspond to any valid mixed strategy $\mathbf{q}$ due to the integral screenee category assignments in pure strategies. Such marginal strategies are said to be *non-implementable*. We present an example of a non-implementable marginal strategy in the Appendix. Our goal is to compute the optimal *implementable* marginal strategy for a given TSG.

While previous work in security games has explored the non-implementability of marginals (Kiekintveld et al. 2009; Letchford and Conitzer 2013), the identified conditions focus specifically on spatio-temporal resource constraints, e.g., constraints on the scheduling and routing of resources. In contrast, the non-implementability issues in TSGs arise out of conflicting demands on resource types — from different team types — potentially resulting in constraint violations. Outside of the scope of security games, work on general randomized resource allocation, specifically (Budish et al. 2013), has explored such issues and provides guarantees for the implementability of marginal strategies if the problem constraints form a structure called a *bihierarchy*, which we define below. Unfortunately, only the most trivial TSGs form a bihierarchy as initially specified and an algorithm that handles non-bihierarchies is not provided in (Budish et al. 2013). Therefore, we have developed a novel approach for manipulating the constraints in TSGs to obtain a bihierarchy and ensure implementability of the marginal strategy.

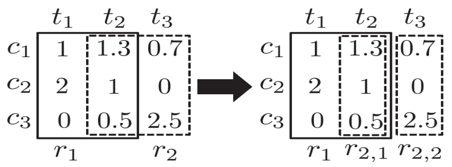


Figure 2: Resolving Overlaps to Form a Bihierarchy

Notation and Definitions In this section, we focus on a single time window, and thus drop the superscript w. The marginal assignments $\mathbf{n}$ form a $|C| \times |T|$ matrix. Each assignment constraint in $MarginalStrategyLP(w)$, notably Equations 8 and 9, is a summation of $n_{c,t}$ over a set $S \subset C \times T$ with an integral upper bound. For example, based on Equation 9, $\{\{c_1, t_1\}, \{c_1, t_2\}, \{c_1, t_3\}\}$ form one constraint subset S for the example TSG from Figure 1. The set of all such S form a *constraint structure* H. To simplify our presentation, we introduce some shorthand. We use $\mathbf{n}[S]$ in place of $\sum_{(c,t)\in S} n_{c,t}$ and $\mathbf{n}(H)$ to denote the set $\{\mathbf{n}|\mathbf{n}$ satisfies constraints given by constraint structure $H\}$.

We say $\mathbf{n}$ is *implementable* with respect to H if $\mathbf{n} \in \mathbf{n}(H)$ and there exists a mixed strategy $\mathbf{q}$ such that $\mathbf{n} = \sum_{P\in\hat{P}} q_P P$. H is *universally implementable* if all $\mathbf{n} \in \mathbf{n}(H)$ are implementable. We say H is a *hierarchy* if, for every pair of elements $S_1, S_2 \in H$, we have $S_1 \subset S_2$, $S_2 \subset S_1$, or $S_1 \cap S_2 = \emptyset$. Thus, H is a hierarchy if, for any two constraint sets, one is a subset of the other or they share no common elements. We say H is a *bihierarchy* if there exists hierarchies H_1 and H_2 such that $H = H_1 \cup H_2$ and $H_1 \cap H_2 = \emptyset$. Utilizing these definitions, we show the following:

Theorem 3. *For TSGs, a bihierarchical constraint structure is necessary and sufficient for universal implementability.*

Theorem 3 implies that we can guarantee the solution to $MarginalStrategyLP(w)$ is the *optimal implementable* marginal strategy if and only if the constraints for the TSG form a bihierarchy. TSGs have one constraint hierarchy H_1 containing the row constraints for each screenee category c: $\sum_{t\in T} n_{c,t} = N_c$. However, the set of column constraints H_2, one for each resource type r, may not form a hierarchy: $\sum_{t\in T} I_r^t \sum_{c\in C} n_{c,t} \leq L_r$. To capture the overlap between resource type constraints, we denote the constraint subset for every resource type r as $S_r = C \times \{t \mid t \in T, r \in t\}$, and the intersection $S_r \cap S_{r'}$ as $S_{r,r'}$. An intersection is *resolved* if we have $S_{r'} \subset S_r$ or $S_r \subset S_{r'}$ or $S_{r,r'} = \emptyset$. If all intersections $S_{r,r'}$ are resolved then H_2 forms a hierarchy by definition. Since H_2 does not form a hierarchy for most TSGs, we concentrate on converting H_2 into a hierarchy in order to achieve universal implementability for a given TSG.

