Strategic Security Placement in Network Domains with Applications to Transit Security

Jason Tsai, Zhengyu Yin, Jun-young Kwak, David Kempe, Christopher Kiekintveld, Milind Tambe

University of Southern California, Los Angeles, CA 90089
{jasontts, zhengyuy, junyounk, dkempe, kiekintv, tambe} @usc.edu

Abstract

Deterministic placement of security personnel creates serious vulnerabilities for any organization attempting to prevent intrusion. Recent work in use at the Los Angeles International Airport (LAX) and in progress with the United States Federal Air Marshal Service (FAMS) has applied game-theoretic analysis to the problem by modeling it as a Stackelberg game wherein security forces are the leaders that commit to a strategy that is observed and countered by attackers.

In this work, we explore efficient techniques for performing the same analysis on games with a graph structure, wherein an attacker must follow a path from an entry point to a target. If we frame these problems in the straightforward manner with leader actions being sets of edges that can be guarded and follower actions being paths from entry to targets, the size of the game increases exponentially, quickly reaching memory limitations when using general Stackelberg solvers.

In this work, we propose a novel linear program that is able to solve this type of problem efficiently. While it provides exact solutions for games where only one checkpoint is allowed, it is an approximation in the general case. Finally, we compare the performance of this and other methods by generating optimal policies for the Seoul Metropolitan Subway in Seoul, South Korea.

1 Introduction

Security of locations within military compounds, airports, and even whole cities presents a serious challenge to law enforcement agencies throughout the world. The primary task in these problems is determining the optimal assignment of available resources. In some cases, there are enough personnel to protect all possible avenues of attack. Oftentimes, however, this is not the case, and there are simply not enough personnel to protect all entry points into an area and organizations must look at how to optimally place their resources within the area.

Our work focuses on a special class of security games that can be represented with a graph structure. A simple example would be a neighborhood in a city where particular buildings are at risk for attack. Suppose there are many possible ways into the neighborhood as well as multiple paths from the entry points to the target buildings. We can model this as a graph with roads as edges, intersections as nodes, critical buildings as sink nodes, and entry points into the neighborhood as source nodes. Thus, the security problem becomes one of determining which edges to place security checkpoints on.

In this subclass of security games, as well as in security games in general, an intelligent and determined adversary should be expected to spend time observing the security practices of the area and planning his attack route based on the defender's policy. Therefore, deterministic allocation of resources, even a rotating schedule that cycles only over long periods of time, will provide predictable weaknesses in the defense that can easily be exploited. A better alternative would be to use some form of randomization in the strategy and protect certain locations with a fixed probability to increase the uncertainty of potential attackers. Furthermore, potential targets within a region inevitably have varying values and risks associated with them. While one target may be an empty warehouse, another may house thousands of people at any given time. Thus, a randomized strategy must correctly account for such variations in the targets that are being protected.

Game theory offers a systematic approach to this problem by providing a framework in which to model the security problem at hand and equilibrium solution concepts to generate optimal security policies. Varying values and risks of the targets for both the attacker and the defender can be modeled by specifying appropriate payoff values in the game. A Stackelberg game model provides the leader/follower framework that we use to evaluate security scenarios. The equilibrium strategies calculated will provide the optimal randomization strategies for the security forces facing intelligent and determined adversaries.

In solving these games, we could use a naïve solution technique wherein we explicitly enumerate all possible paths from sources to sinks as attacker actions and all possible combinations of edges as defender actions and solve the resulting Stackelberg game for a Strong Stackelberg Equilibrium. While this method works, it is prohibitively slow in practice and is primarily useful as a benchmark as we explore more ef-

ficient techniques that exploit the graph and payoff-structure inherent in such games.

We propose graph-based techniques to efficiently solve various classes of these games based on the number of sources, sinks, and available resources. For this work, we assume that the games are zero-sum, wherein objectives that benefit the attacker are correspondingly detrimental to the defender, and look to relax this assumption in the near future. In games with only a single source and a single target, we use a minimum s-t cut strategy to determine an optimal deployment strategy, as done in [Washburn and Wood, 1995]. With multiple targets of equal value or multiple sources, the game can again be optimally solved in polynomial time with minor extensions of the same strategy.

A more interesting class of games would be ones with multiple targets of varying value. For this type of game, we introduce a novel linear programming formulation for the problem. In cases with only one available resource, it is guaranteed to provide optimal solutions. With multiple resources, however, we provide a basic sampling technique that provides an approximation that is within $1 - (1 - \frac{1}{r})^r$ of the optimal solution in polynomial time, where r is the number of resources. Finally, we apply these techniques on a real-world domain, the Seoul Metropolitan Subway, to evaluate their runtime performance and optimality.

2 Motivating Domain

The graph-based model proposed is applicable to security domains where attackers must follow a path to attack targets. One example of such a domain might be bus systems where attackers will travel along bus routes to reach target stations to attack. Another example would be a computer network, where a malicious hacker could find a vulnerable computer and manipulate it for access to adjacent computers in the network, eventually reaching a machine with confidential information. Protection of cities can also be thought of as a graph, with important buildings as targets and the roads through the city as paths that potential attackers may follow.

