A Double Oracle Algorithm for Zero-Sum Security Games on Graphs

Manish Jain*, Dmytro Korzhyk†, Ondřej Vaněk+, Vincent Conitzer†,
Michal Pěchouček+, Milind Tambe*
* Computer Science Department, University of Southern California, Los Angeles, CA. 90089
{manish.jain,tambe}@usc.edu
† Department of Computer Science, Duke University, Durham, NC. 27708
{dima,conitzer}@cs.duke.edu
+ Department of Cybernetics, Czech Technical University, Prague. Czech Republic.
{vanek,pechoucek}@agents.felk.cvut.cz

ABSTRACT

In response to the Mumbai attacks of 2008, the Mumbai police have started to schedule a limited number of inspection checkpoints on the road network throughout the city. Algorithms for similar security-related scheduling problems have been proposed in recent literature, but security scheduling in networked domains when targets have varying importance remains an open problem at large. In this paper, we cast the network security problem as an attacker-defender zero-sum game. The strategy spaces for both players are exponentially large, so this requires the development of novel, scalable techniques.

We first show that existing algorithms for approximate solutions can be arbitrarily bad in general settings. We present RUGGED (*Randomization in Urban Graphs by Generating strategies for Enemy and Defender*), the first scalable optimal solution technique for such network security games. Our technique is based on a double oracle approach and thus does not require the enumeration of the entire strategy space for either of the players. It scales up to realistic problem sizes, as is shown by our evaluation of maps of southern Mumbai obtained from GIS data.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Security, Performance

Keywords

Game theory, Double oracle, Zero-sum games, Minimax equilibrium

1. INTRODUCTION

Securing urban city networks, transportation networks, computer networks and other critical infrastructure is a large and growing area of concern. The key challenge faced in these domains is to effectively schedule a *limited number of resources* to protect against an intelligent and adaptive attacker. For example, a police force has limited personnel to patrol, operate checkpoints, or conduct searches. The adversarial aspect poses significant challenges for the resource allocation problem. An intelligent attacker may observe the strategy of the defender, and then plan more effective attacks. Predictable scheduling of defender resources can be exploited by the attacker. *Randomization* has thus been used to keep attackers at bay by increasing the uncertainty they face.

Game theory offers a principled way of achieving effective randomization. It models the varying preferences of both the defender and the attacker, and allows us to solve for optimal strategies. Recent work has also used and deployed game-theoretic techniques in real-world attacker-defender scenarios, for example, ARMOR [13] and IRIS [10].

In this paper, we model an urban network security problem as a game with two players: the defender and the attacker. The pure strategies of the defender correspond to allocations of resources to edges in the network—for example, an allocation of police checkpoints to roads in the city. The pure strategies of the attacker correspond to paths from any *source* node to any *target* node—for example, a path from a landing spot on the coast to the airport.

The strategy space of the defender grows exponentially with the number of available resources, whereas the strategy space of the attacker grows exponentially with the size of the network. For example, in a fully connected graph with 20 nodes and 190 edges, the number of defender actions for only 5 resources is $\binom{190}{5} \approx 2$ billion, while the number of possible attacker paths without any cycles is $\approx 6.6^{18}$. Real-world networks are significantly larger, e.g., a simplified graph representing the road network in southern Mumbai has more than 250 nodes (intersections) and 600 edges (streets), and the security forces can deploy tens of resources.

We model the scenario as a zero-sum game, where the attacker gets a positive payoff in case of a successful attack and 0 otherwise, and the payoff to the defender is the negative of the attacker's payoff. Our goal is to find a minimax strategy for the defender, that is, a strategy that minimizes the maximum expected utility that the attacker can obtain.[1] The extremely large size of the games

Cite as: A Double Oracle Algorithm for Zero-Sum Security Games on Graphs, Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, Milind Tambe, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX-XXX.

[1]Because in this work, we assume the game to be zero-sum, a minimax strategy is equivalent to a Stackelberg strategy (where the defender finds the optimal mixed strategy to commit to); moreover, via von Neumann's minimax theorem [12] (or linear programming duality), minimax strategies also correspond exactly to Nash equi-

inhibits the direct application of standard methods for finding minimax strategies. Thus, we propose a double-oracle based approach that does not require the *ex-ante* enumeration of all pure strategies for either of the players. We propose algorithms for both the defender's and the attacker's oracle problems, which are used iteratively to provide pure-strategy best responses for both players. While we present NP-hardness proofs for the oracle problems for both players, the entire approach remains scalable in practice, as is shown in our experiments.