Marginal Guided Algorithm (MGA) Our approach focuses on resolving intersections between the resource type constraints in H_2 to produce a hierarchy, which results in the TSG constraint structure forming a bihierarchy.

Figure 2 shows a possible way of resolving overlapping constraints in H_2 for the example TSG. Note that the constraints (on the left) do not form a hierarchy as $S_{r_1,r_2} =$

$C \times \{t_2\}$ and $S_{r_1} \not\subseteq S_{r_2}$ and $S_{r_2} \not\subseteq S_{r_1}$. Thus, a possible way of resolving the intersection between r_1 and r_2 is to split r_2 into $r_{2,1}$ and $r_{2,2}$, with new capacity constraints $\mathbf{n}[S_{r_{2,1}}] \leq L_{r_{2,1}}$ and $\mathbf{n}[S_{r_{2,2}}] \leq L_{r_{2,2}}$ respectively to replace the original capacity constraint $\mathbf{n}[S_{r_2}] \leq L_{r_2}$ on resource type r_2. Resolving this intersection results in H_2 forming a hierarchy, as $S_{r_{2,1}} \subset S_{r_1}$ and $S_{r_{2,1},r_{2,2}} = \emptyset$ and $S_{r_1,r_{2,2}} = \emptyset$. H_1 and H_2 now combine to form a bihierarchy.

However, it must hold that $L_{r_{2,1}} + L_{r_{2,2}} = L_{r_2} = 6$, yet notice that there are no satisfying, integral values of $L_{r_{2,1}}$ and $L_{r_{2,2}}$ (e.g., $\{2,4\}$,$\{3,3\}$,$\{4,2\}$) that still capture the example marginal strategy, specifically $\mathbf{n}[S_{r_{2,1}}] = 2.8$ and $\mathbf{n}[S_{r_{2,2}}] = 3.2$. That is, manipulating the constraints in this way would cut out the marginal strategy from the solution space. Therefore, if the marginal strategy was in fact optimal, then breaking down r_2 would remove an important part of the feasible space, resulting in a loss of solution quality. Instead, it would have better to break down r_1 into $r_{1,1}$ and $r_{1,2}$ with $L_{r_{1,1}} = L_{r_{1,2}} = 3$, as resolving the intersection that way produces a bihierarchy that still captures the marginal strategy, specifically $\mathbf{n}[S_{r_{1,1}}] = 3$ and $\mathbf{n}[S_{r_{1,2}}] = 2.8$.

The example above clearly highlights the importance of resolving intersections intelligently. Therefore, we present the Marginal Guided Algorithm (MGA) that uses the optimal marginal strategy $\mathbf{n}^*$ from $MarginalStrategyLP$ to search for the optimal implementable marginal strategy, the intuition being that it exists either at, or near, $\mathbf{n}^*$.

The first step in MGA (shown in Algorithm 1) is to solve $MarginalStrategyLP$ to obtain $\mathbf{n}^*$ (Line 1), which may be non-implementable. MGA then builds a tree by starting with a root node that contains the original constraint structure (Line 2), and repeats the resolution steps, described below, until there are no unresolved intersections. We resolve the intersections guided by $\mathbf{n}^*$ using three types of resolution to replace the original resource type constraint for r:

- *Integral resolution*: If $\mathbf{n}^*[S_{r,r'}]$ is integral then add the pair of constraints $\mathbf{n}[S_{r,r'}] \leq \mathbf{n}^*[S_{r,r'}]$, $\mathbf{n}[S_r \backslash S_{r,r'}] \leq L_r - \mathbf{n}^*[S_{r,r'}]$. $\mathbf{n}^*$ still satisfies the resulting constraints.
- *Slack resolution*: If $L_r - \mathbf{n}^*[S_r] \geq 1$ then add the pair of constraints $\mathbf{n}[S_{r,r'}] \leq \lceil \mathbf{n}^*[S_{r,r'}] \rceil$, $\mathbf{n}[S_r \backslash S_{r,r'}] \leq \lceil \mathbf{n}^*[S_r \backslash S_{r,r'}] \rceil$. $\mathbf{n}^*$ still satisfies the resulting constraints.
- *Tight resolution*: If $L_r - \mathbf{n}^*[S_r] < 1$ then produce two sets, each containing a pair of constraints: $\mathbf{n}[S_{r,r'}] \leq \lceil \mathbf{n}^*[S_{r,r'}] \rceil$, $\mathbf{n}[S_r \backslash S_{r,r'}] \leq L_r - \lceil \mathbf{n}^*[S_{r,r'}] \rceil$ and $\mathbf{n}[S_{r,r'}] \leq \lfloor \mathbf{n}^*[S_{r,r'}] \rfloor$, $\mathbf{n}[S_r \backslash S_{r,r'}] \leq L_r - \lfloor \mathbf{n}^*[S_{r,r'}] \rfloor$, respectively. $\mathbf{n}^*$ may not satisfy either of the two new resulting sets of constraints.

A single leaf node is added for integral and slack resolutions, while two leaf nodes are added for tight resolutions. Thus, we order the intersections so that integral and slack resolutions happen first (Line 3 and 7), to avoid tight resolution and thus preserve $\mathbf{n}^*$. After every resolution, leaves are added to the tree (Line 6) that contain all constraints of the parent node except with the constraint for r removed and the new pair of constraints added, then the intersections are recomputed (Line 7). At the end of the loop, each leaf i contains a bihierarchy H_i, as all intersections have been resolved. A convex hull of the sets of points $\mathbf{n}(H_i)\ \forall i$ is computed, and

Algorithm 1: MGA(H)

```
1 $\mathbf{n}^* \leftarrow MarginalStrategyLP(H)$
2 $Tree.Root \leftarrow H$
3 $I \leftarrow Calc\&OrderIntersections(H)$
4 while ($I \neq \{\emptyset\}$) do
5     $C \leftarrow ResolveIntersection(I, \mathbf{n}^*)$
6     $Tree \leftarrow UpdateTree(Tree, C)$
7     $I \leftarrow Calc\&OrderIntersections(Tree.Leaf)$
8 $Hull \leftarrow ConvexHull(Tree.Leaf)$
9 return $\mathbf{n}', \lambda, \{\mathbf{n}_i\} \leftarrow MarginalStrategyLP(Hull)$
```

$MarginalStrategyLP$ is solved using the convex hull instead of Equations 8 and 9 to get the marginal strategy $\mathbf{n}'$.

Convex Hull As each bihierarchy is a set of linear constraints, let bihierarchy H_i be written as $A_i\mathbf{n} \leq b_i$ for matrix A_i and vector b_i. Thus, by definition $\mathbf{n}(H_i) = \{\mathbf{n} | A_i\mathbf{n} \leq b_i\}$. The natural way to represent the convex hull is $conv(\mathbf{n}(H_1), \ldots, \mathbf{n}(H_k)) = \{\mathbf{n} | \mathbf{n} = \sum_i \lambda_i \mathbf{n}_i, A_i\mathbf{n}_i \leq b_i, \lambda_i \geq 0, \sum_i \lambda_i = 1\}$. We use a result from (Balas 1985) that represents the convex hull using *linear constraints*: $conv(\mathbf{n}(H_1), \ldots, \mathbf{n}(H_k)) = \{\mathbf{n} | \mathbf{n} = \sum_i \mathbf{n}_i, A_i\mathbf{n}_i \leq \lambda_i b_i, \lambda_i \geq 0, \sum_i \lambda_i = 1\}$. MGA outputs the values of λ_i and $\mathbf{n}_i$ that form the returned solution $\mathbf{n}'$ (Line 10).