Subway and commuter rail systems are also an example of a graph-based security domain, and the Seoul Metropolitan Subway (SMS) in Seoul, South Korea represents one of the more complex instances of such a problem. The SMS has over 250 stations throughout Seoul and adjacent provinces.

Given the pattern of attacks on subways and buses in Madrid and London [Thornton, 2005; Blanco *et al.*, 2007], we assume that a potential terrorist enters the system at one of a subset of known locations and plans to damage one out of a set of targets with preferences. We also assume the attacker travels by train from an entry station to target stations. Alternatively, one could consider trains themselves as targets and our modeling of stations as an approximation of the trains' values as they pass through the stations. In this work, we are primarily concerned with law enforcement aboard the trains and will not consider security within the stations themselves.

Modeling this in our graph framework, we can model stations as vertices, train paths between stations as edges, and police patrolling on trains between two stations can be viewed as checkpoints. In practice, while there might not be a law enforcement official aboard every single train that passes between two stations, we simplify the problem for the purposes of the model and assume this is the case. Finally, not all train paths can be protected at all times due to the limited number of law enforcement personnel available.

In the SMS domain, there is obvious variation in target value. While a few stations are in densely populated communities, some stations may not be. For example, a commercial building in downtown Seoul is generally more valuable to terrorists than a station in a residential district. We need to account for these variations of target values when randomizing law enforcement allocation on train routes.

One approach to this class of problems has been explored previously, where they were cast as Stackelberg games with security forces as leaders that commit to a strategy and attackers as followers that observe the strategy and then attack in the manner that achieves the highest expected payoff. However, if we simply transform the problem into a Stackelberg game, we will be required to solve a massive game where the leader's action set comprises all combinations of placing checkpoints on edges and the follower's action set comprises all possible paths from the source to any targets. As we can see, the leader's action set grows combinatorially with the increasing number of checkpoints and edges and the follower's action set can also grow combinatorially with the number of edges as well. As will be shown in Section 6, it quickly becomes infeasible for DOBSS [Paruchuri *et al.*, 2008], the fastest known general solver for this class of problems, to solve the game when the problem scales up. Therefore, the algorithm we design is primarily motivated by scalability in practical implementation.

3 Related Work

There have been significant amounts of work done in security domains that are similar but consequentially distinct from ours. In the ARMOR project at Los Angeles International Airport [Pita *et al.*, 2008] and the Federal Air Marshals Service work [Tsai *et al.*, 2009], the defender announces and commits to a mixed strategy of randomly placing checkpoints over a set of targets, and the attacker can fully observe the mixed strategy and attack the target with the maximal expected payoff. In neither of these domains is there an underlying graph structure with the adversary seeking to move from one node to another node. More importantly, however, the algorithms are Mixed-Integer Linear Programming models that model the game based on the number of actions. As mentioned before, this causes scalability issues very rapidly as we will show in Section 6.

In the Operations Research and Multi-Agent Systems literature, researchers have studied many types of graph-based security games. At the mention of graph-based games, one may immediately think of graphical games, as introduced in [Kearns *et al.*, 2001]. However, graphical games are actually graph-based representations of games with multiple players, which is completely unlike the type of problems we are interested in here. Another, more related class of graph-based games is the pursuit-evasion game [Adler *et al.*, 2003] in which a hunter moves over the graph to catch a rabbit. In

this work, however, the game possesses a temporal component with each player moving to adjacent edges per round and does not have set target nodes that are desirable to reach. Another similar example is a virus spreading problem [Aspnes *et al.*, 2006]. In this problem, a virus starts at a random initial point in the graph and each node chooses to either install anti-virus software at some known cost C, or risk being infected and a loss L if the virus can reach it without being stopped by some intermediate node. This closely resembles the SMS domain where the virus is the attacker and the computers that decide to install anti-virus software represent where the defender places checkpoints. However, the spreading nature of the 'attacker' does not properly model the intelligent, source-to-target behavior that exists in the SMS domain's attacker. As a result, the deterministic nature of solutions for this class of problems cannot be applied to the SMS domain.

Perhaps the most related class of problems is known in Operations Research as network interdiction problems [Wood, 1993; Washburn and Wood, 1995; Israeli and Wood, 2002; Cormican *et al.*, 1998; 1998]. A basic version of this problem is a simple shortest-path network interdiction game [Israeli and Wood, 2002]. It is a Stackelberg game, where a network user as the follower, wishes to traverse a path of minimum-length between a source s and a target t, in a directed graph. The leader can destroy certain arcs, or increase their effective length, and thereby increase the follower's shortest s - t path length. In SMS, this type of solution would translate into a deterministic defensive strategy, which is one of the things we are striving to avoid. The work of [Washburn and Wood, 1995] introduced a two-person zero-sum game for network interdiction which is quite close to our problem and from which we derive our minimum s-t cut algorithms. However, they do not account for targets of varying values, focusing instead on domains where the 'attacker' is simply trying to traverse the graph from source to sink.