We also provide experimental results on real-city networks, specifically on graphs obtained from the GIS data of southern Mumbai. The graph representation of southern Mumbai has 250 nodes and 600 edges. The placement of sources and targets in the experiment was inspired by the Mumbai 2008 attacks where the targets were important economic and political centers and the sources were placed along the coast line. Our experimental results show that this problem remains extremely difficult to solve. While we show the previous approximation method to not be ready for deployment, our own techniques will need to be enhanced further for real deployments in the city of Mumbai. We believe the problem remains within reach, and is clearly an exciting and important area for continued research.

2. RELATED WORK

Game theory has been applied to a wide range of problems where one player — the evader — tries to minimize the probability of detection by and/or encounter with the other player — the patroller; the patroller wants to thwart the evader's plans by detecting and/or capturing him. The formalization of this problem led to a family of games, often called *pursuit-evasion games* [1]. As there are many potential applications of this general idea, more specialized game types have been introduced, e.g., *hider-seeker games* [7, 9] and *infiltration games* [2] with mobile patrollers and mobile evaders; *search games* [8] with mobile patrollers and immobile evaders; and *ambush games* [14] with the mobility capabilities reversed. In the game model proposed in this paper, the evader is mobile whereas the patroller is not, just like in ambush games. However, in contrast with ambush games, we consider *targets* (termed *destinations* in ambush games) of varying importance.

Our game model is most similar to that of *interdiction games* [17], where the evading player — the attacker — moves on an arbitrary graph from one of the origins to one of the destinations (aka. *targets*); and the interdicting player — the defender — inspects one or more edges in the graph in order to detect the attacker and prevent him from reaching the target. As opposed to interdiction games, we do not consider the detection probability on edges, but we allow different values to be assigned to the targets, which is crucial for real-world applications.

Recent work has also considered scheduling multiple-defender resources using cooperative game-theory, as in *path disruption games* [3], where the attacker tries to reach a single known target. In contrast with the static asset protection problem [6], we attribute different importance to individual targets and unlike its dynamic variant [6], we consider only static target positions. Recent work in security games and robotic patrolling [4, 10] has focused on concrete applications. However, they have not considered the scale-up for both defender and attacker strategies. For example, in ASPEN, the attacker's pure strategy space is polynomially large, since the attacker is not following any path and just chooses exactly one target to attack. Our game model was introduced by Tsai et al. [15]; however, their approximate solution technique can be suboptimal. We discuss the shortcomings of their approach in Section 4, and provide an optimal solution algorithm for the general case.

Techniques used by RUGGED are based on a double oracle approach, as proposed by McMahan et al. [11] (corresponding exactly to the notion of constraint and column generation in linear programming). This technique is intended to solve large-scale games, and is especially useful in settings where efficient algorithms for the best-response oracle problems are available. Double oracle algorithms have subsequently been applied to various pursuit-evasion games [9, 16]. While the best-response oracle problems are NP-hard in our setting (as we show in Sections 5.3 and 5.4), we give algorithms for these problems that allow the approach to still scale to realistic instances.

librium strategies. For a discussion of the relationships among these concepts in security games, which include zero-sum games, see Yin et al. [18].

3. PROBLEM DESCRIPTION

A network security domain, as introduced by Tsai et al. [15], is modeled using a graph $G = (N, E)$. The attacker starts at one of the source nodes $s \in S \subset N$ and travels along a path of his choosing to any one of the targets $t \in T \subset N$. The attacker's pure strategies are thus all the possible $s-t$ paths from any source $s \in S$ to any target $t \in T$. The defender tries to catch the attacker before he reaches any of the targets by placing k available (homogeneous) resources on edges in the graph. The defender's pure strategies are thus all the possible allocations of k resources to edges, so there are $\binom{|E|}{k}$ in total. Assuming the defender plays allocation $X_i \subseteq E$, and the attacker chooses path $A_j \subseteq E$, the attacker succeeds if and only if $X_i \cap A_j = \emptyset$. Additionally, a payoff $\mathcal{T}(t)$ is associated with each target t, such that the attacker gets $\mathcal{T}(t)$ for a successful attack on t and 0 otherwise. The defender receives $-\mathcal{T}(t)$ in case of a successful attack on t and 0 otherwise. The network security domain is modeled as a complete-information zero-sum game, where the set S of sources, T of targets, the payoffs $\mathcal{T}$ for all the targets and the number of defender resources k are known to both the players *a-priori*. The objective is to find the mixed strategy $\mathbf{x}$ of the defender, corresponding to a Nash equilibrium (equivalently, a minimax strategy) of this *network security game*. The notation used in the paper is described in Table 1.