The intuition behind taking the convex hull is that the bihierarchies capture many pure strategies near $\mathbf{n}^*$. Coupled with the fact that the optimal implementable strategy is a convex combination of pure strategies possibly near $\mathbf{n}^*$, the convex hull of the bihierarchies should at least be close to the optimal implementable strategy. This assumption is justified by the empirical results from tests on MGA. Importantly, we show $\mathbf{n}'$ is always implementable and give conditions under which $\mathbf{n}'$ is the optimal implementable marginal strategy:

Lemma 1. *Given any set of constraints H for a TSG, if $\mathbf{n}(H_i) \subseteq \mathbf{n}(H)$ for bihierarchy H_i ($i \in 1, 2, \ldots$), then all $\mathbf{n} \in conv(\mathbf{n}(H_1), \mathbf{n}(H_2), \ldots)$ are implementable.*

Theorem 4. *MGA provides the optimal implementable marginal strategy if tight resolution is never used.*

If tight resolution is never used, then $\mathbf{n}' = \mathbf{n}^*$. In our evaluation, we show that even when tight resolution is used and $\mathbf{n}' \neq \mathbf{n}^*$, $\mathbf{n}'$ is empirically equal to $\mathbf{n}^*$ in terms of screener utility. While not guaranteed, empirically it would thus appear MGA is capable of providing an utility-equivalent *optimal implementable* marginal strategy $\mathbf{n}'$ in place of the possibly non-implementable marginal strategy $\mathbf{n}^*$.

Sampling Viewing λ_i as a probability, we sample a bihierarchy H_i (and $\mathbf{n}_i$) with probability λ_i. We know that $\mathbf{n}_i/\lambda_i \in \mathbf{n}(H_i)$, thus, we use the sampling algorithm for a bihierarchy from (Budish et al. 2013) to efficiently sample a pure strategy from the implementable strategy $\mathbf{n}_i/\lambda_i \in H_i$.

Full Resolution Algorithm (FRA) This alternative to MGA uses a different way of resolving intersections which we call *full resolution*: given an unresolved intersection $S_{r,r'}$, split the constraint for resource type r into $L_r + 1$ possible pairs $\mathbf{n}[S_{r,r'}] \leq L_r - i$, $\mathbf{n}[S_r \backslash S_{r,r'}] \leq i$ for $i \in \{0, 1, \ldots, L_r\}$. FRA works similarly to MGA, except it always uses full resolution. While we prove the following about FRA, it is computationally infeasible in practice.

Theorem 5. *FRA always provides the optimal implementable marginal strategy.*

Evaluation

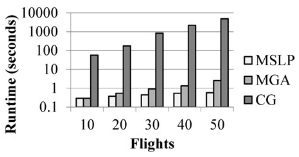
We evaluate the TSG model using experiments inspired by the TSA DARMS passenger screening domain. The game payoffs are zero-sum and randomly generated with $U^u_{\theta,c}$ uniformly distributed in [1,10] and $U^u_{s,c} = -U^u_{\theta,c}$. The remaining game payoffs $U^d_{s,c}$ and $U^d_{\theta,c}$ are fixed to 0. The default settings for each experiment are 5 screenee risk levels, 5 screening resource types, 10 screening team types, 3 attack methods, and 1 time window. All results are averaged over 30 randomly generated game instances.

Algorithmic Approach Figure 3 shows a runtime and solution quality comparison of three approaches for solving TSGs: $MarginalStrategyLP$ (MSLP), Marginal-Guided Algorithm (MGA), and $MixedStrategyLP$ using column generation (CG) with a cutoff after 1000 iterations.

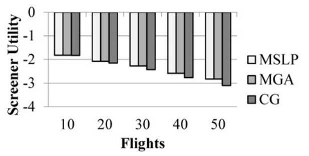
In Figure 3(a), the x-axis denotes the number of flights, and the y-axis the runtime needed to solve the TSG (or reach the iteration cutoff in the case of CG). CG consistently produces the slowest runtimes, as the TSG pure strategy representation results in large support sets within the mixed strategy causing the CG approach to often terminate from reaching the iteration cutoff rather than converging to the optimal solution. In contrast, by exploiting a compact game representation and avoiding pure strategy enumeration, both MSLP and MGA have runtimes which are multiple orders of magnitude lower than CG. Furthermore, MGA is able to guarantee the implementability of the generated marginal strategy with minimal runtime overhead compared to MSLP.