4 Background

Our work focuses on applying game-theoretic techniques to a subclass of security games that can be modeled with a graph structure. We propose using graph-based algorithms to solve them. However, since the Stackelberg framework provides a basis for our approach, we introduce it briefly here. Then we discuss how we map our graph-based security domain to a standard Stackelberg game. Finally, we provide a formal overview of the minimum s-t cut algorithms and extensions proposed by [Washburn and Wood, 1995] and their application to our domain.

4.1 Stackelberg Games

In Stackelberg games [von Stackelberg, 1934], there are two players: a leader and a follower. The leader commits to a strategy first, after which the follower makes his strategy choice. Since the follower can optimize his strategy based on the leader's strategy, intuition might suggest that the follower has a strict advantage in this type of game. However, an ability to commit to a *mixed* strategy first actually always weakly increases the leader's expected equilibrium payoff [von Stengel and Zamir, 2004]. Broadly speaking, this is due to the fact that through commitment, the leader is able to constrain the follower into a subset of the action space.

As an example, consider the game illustrated in Table 1. As a simultaneous game, the only pure-strategy Nash equilibrium is when the row player plays A and the column player plays C, giving the row player a payoff of 2. However, if the row player was the leader and the column player the follower in a Stackelberg game, the leader can commit to a uniform mixed strategy of playing each A and B half the time. This will force the optimizing follower to choose action D, giving the leader an expected payoff of 3.5.

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

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

4.2 Graph-based Security Games

In a security game, there is a defender that must continuously defend the designated area and an attacker that can observe the defender's strategy and attack where success seems most likely. This fits neatly into the description of a Stackelberg game if we map the attacker to the follower's role and the defender to the leader's role [Avenhaus *et al.*, 2002; Brown *et al.*, 2006].

As described in Section 1, a graph-based security game is one in which the domain being protected can be mapped into a graph, with nodes as locations and edges as connecting paths between the locations. In such a game, the intruder starts from a source node $s \in S$, where S is the set of source nodes, and attempts to reach one of n targets $t_1, \ldots, t_n$, which have values $c_1, \ldots, c_n \geq 0$. If the intruder reaches t_i without being detected, then the damage is c_i. The defender is allowed to place r checkpoints on the edges of the graph $G(V, E)$.

Thus, in a straightforward model of the problem, each attacker action represents a selection of which s - t path P to take, and each defender action is a set $l \in L$ of edges to place the r checkpoints. The intruder is captured if $P \cap l \neq \emptyset$ and makes it to the target if $P \cap l = \emptyset$. In equilibrium, the defender determines an optimal mixed strategy to commit to, which the attacker observes. Based on the observed strategy, the attacker chooses the path P that has the highest expected reward. We could solve the resulting game using DOBSS, but the combinatorial increase in action space for both attacker and defender quickly result in games that cannot be solved in a reasonable timeframe, as we show in Section 6.

4.3 MCTE

As mentioned in Section 3, [Washburn and Wood, 1995] use minimum s-t cut techniques to solve similar versions of this problem with targets of equal value. We refer to this class of solution techniques as Minimum Cut Techniques with Extensions (MCTE). Since our linear programming solution is based on these methods and [Washburn and Wood, 1995] did not offer formal justification for some of their techniques, we go over them in detail here.

We will begin with the most basic case of the game with one entry point and one target. We then show how to build on this solution by proving methods for handling multiple targets of equal value and multiple entry points. These techniques are based on the insight that when targets are all of equal value, the defender's maximization of expected remaining sum of values of the targets is equivalent to maximizing the chance of capturing the attacker.

Single-Target, Single-Entry

In games with a single entry point and a single target, we have an optimal solution strategy that can be determined in polynomial time by considering the fact that all paths must pass through the minimum s-t cut of the graph. Suppose we have a graph G with a minimum s-t cut, M, of size k, and $r \leq k$ checkpoints are allowed.

Lemma 4.1. *For every graph G, there exists a strategy that guarantees capture with a probability of at least* $\frac{r}{k}$.

Proof. Let the strategy be to place checkpoints on a uniformly random subset $S \subseteq M$ of size r. Any path the intruder chooses will contain at least one edge of the minimum s-t cut and is checked with probability at least $\frac{r}{k}$. Therefore, the strategy guarantees capture with probability at least $\frac{r}{k}$. □

Lemma 4.2. *For every graph G, there exists an intruder strategy against which the defender has no strategy that achieves better capture probability than* $\frac{r}{k}$.

Proof. Because the minimum s-t cut has size k, there is a set of k edge-disjoint s-t paths $P_1, ..., P_k$. For any set of r edges, the probability of being on the attacker's chosen path is at most $\frac{r}{k}$, because no edge lies on two of the paths P_i. Therefore, the overall probability for any randomization over sets of size r is at most $\frac{r}{k}$. □

Theorem 4.3. *A coverage strategy of uniform randomization of* r *checkpoints across the* k *edges of the minimum* s-t *cut is an optimal strategy.*

Theorem 4.3 follows immediately from Lemmas 4.1 and 4.2. Finding a minimum s-t cut can be done in polynomial time, so this algorithm is polynomial time and optimal.