$G(\mathbf{N}, \mathbf{E})$	Urban network graph
$\mathcal{T}$	Target payoff
k	Defender resources
$\mathbf{X}$	Set of defender allocations, $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$
X_i	i^{th} defender allocation. $X_i = \{X_{ie}\} \forall e, X_{ie} \in \{0, 1\}$
$\mathbf{A}$	Set of attacker paths, $\mathbf{A} = \{A_1, A_2, \ldots, A_m\}$
A_j	j^{th} attacker path. $A_j = \{A_{je}\} \forall e, A_{je} \in \{0, 1\}$
$\mathbf{x}$	Defender's mixed strategy over $\mathbf{X}$
$\mathbf{a}$	Adversary's mixed strategy over $\mathbf{A}$
$U_d(\mathbf{x}, A_j)$	Defender's expected utility playing $\mathbf{x}$ against A_j
Λ	Defender's pure strategy best response
Γ	Attacker's pure strategy best response

Table 1: Notation

4. RANGER COUNTEREXAMPLE

RANGER was introduced by Tsai et al. [15] and was designed to obtain approximate solutions for the defender for the network security game. Its main component is a polynomial-sized linear program that, rather than solving for a distribution over allocations, solves for the *marginal* probability with which the defender covers

each edge. It does this by approximating the capture probability as the sum of the marginals along the attacker's path. It further presents some sampling techniques to obtain a distribution over defender allocations from these marginals. What was known before was that the RANGER solution (regardless of the sampling method used) is suboptimal in general, because it is not always possible to find a distribution over allocations such that the capture probability is indeed the sum of marginals on the path. In this paper, we show that RANGER's error can be arbitrarily large.

Let us consider the example graph shown in Figure 1. This multi-graph[2] has a single source node, s, and two targets, t_1 and t_2; the defender has 2 resources. Furthermore, the payoffs $\mathcal{T}$ of the targets are defined to be 1 and 2 for targets t_1 and t_2 respectively.

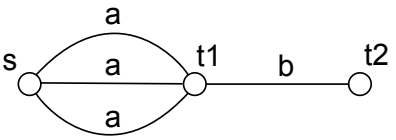

Figure 1: This example is solved *incorrectly* by RANGER. The variables a, b are the coverage probabilities on the corresponding edges.

RANGER *solution:*

Suppose RANGER puts marginal coverage probability a on each of the three edges between s and t_1,[3] and probability b on the edge between t_1 and t_2, as shown in Figure 1. RANGER estimates that the attacker gets caught with probability a when attacking target t_1 and probability $a + b$ when attacking target t_2. RANGER will attempt to make the attacker indifferent between the two targets to obtain the minimax equilibrium. Thus, RANGER's output is $a = 3/5, b = 1/5$, obtained from the following system of equations:

$$1(1 - a) = 2(1 - (a + b)) \quad (1)$$

$$3a + b = 2 \quad (2)$$

However, there can be no allocation of 2 resources to the edges such that the probability of the attacker being caught on his way to t_1 is $3/5$ and the probability of the attacker being caught on his way to t_2 is $4/5$. (The reason is that in this example, the event of there being a defensive resource on the second edge in the path cannot be disjoint from the event of there being one on the first edge.) In fact, for this RANGER solution, the attacker cannot be caught with a probability of more than $3/5$ when attacking target t_2, and so the defender utility cannot be greater than $-2(1 - 3/5) = -4/5$.

Optimal solution:

Figure 2 shows the six possible allocations of the defender's two resources to the four edges. Three of them block some pair of edges between s and t_1. Suppose that each of these three allocations is played by the defender with probability x.[4] Each of the other three allocations blocks one edge between s and t_1 as well as the edge between t_1 and t_2. Suppose the defender chooses these allocations with probability y each (refer Figure 2). The probability of the attacker being caught on his way to t_1 is $\frac{2}{3}3x + \frac{1}{3}3y$, or $2x + y$. Similarly, the probability of the attacker being caught on his way to t_2 is $2x + 3y$. Thus, a minimax strategy for this problem is the solution of Equations (3) and (4), which make the attacker indifferent between targets t_1 and t_2.