In Figure 3(b), the x-axis again indicates the number of flights, but now the y-axis is the screener utility of the solution returned by solving the TSG (or reaching the iteration cutoff with CG). The solution quality of MSLP (whose marginal strategy may or may not be implementable) represents the theoretical upper bound against which we compare the other two approaches. Despite being an optimal approach, CG provides the worst solution quality of the three approaches again because the iteration cutoff prevents convergence to the optimal solution. Combined with the runtime results, it is clear that CG is not suitable for solving large-scale TSGs. In contrast, MGA, while not guaranteed to be optimal in all cases, matches the optimal solution quality of MSLP empirically for the randomly generated games tested. Thus, in practice, MGA is both an efficient and effective algorithm for real-world threat screening domains.

MGA Analysis To further emphasize the ability of MGA to perform well even for TSGs which require increasingly complex intersection resolutions, we provide a more detailed analysis in Figure 4. The x-axis remains the number of flights, while the left y-axis (bar graph) is the number of leaf nodes generated by MGA, i.e., the number of bihierarchies in the convex hull, and the right y-axis (line graph) is the percentage of the games tested for which MGA generates only a single leaf node. For small numbers of

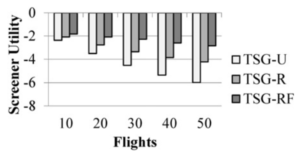


(a) Runtime

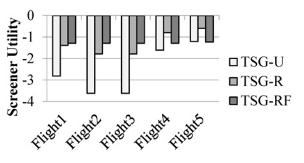


(b) Screener Utility

Figure 3: Comparison of TSG Solution Approaches



(a) Screener Utility



(b) Example Game Instance

Figure 5: Comparison of Screening Approaches

flights, MGA generates a single leaf node in a majority of games resulting in a low average number of leaf nodes. A single leaf node implies that all intersection resolutions were either integral or slack and the optimal solution from MSLP is maintained. As the number of flights increases, the average number of leaf nodes increases, while the percentage of games with a single leaf node decreases. This makes sense as with more screenees it becomes more difficult to eliminate all intersections while avoiding tight resolutions which create multiple leaf nodes. However, even as the number of generated leaf nodes increases, MGA still manages to provide an optimal implementable solution.

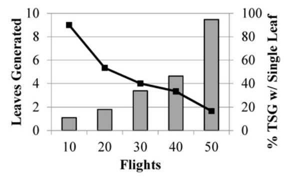


Figure 4: MGA Analysis

Screening Approach The TSA DARMS passenger screening domain utilizes screenee categories defined as ⟨risk level, flight⟩. To show the benefit of using this screenee categorization, we compare the resulting screening strategy (TSG-RF) against two baselines: (1) TSG-U, where screenees are placed in a single uniform category, and (2) TSG-R, where screenees are categorized by ⟨risk level⟩. Figure 5(a) shows the solution quality comparison of the three approaches, where the x-axis is the number of flights and the y-axis is screener utility. As expected, TSG-RF achieves the highest screener utility for all numbers of flights. Figure 5(b) shows a comparison for a specific game instance with 5 flights and provides intuition as to why TSG-RF performs so well. The inability to adjust screening flight by flight results in both TSG-U and TSG-R protecting some flights adequately (Flight 5), while leaving other flights vulnerable (Flight 3), leading to lower solution quality than TSG-RF.

Summary and Related Work

Beyond the (1) TSG model, our contributions are (2) an NP-hardness proof for optimally solving TSGs; (3) a decomposition scheme for TSGs to improve scalability; (4) MGA, a novel approach for efficiently solving TSGs; and (5) an empirical comparison of MGA against column generation, the previous state-of-the-art approach for large-scale game-theoretic resource allocation problems. While we used our joint work with the TSA on airport passenger screening to motivate our work, TSGs are applicable to many other screening domains (e.g., borders, stadiums).

We have dealt with key related work for security games throughout the paper. In addition, the topic of threat screening has been explored extensively, e.g., for shipping containers (Anand et al. 2006), stadium patrons (Ricks et al. 2014), and airport passengers (Nie et al. 2009; McLay, Lee, and Jacobson 2010). However, these approaches do not model the game-theoretic aspect of screening problems and thus does not consider an adversary that exploits screening vulnerabilities. (Wang, Song, and Zhuang 2015) incorporate a game-theoretic approach, but do not consider multiple screening resources types, screenee categories, and attack methods. Therefore, the TSG model provides the most general and extensible framework for game-theoretic threat screening.