Multiple Targets of Equal Value, Multiple Entry Points

In games where we have multiple targets of equal value and multiple entry points, [Washburn and Wood, 1995] provide constructions for these extensions that allow us to use the simple model presented earlier. Here we provide brief sketches of similar constructions.

For multiple targets of equal value, we can combine the targets into a single super-target. In particular, given $G(V, E)$, with a set, T, of n target nodes, $t_1, t_2, ..., t_n$, let E_t be the set of all edges incident to T and U be the set of nodes adjacent to T. Construct $G'(V', E')$ such that $V' = V - T + \{t'\}$, $E' = E - E_t + U_t$ where U_t is a new set of edges from U to t'. In G', t' is the new target node which will have the same value as each of the nodes in T. Also, we will assume that no attacker will attack via a path that passes through a target and ends on another target. This is a reasonable assumption since all targets are of equal value, so a rational attacker would end his path on the first target reached to minimize his chance of being captured.

For games with multiple entry points where the attacker can decide which to use, we can perform the same transformation on the sources and replace them with a single super-source node.

5 ESVVT

In Section 4.2, we introduced a general solution method using DOBSS that is able to handle any security problem of this type. However, due to the method's inability to scale well, we discussed the use of minimum s-t cut techniques as proposed in [Washburn and Wood, 1995]. This class of solutions, though efficient, lacks the ability to handle the more interesting class of graph-based security games in which targets have varying value.

To address this, we introduce a linear programming approach to solve games with multiple targets of varying value, referring to it as Efficient Solution for Varying Value Targets (ESVVT). While this solution technique provides an exact solution when only one checkpoint is allowed at a time, it is only an approximation when there are more than one allowed at a time. As we show in Section 6, however, the algorithm appears to work well in practice and provides an efficient alternative to a direct application of DOBSS to this class of problems.

Since the game is zero-sum, the defender's optimal strategy will be the assignment of coverage probability to edges that maximizes the expected remaining sum of values of the targets. We assume that there is only one intruder who can observe the exact probability distribution over all combinations of r checkpoints. We phrase this as an LP as follows: for each edge e, we have a variable x_e, which we interpret as the probability of placing a checkpoint on e. $C = \sum_{t_i \in T} c_i$ is the overall value of all targets. For each vertex v, we use d_v to represent the probability of capturing the intruder if the intruder chooses the best source $s \in S$ to start with and follows the best path from s to v, where 'best' for the intruder represents the option with the lowest chance of being captured. The expected damage is $c_i \cdot (1 - d_{t_i})$ if the intruder were to attack t_i.

$$
\begin{aligned}
Maximize \quad & R && (1)\\
s.t. \quad & R \leq C - c_i \cdot (1 - d_{t_i}), \forall t_i \in T && (2)\\
& \sum_{e \in E} x_e \leq r && (3)\\
& d_s = 0, \forall s \in S && (4)\\
& d_v \leq d_u + x_e, \forall e = (u, v) \in E && (5)\\
& 0 \leq x_e \leq 1, \forall e \in E && (6)\\
& 0 \leq d_v \leq 1, \forall v \in V && (7)
\end{aligned}
$$

$R = \min_{t_i \in T}\{C - c_i \cdot (1 - d_{t_i})\}$ is the expected remaining value assuming the intruder always chooses the best source, target and path to maximize expected damage (Line **2**). The objective of the LP is to maximize R (Line **1**) subject to the following additional constraints:

- The probability of placing a checkpoint on any edge e, x_e, is between 0 and 1 (Line **6**).
- For any vertex v, the probability of capturing the intruder before he reaches v is between 0 and 1 (Line **7**).
- The probabilities of placing checkpoints on all edges sum up to at most r (Line **3**). This is because the maximum number of checkpoints we can place on the graph is r.
- For any source s, the probability of capturing the intruder before he reaches s is 0 (Line **4**). This follows naturally from the definition of a source as an entry point.
- From the intruder's perspective, for an edge e between u and v, the probability of being captured on his way to node v from a starting node is d_v which must be less than or equal to $d_u + x_e, \forall u$ (Line **7**). Here, d_u is the probability of being captured on the way from s to u, and x_e is the probability of being captured on edge e. The reason is obvious – if the intruder goes to u first and then goes to v from there through e, the probability of being captured is at most $d_u + x_e$. The optimal way of reaching v must be at least as good as this.