[2]We use a multi-graph for simplicity. This counterexample can easily be converted into a similar counterexample that has no more than one edge between any pair of nodes in the graph.

[3]We can assume without loss of solution quality that symmetric edges will have equal coverage.

[4]Again, this can be assumed without loss of generality for symmetric edges.

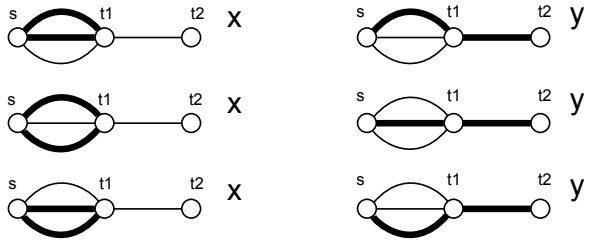

Figure 2: The possible allocations of two resources to the four edges. The blocked edges are shown in bold. The probabilities (x or y) are shown next to each allocation.

$$1(1 - 2x - y) = 2(1 - 2x - 3y) \quad (3)$$

$$3x + 3y = 1 \quad (4)$$

The solution to the above system is $x = 2/9$, $y = 1/9$, so that the expected attacker utility is $4/9$. Thus, the expected defender utility is $-4/9$, which is higher than the expected defender utility of at most $-4/5$ resulting from using RANGER.

RANGER *sub-optimality:*

Suppose the payoff $\mathcal{T}(t_2)$ of target t_2 in the example above was $H, H > 1$. The RANGER solution in this case, again obtained using Equations 1 and 2, would be $a = \frac{(H+1)}{(2H+1)}, b = \frac{(H-1)}{(2H+1)}$.

Then, consider an attacker who attacks the target t_2 by first going through one of the three edges from s to t_1 uniformly at random (and then on to t_2). The attacker will fail to be caught on the way from t_1 to t_2 with probability $(1 - b)$, given that the defender's strategy is consistent with the output of RANGER. Even conditional on this failure, the attacker will fail to be caught on the way from s to t_1 with probability at least $1/3$, because the defender has only 2 resources. Thus, the probability of a successful attack on t_2 is at least $(1 - b)(1/3)$, and the attacker's best-response utility is at least:

$$\frac{H(1-b)}{3} = \frac{H(H+2)}{3(2H+1)} > \frac{H(H+0.5)}{3(2H+1)} = \frac{H}{6} \quad (5)$$

Thus, the true defender utility for any strategy consistent with RANGER is at most $-\frac{H}{6}$.

Now, consider another defender strategy in which the defender always blocks the edge from t_1 to t_2, and also blocks one of the three edges between s and t_1 uniformly at random. For such a defender strategy, the attacker can reach t_1 with probability $2/3$, but cannot reach target t_2 at all. Thus, the attacker's best-response utility in this case is $2/3$. Therefore, the optimal defender utility is at least $-2/3$. Therefore, any solution consistent with RANGER is at least $\frac{H}{6}/\frac{2}{3} = \frac{H}{4}$ suboptimal. Since H is arbitrary, RANGER solutions can be arbitrarily suboptimal. This motivates our exact, double-oracle algorithm, RUGGED.

5. DOUBLE-ORACLE APPROACH

In this section, we present RUGGED, a double-oracle based algorithm for network security games. We also analyze the computational complexity of determining best responses for both the defender and the attacker, and, to complete the RUGGED algorithm, we give algorithms for computing the best responses.

5.1 Algorithm

The algorithm RUGGED is presented as Algorithm 1. $\mathbf{X}$ is the set of defender allocations generated so far, while $\mathbf{A}$ is the set of attacker paths generated so far. *CoreLP*$(\mathbf{X}, \mathbf{A})$ finds an equilibrium (and hence, minimax and maximin strategies) of the two-player zero-sum game consisting of the sets of pure strategies, $\mathbf{X}$ and $\mathbf{A}$, generated so far. *CoreLP* returns $\mathbf{x}$ and $\mathbf{a}$, which are the current equilibrium mixed strategies for the defender and the attacker over $\mathbf{X}$ and $\mathbf{A}$ respectively. The *defender oracle* (DO) generates a defender allocation Λ that is a best response for the defender against $\mathbf{a}$. (This is a best response among *all* allocations, not just those in $\mathbf{X}$.) Similarly, the *attacker oracle* (AO) generates an attacker path Γ that is a best response for the attacker against $\mathbf{x}$.