Acknowledgments: This research was supported by the United States Department of Homeland Security through the National Center for Risk and Economic Analysis of Terrorism Events (CREATE) under award number 2010-ST-061-RE0001.

References

AAAE. 2014. Transportation security policy. Technical report, American Association of Airport Executives.

An, B.; Tambe, M.; Ordóñez, F.; Shieh, E.; and Kiekintveld, C. 2011. Refinement of strong stackelberg equilibria in security games. In *Proceedings of the 25th AAAI conference on Artificial Intelligence (AAAI'11)*.

Anand, S.; Madigan, D.; Mammone, R.; Pathak, S.; and Roberts, F. 2006. Experimental analysis of sequential decision making algorithms for port of entry inspection procedures. In *Intelligence and Security Informatics*. Springer. 319–330.

Balas, E. 1985. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods* 6(3):466–486.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 57–64. International Foundation for Autonomous Agents and Multiagent Systems.

Blocki, J.; Christin, N.; Datta, A.; Procaccia, A.; and Sinha, A. 2013. Audit games. In *Proceedings of the 23rd international joint conference on Artificial Intelligence (IJCAI'13)*.

Blocki, J.; Christin, N.; Datta, A.; Procaccia, A.; and Sinha, A. 2015. Audit games with multiple defender resources. In *Proceedings of the 29th AAAI conference on Artificial Intelligence (AAAI'15)*.

Budish, E.; Che, Y.-K.; Kojima, F.; and Milgrom, P. 2013. Designing random allocation mechanisms: Theory and applications. *The American Economic Review* 103(2):585–623.

Jain, M.; Tsai, J.; Pita, J.; Kiekintveld, C.; Rathi, S.; Tambe, M.; and Ordóñez, F. 2010a. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces* 40(4):267–290.

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

Kiekintveld, C.; Jain, M.; Tsai, J.; Pita, J.; Ordez, F.; and Tambe, M. 2009. Computing optimal randomized resource allocations for massive security games. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems*.

Korzhyk, D.; Conitzer, V.; and Parr, R. 2010. Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *Proceedings of the 24th AAAI conference on Artificial Intelligence (AAAI)*, 805–810.

Letchford, J., and Conitzer, V. 2013. Solving security games on graphs via marginal probabilities. In *AAAI*. Citeseer.

McLay, L. A.; Lee, A. J.; and Jacobson, S. H. 2010. Risk-based policies for airport security checkpoint screening. *Transportation science* 44(3):333–349.

Nie, X.; Batta, R.; Drury, C. G.; and Lin, L. 2009. Passenger grouping with risk levels in an airport security system. *European Journal of Operational Research* 194(2):574–584.

Pita, J.; Tambe, M.; Kiekintveld, C.; Cullen, S.; and Steigerwald, E. 2011. Guards: Innovative application of game theory for national airport security. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11.

Ricks, B. C.; Nakamura, B.; Almaz, A.; DeMarco, R.; Hui, C.; Kantor, P.; Matlin, A.; Nelson, C.; Powell, H.; Roberts, F.; et al. 2014. Modeling the impact of patron screening at an nfl stadium. In *IIE Annual Conference. Proceedings*, 3086. Institute of Industrial Engineers-Publisher.

Shieh, E. A.; Jiang, A. X.; Yadav, A.; Varakantham, P.; and Tambe, M. 2014. Unleashing dec-mdps in security games: Enabling effective defender teamwork. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, 819–824.

Vorobeychik, Y.; An, B.; Tambe, M.; and Singh, S. 2014. Computing solutions in infinite-horizon discounted adversarial patrolling games. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*.

Wang, X.; Song, C.; and Zhuang, J. 2015. Simulating a multi-stage screening network: A queueing theory and game theory application. In *Game Theoretic Analysis of Congestion, Safety and Security*. Springer. 55–80.

Yang, R.; Jiang, A. X.; Tambe, M.; and Ordóñez, F. 2013. Scaling-up security games with boundedly rational adversaries: A cutting-plane approach. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. AAAI Press.