Theorem 5.1. *Let OPT be the optimal strategy and R_{OPT} be the expected remaining value, then $R_{LP} \geq R_{OPT}$, where R_{LP} is the expected remaining value returned by the LP.*

Proof. Let L be the set of distinct combinations of placing r checkpoints. The optimal mixed strategy OPT can be represented as a probability array $\{p_l\}$, where $l \in L$, $0 \leq p_l \leq 1$, and $\sum_{l \in L} p_l = 1$. For example, if we have 3 edges and 2 checkpoints, $L = \{\{1,2\},\{1,3\},\{2,3\}\}$, where $\{1,2\}$ means placing the checkpoints at edge 1 and 2. Let x'_e be the probability that at least one checkpoint is placed on e in OPT. x'_e can be computed by summing up the probabilities of all possible ways of placing checkpoints which will include e. We know $0 \leq x'_e \leq 1$ and $\sum_{e \in E} x'_e = \sum_{l \in L} |l| \cdot p_l \leq r$. For each vertex u, let d'_u be the probability of capturing the intruder assuming he chooses the best path to reach u. Then $0 \leq d'_u \leq 1$ and for each edge $e(u,v)$, $d'_v \leq d'_u + x'_e$. We can see d' and x' derived from the optimal strategy also satisfy the constraints in our LP. Therefore the optimal solution of the LP, R_{LP}, is at least as good as R_{OPT}. □

Notice that sometimes it is possible for the LP to return a solution with a payoff *even greater* than the optimal strategy. In such cases, the solutions returned by LP are unachievable because we cannot find mixed strategies $\{p_l\}$ to sample from that attain the d_v's and x_e's returned by the LP.

Given the set of x_e's found by the LP, it may still be nontrivial to generate an actual mixed strategy $\{p_l\}$ to sample correctly from even if one exists. In the worst case, it may require consideration of the defender's uncompressed, combinatorial action space. However, while optimality is nontrivial to achieve in general, we can easily find a good approximation by independently sampling for the placement of each checkpoint with a probability of x_e/r for each edge e. We show in Theorem 5.3 that even a strategy generated by a simple independent sampling approach guarantees a fraction of $1 - \left(1 - \frac{1}{r}\right)^r$ of the optimal strategy.

In terms of independent sampling, we refer to sampling r checkpoints separately and independently. In particular, for each checkpoint, edge e is chosen mutually exclusively with probability $\frac{x_e}{r}$.

Before we formally discuss Theorem 5.3, we first introduce the following lemma.

Lemma 5.2. *For any path P chosen by the intruder subject to $\sum_{e \in P} x_e \leq 1$, the probability of capturing the intruder is at least $[1 - (1 - \frac{1}{r})^r] \cdot \sum_{e \in P} x_e$.*

Proof. Let $p = \sum_{e \in P} x_e$. Since r checkpoints are placed independently and the probability that the i-th checkpoint is not on P is $1 - \sum_{e \in P} x_e/r = 1 - p/r$, the probability that there is no checkpoint on P is $(1-p/r)^r$. Thus the probability that there is at least one checkpoint on P is $1 - (1 - p/r)^r$. We only need to prove that for $0 < p \leq 1$, $1 - (1 - p/r)^r \geq [1 - (1 - \frac{1}{r})^r]p$. Let $f(p) = \frac{1-(1-p/r)^r}{p}$. We can prove $f(p)$ is monotonically decreasing on $(0, 1]$ by taking the derivative. Since $f'(p) \leq 0$ for any $0 < p \leq 1$, the minimum value of $f(p)$ on $(0, 1]$ is $f(1) = 1 - (1 - 1/r)^r$. □

Theorem 5.3. *Sampling r checkpoints independently guarantees a fraction of $1 - \left(1 - \frac{1}{r}\right)^r$ over the optimal strategy, i.e., $R_{IS} \geq \left[1 - \left(1 - \frac{1}{r}\right)^r\right] R_{OPT}$.*

Proof. Assume the defender solves the LP and adopts the independent sampling method. Fix an arbitrary path P, and write $p = \sum_{e \in P} x_e$. The probability of capturing the intruder on path P is $1 - \left(1 - \frac{p}{r}\right)^r$. Therefore, for any target t_i, the best path for the intruder to reach t_i is the 'shortest' path from s to t_i. Consider any target t_j: since $d_{t_j} \leq 1$, by Lemma 5.2, we know that the probability of capturing the intruder on the shortest path from s to t_j is at least $[1 - (1 - 1/r)^r] d_{t_j}$. Therefore the expected remaining value when t_j is attacked is,

$$\begin{aligned} R_{IS}(t_j) &\geq \sum_{i \neq j} c_i + \left[1 - \left(1 - \frac{1}{r}\right)^r\right] d_{t_j} c_j \\ &\geq \left[1 - \left(1 - \frac{1}{r}\right)^r\right] \left(\sum_{i \neq j} c_i + d_{t_j} c_j\right) \\ &= \left[1 - \left(1 - \frac{1}{r}\right)^r\right] \left[C - c_i \cdot (1 - d_{t_j})\right] \end{aligned}$$

Notice that $R_{IS} = \min_j \{R_{IS}(t_j)\}$ and $R_{LP} = \min_j \left\{C - c_i \cdot (1 - d_{t_j})\right\}$, we have that

$$R_{IS} \geq \left[1 - \left(1 - \frac{1}{r}\right)^r\right] R_{LP} \geq \left[1 - \left(1 - \frac{1}{r}\right)^r\right] R_{OPT}$$

In games with a single checkpoint, $r = 1$, $R_{IS} = R_{OPT}$, and the independent sampling method leads to an optimal solution in expectation. □

6 Experiments

This section demonstrates the performance of the algorithms in this paper. To that end, we evaluate the performance of each method on a set of problems mainly motivated by the SMS domain (described in Section 2) by comparing its running time and solution value (the expected defender's payoff) with the fastest-known exact algorithm for solving general Bayesian Stackelberg games, DOBSS. A related solution algorithm, ERASER [Kiekintveld *et al.*, 2009], exploits certain structural properties that exist in some security domains to create a compact game representation. While this technique proved useful in the Federal Air Marshal domain they considered, these properties do not exist in our domain due to the network structure of the situation in consideration, making direct comparison unreasonable.