Algorithm 1 Double Oracle for Urban Network Security

1. Initialize $\mathbf{X}$ by generating arbitrary candidate defender allocations.
2. Initialize $\mathbf{A}$ by generating arbitrary candidate attacker paths.
repeat
3. $(\mathbf{x}, \mathbf{a}) \leftarrow$*CoreLP*$(\mathbf{X}, \mathbf{A})$.
4a. $\Lambda \leftarrow DO(\mathbf{a})$.
4b. $\mathbf{X} \leftarrow \mathbf{X} \cup \{\Lambda\}$.
5a. $\Gamma \leftarrow AO(\mathbf{x})$.
5b. $\mathbf{A} \leftarrow \mathbf{A} \cup \{\Gamma\}$.
until convergence
7. Return $(\mathbf{x}, \mathbf{a})$

The double oracle algorithm thus starts with a small set of pure strategies for each player, and then grows these sets in every iteration by applying the best-response oracles to the current solution. Execution continues until convergence is detected. Convergence is achieved when the best-response oracles of both the defender and the attacker do not generate a pure strategy that is better for that player than the player's strategy in the current solution (holding the other player's strategy fixed). In other words, convergence is obtained if, for both players, the reward given by the best-response oracle is no better than the reward for the same player given by the *CoreLP*.

The correctness of best-response-based double oracle algorithms for two-player zero-sum games has been established by McMahan et al [11]; the intuition for this correctness is as follows. Once the algorithm converges, the current solution must be an equilibrium of the game, because each player's current strategy is a best response to the other player's current strategy—this follows from the fact that the best-response oracle, which searches over all possible strategies, cannot find anything better. Furthermore, the algorithm must converge, because at worst, it will generate all pure strategies.

5.2 CoreLP

The purpose of *CoreLP* is to find an equilibrium of the restricted game consisting of defender pure strategies $\mathbf{X}$ and attacker pure strategies $\mathbf{A}$. Below is the standard formulation for computing a maximin strategy for the defender in a two-player zero-sum game.

$$\max_{U_d^*, \mathbf{x}} \quad U_d^* \tag{6}$$

$$\text{s.t.} \quad U_d^* \leq U_d(\mathbf{x}, A_j) \quad \forall j = 1, \ldots, |\mathbf{A}| \tag{7}$$

$$\mathbf{1}^{\mathrm{T}}\mathbf{x} = 1 \tag{8}$$

$$\mathbf{x} \in [0,1]^{|X|} \tag{9}$$

The defender's mixed strategy $\mathbf{x}$, defined over $\mathbf{X}$, and utility U_d^* are the variables for this problem. Inequality (7) is family of constraints; there is one constraint for every attacker path A_j in $\mathbf{A}$. The function $U_d(\mathbf{x}, A_j)$ is the expected utility of the attacker path A_j. Given A_j, the probability that the attacker is caught is the sum of the probabilities of the defender allocations that would catch the attacker. (We can sum these probabilities because they correspond to disjoint events.) More precisely, let z_{ij} be an indicator for whether allocation X_i intersects with path A_j, that is,

$$z_{ij} = \begin{cases} 1 & \text{if } X_i \cap A_j \neq \varnothing \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

These z_{ij} are not variables of the linear program; they are parameters that are determined at the time the best responses are generated. Then, the probability that an attacker playing path A_j is caught is $\sum_i z_{ij} x_i$, and the probability that he is *not* caught is $\sum_i (1 - z_{ij}) x_i$. Thus, the payoff function $U_d(\mathbf{x}, A_j)$ for the defender for choosing a mixed strategy $\mathbf{x}$ when the attacker chooses path A_j is given by Equation (11), where $\mathcal{T}(t_j)$ is the attacker's payoff for reaching t_j.

$$U_d(\mathbf{x}, A_j) = -\mathcal{T}(t_j) \cdot (\sum_i (1 - z_{ij}) x_i) \tag{11}$$

The dual variables corresponding to Inequality (7) give the attacker's mixed strategy $\mathbf{a}$, defined over $\mathbf{A}$. The expected utility for the attacker is given by $-U_d^*$.