Experiments were run on quad-core Intel 3.0GHz processors with 3GB of RAM. Each approach was run 100 times on each problem and we report the average time and expected value of the optimal strategy. All approaches are given a maximum time of 20 minutes. In this experiment, we do not alter the sparseness of the graph and consider the most complex type of graph (i.e., complete graph). The reward value on the targets are 1.0 if reward values are not explicitly given.

6.1 Comparison among MCTE, ESVVT, and DOBSS

Graph and Checkpoints Scale-up: Small graphs

This set of experiments shows that both MCTE and ESVVT significantly outperform DOBSS for even relatively small problems. We use small games so that DOBSS can solve the given problem within the cutoff time. Unless otherwise indicated, every experiment has one source, one sink, and two checkpoints. The number of vertices changes from 3 to 10 (3 to 45 edges). Figure 1a shows scaling of each method's runtime with respect to the size of the graph. The x-axis shows the number of vertices in the graph and the y-axis shows runtime in seconds. As shown in the figure, DOBSS is only able to solve the problem of up to 9 vertices within the time limit and requires nearly 10^4 times more time than MCTE and ESVVT. ESVVT shows the same reward as DOBSS, indicating that it also achieved an optimal solution.

The second set of experiments show how each algorithm performs when the number of checkpoints changes. All experiments have a single source, a single sink, and 7 vertices (21 edges) in the graph. Figure 1b shows the running time with various number of checkpoints (1 to 6). These experiments show increased number of checkpoints leads to higher runtime for DOBSS while MCTE and ESVVT keep roughly same running time regardless of the number of checkpoints. Specifically, DOBSS is able to solve the problem up through 5 checkpoints, but takes at least 3.9 times more time than MCTE and ESVVT, while all methods provide the same solution quality in terms of expected reward.

Graph and Checkpoints Scale-up: Large graphs

The third set of experiments show that ESVVT can efficiently scale-up to graphs having a very large number of vertices and edges. DOBSS cannot solve these problems within a reasonable amount of time due to time and memory constraints.

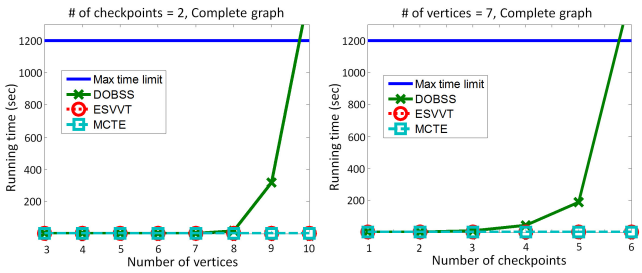

Figure 1: Scale-Up for a small problem.

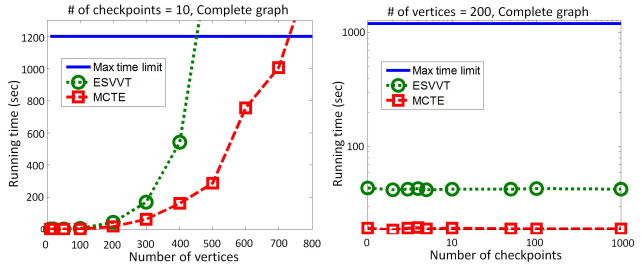

Figure 2: Scale-Up for a large problem.

Every experiment has a single source, a single sink, and 10 checkpoints. Figure 2a shows the running time accrued on problems with 10–800 vertices (45 to 319,600 edges). As shown in Figure 2, ESVVT is able to stay within 1.7–6.4 times the runtime of MCTE while finding solutions of the same quality.

The last set of experiments consider an increasing number of checkpoints from 1 to 1,000, shown in Figure 2b (x-axis & y-axis: log-scale). All experiments have a single source, a single target, and 200 vertices (19,900 edges). Unlike DOBSS, MCTE and ESVVT keep roughly the same running time (MCTE is about 2.2 times faster than ESVVT) as the number of checkpoints increases. As in the other cases, MCTE and ESVVT show the same expected reward, and, as expected, they are able to find better solutions as the number of checkpoints allowed increases.

6.2 The SMS Domain

Now we show the preliminary results for each algorithm on the SMS domain. The original SMS domain is very complex, with the majority of stations having very low values compared to other, high value stations. Thus, we use a simplified graph, keeping only the pivotal nodes such as those at the intersection of two lines, entry stations, and stations with medium or high value. Specifically, the graph shown in Figure 3 has 46 vertices and 62 edges, as compared to the 250+-station Seoul Metropolitan Subway system. Target values are given based on estimates of the amount of traffic passing through each station.

As shown in Table 2, DOBSS is only able to solve the simplest case within the 20-minute timeframe. DOBSS requires 10^4 times more time than MCTE, and 10^3 times more time than ESVVT, while they are able to find optimal solutions having the same expected reward. For the cases in which

# of entries/targets/checkpoints	DOBSS		MCTE		ESVVT	
	Time (sec)	Reward	Time (sec)	Reward	Time (sec)	Reward
1 / 1 (5/target, total:5) / 2	275.0	5.0	0.021	5.0	0.31	5.0
2 / 2 (5/target, total:10) / 3	-	-	0.024	8.0	0.318	8.0
2 / 4 (5/target, total:20) / 3	-	-	0.025	16.875	0.314	16.875
3 / 4 (5/target, total:20) / 3	-	-	0.026	16.50	0.336	16.50
3 / 6 (5/target, total:30) / 3	-	-	0.025	26.50	0.338	26.50
5 / 6 (5/target, total:30) / 3	-	-	0.027	26.25	0.447	26.25
5 / 6 (5-10/target, total:40) / 3	-	-	-	-	0.446	33.333
5 / 15 (1-10/target, total:75) / 3	-	-	-	-	0.444	67.727
5 / 15 (1-10/target, total:75) / 10	-	-	-	-	0.468	73.462
5 / 15 (1-10/target, total:75) / 15	-	-	-	-	0.356	75.0

Table 2: Comparison in the SMS domain

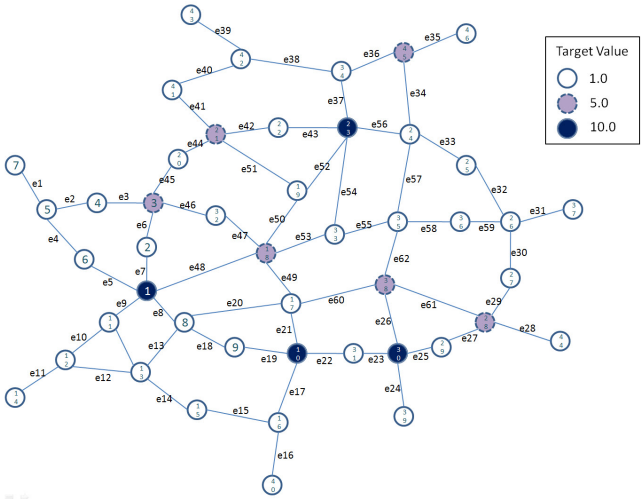

Figure 3: The simplified SMS graph.

multiple targets have equal values, MCTE and ESVVT can quickly find the same solution quality, with ESVVT requiring about 10 times more time than MCTE.

Finally, we illustrate ESVVT's applicability to more general problems with varying target values. In Table 2, the last 4 cases show the performance of ESVVT when the targets have different rewards. ESVVT's running time on problems with various configurations (even large number of vertices and checkpoints) remains less than one second.

6.3 Approximation of ESVVT

Although ESVVT has found the same reward value as MCTE and DOBSS in all the tests conducted thus far, it remains an approximate algorithm in the general case, as discussed in Section 5. Here we provide experimental verifications of this result. Consider the following example shown in Figure 4. There is a source s, and two targets t_1, t_2 of values 1 and 2. t_1 is connected to s with three parallel edges (e_1, e_2, e_3), t_2 to t_1 with one edge (e_4), and two checkpoints are allowed.

If we solve the problem using DOBSS, it returns a unique solution with an expected reward of 2.556, which can be achieved with the following strategy. Place checkpoints on

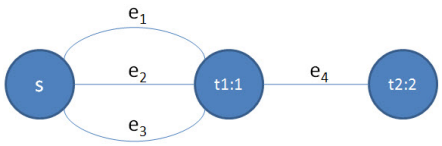

Figure 4: Graph with an unachievable ESVVT solution.

any pair of $\{e_1, e_2, e_3\}$ with probability $\frac{2}{9}$, and place checkpoints on e_4 and one of $\{e_1, e_2, e_3\}$ with probability $\frac{1}{9}$. However, the unique optimum solution to the ESVVT linear program assigns coverage probabilities of x_e = 0.6 to the three parallel edges, and x_e = 0.2 to the other edge and results in an expected reward of 2.60.

The reward of ESVVT is, in fact, an overestimate of the achievable expected reward due to its assumption that probabilities of sequential checkpoints can be added together. To illustrate this more clearly, suppose the intruder chooses a path from s to t_2. The actual probability of being caught is the probability that we do not choose the wrong parallel two edges. In the ESVVT solution, however, the linear program calculates it to be the sum of the probabilities of catching the intruder before he reaches t_1 and catching him on e_4: $0.6+0.2 = 0.8$. In the DOBSS solution, it is $1-\frac{2}{9} = \frac{7}{9} < 0.8$. Clearly, the DOBSS solution does not attain the coverage probability required by the ESVVT solution. However, since DOBSS' solution is a guaranteed optimal solution, this actually means that we cannot match the value from ESVVT with any actual distributions and it is simply an overestimate of the true optimal achievable expected reward.

To further investigate the practical performance of ESVVT, we generated 1,000 random graphs and compared the results found by DOBSS and ESVVT. Each graph has a random configuration of vertices ($|V| = 2 - 20$), edges ($|E| = 1 - 20$), checkpoints ($|R| = 1-20$), sources ($|S| = 1-(|V|-1)$, targets ($|T| = 1 - (|V| - |S|)$), and target values ($c_i = 1 - 10$). ESVVT found the same solution strategy as DOBSS did in 84.10% of the 1,000 trials conducted.

We have shown that MCTE and ESVVT scale better than DOBSS with respect to graph size and the number of checkpoints allowed. In complete graphs and the SMS graph,

ESVVT found the same expected reward as DOBSS and MCTE. In random graphs, ESVVT was able to find the optimal solutions for most cases. Although it lacks DOBSS' guarantee of optimality, ESVVT shows promising results in practice and is a polynomial time algorithm that is more generally applicable than MCTE.

7 Conclusions

Security in domains such as cities or subway networks where potential attackers must take paths from entry points to targets via defined roads is an area that has not seen much work. Although a large body of related work in Operations Research has examined the problem of capturing entities attempting to pass through the network, the techniques have not been translated into patrolling domains with targets of varying importance. Although [Washburn and Wood, 1995] provides techniques that are easily applicable to a simplified version of the problem with targets of equal value, the introduction of variation in importance between targets requires a fundamental shift away from minimum s-t cut algorithms.

While game theory offers a solid foundation from which to approach this class of problems with varying target values, methods that have been used previously for similar problems scale extremely poorly here. The general, exact solution method used in the LAX security domain, DOBSS, provides a benchmark for comparison, but quickly declines in applicability due to its inability to scale to problems of a reasonable size. The compact representation techniques used in the Federal Air Marshals Service domain [Kiekintveld *et al.*, 2009] scale exponentially with the size of the problem and are still only able to provide approximate solutions.

Therefore, we develop a novel linear program, ESVVT, to efficiently solve these problems. Although only approximate in the general case, we believe that the linear program performs well in real domains due to real-world restrictions on resources and graph structure. We hope to apply these techniques to more domains in the real world as natural extensions to this work and also look to develop new algorithms and sampling techniques that are able to solve this class of problems both efficiently and optimally.

References

[Adler *et al.*, 2003] Micah Adler, Harald Racke, Naveen Sivadasan, Christian Sohler, and Berthold Vocking. Randomized pursuit-evasion in graphs. *Combinatorics, Probability and Computing*, 12(03):225–244, 2003.

[Aspnes *et al.*, 2006] James Aspnes, Kevin Chang, and Aleksandr Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *Journal of Computer and System Sciences*, 72(6):1077 – 1093, 2006.

[Avenhaus *et al.*, 2002] Rudolf Avenhaus, Bernhard von Stengel, and Shmuel Zamir. Inspection games. In Robert J. Aumann and Sergui Hart, editors, *Handbook of Game Theory*, volume 3, chapter 51, pages 1947–1987. North-Holland, Amsterdam, 2002.

[Blanco *et al.*, 2007] Mikel Blanco, Aurelia Valino, Joost Heijs, Thomas Baumert, and Javier Gonzalez Gomez. The Economic Cost of March 11: Measuring the direct economic cost of the terrorist attack on March 11, 2004 in Madrid. *Terrorism and Political Violence*, 19(4):489–509, 2007.

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

[Cormican *et al.*, 1998] Kelly J. Cormican, David P. Morton, and R. Kevin Wood. Stochastic network interdiction. *Oper. Res.*, 46(2):184–197, 1998.

[Israeli and Wood, 2002] Eitan Israeli and R. Kevin Wood. Shortest-path network interdiction. *Networks*, 40:2002, 2002.

[Kearns *et al.*, 2001] Michael Kearns, Michael L. Littman, and Satinder Singh. Graphical models for game theory. In *In UAI*, pages 253–260, 2001.

[Kiekintveld *et al.*, 2009] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Milind Tambe, and Fernando Ordonez. Computing Optimal Randomized Resource Allocations for Massive Security Games. In *AAMAS-09*, 2009.

[Paruchuri *et al.*, 2008] Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *AAMAS-08*, pages 895–902, 2008.

[Pita *et al.*, 2008] James Pita, Manish Jain, Craig Western, Christopher Portway, Milind Tambe, Fernando Ordonez, Sarit Kraus, and Praveen Parachuri. Depoloyed ARMOR protection: The application of a game-theoretic model for security at the Los Angeles International Airport. In *AAMAS-08 (Industry Track)*, 2008.

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

[Tsai *et al.*, 2009] Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordonez, and Milind Tambe. IRIS A Tool fpr Strategic Security Allocation in Transportation Networks. In *AAMAS-09 (Industry Track)*, 2009.

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

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

[Washburn and Wood, 1995] Alan Washburn and Kevin Wood. Two-person zero-sum games for network interdiction. *Operations Research*, 43(2):243–251, 1995.

[Wood, 1993] R. Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modeling*, 17(2):1–18, 1993.