5.3 Defender Oracle

This section concerns the best-response oracle problem for the defender. The *Defender Oracle* problem is stated as follows: generate the defender pure strategy (resource allocation) Λ allocating k resources over the edges E that maximizes the defender's expected utility against a given attacker mixed strategy $\mathbf{a}$ over paths $\mathbf{A}$.

Defender Oracle problem is NP-hard: We show this by reducing the set cover problem to it. **The Set-Cover problem:** Given are a set U, a collection $\mathcal{S}$ of subsets of U (that is, $\mathcal{S} \subseteq 2^U$), and an integer k. The question is whether there is a cover $\mathcal{C} \subseteq \mathcal{S}$ of size k or less, that is, $\bigcup_{c \in \mathcal{C}} c = U$ and $|\mathcal{C}| \leq k$. We will use a modification of this well-known NP-hard problem so that $\mathcal{S}$ always contains all singleton subsets of U, that is, $x \in U$ implies $\{x\} \in \mathcal{S}$. This modified problem remains NP-hard.

THEOREM 1. *The Defender Oracle problem is NP-hard, even if there is only a single source and a single target.*

PROOF. **Reduction from Set-Cover to *Defender Oracle*:** We convert an arbitrary instance of the set cover problem to an instance of the defender oracle problem by constructing a graph G with just 3 nodes, as shown in Figure 3. The graph G is a multi-graph[5] with just three nodes, so that $\mathbf{N} = \{s, v, t\}$, where s is the only source and t is the only target (with arbitrary positive value). There are up to $|\mathcal{S}|$ loop edges adjacent to node v; each loop edge corresponds to a unique non-singleton subset in $\mathcal{S}$. There are $|U|$ edges between s and v, each corresponding to a unique element in U. There are also $|U|$ edges between v and t, each corresponding to a unique element in U. The attacker's paths correspond to the elements in U. A path that corresponds to $u \in U$ starts with the edge between s and v that corresponds to u, then loops through all the edges that correspond to non-singleton subsets in $\mathcal{S}$ that contain u, and finally ends with the edge between v and t that corresponds to u. Hence, any two paths used by the attacker can only intersect at the loop edges. The probabilities that the defender places on these paths are arbitrary positive numbers. We now show that set U can be covered with k subsets in $\mathcal{S} \subseteq 2^U$ *if and only if* the defender can block all of the attacker's paths with k resources in the corresponding defender oracle problem instance.

[5]Having a multi-graph is not essential to the NP-hardness reduction.

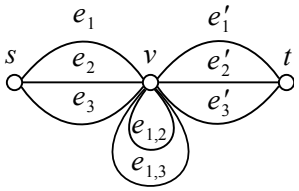

Figure 3: A *defender oracle* problem instance corresponding to the SET-COVER instance with $U = \{1,2,3\}$, $\mathcal{S} = \{\{1\},\{2\},\{3\},\{1,2\},\{1,3\}\}$. Here, the attacker's mixed strategy uses three paths: $(e_1, e_{1,2}, e_{1,3}, e'_1)$, $(e_2, e_{1,2}, e'_2)$, $(e_3, e_{1,3}, e'_3)$. Thus, the SET-COVER instance has a solution of size 2 (for example, using $\{1,2\}$ and $\{1,3\}$); correspondingly, with 2 resources, the defender can always capture the attacker (for example, by covering $e_{1,2}, e_{1,3}$).

The "if" direction: If the defender can block all the paths used by the attacker with k resources, then the set U can be covered with $\mathcal{C} \subseteq \mathcal{S}$, where $|\mathcal{C}| = k$ and is constructed as follows. If the defender places a resource on a loop edge, then $\mathcal{C}$ includes the non-singleton subset in $\mathcal{S}$ that corresponds to that loop edge. If the defender blocks any other edge then $\mathcal{C}$ includes the corresponding singleton subset.

The "only if" direction: If there exists a cover $\mathcal{C}$ of size k, then the defender can block all the paths by placing a defensive resource on every loop edge that corresponds to a non-singleton subset in $\mathcal{C}$, and placing a defensive resource on the corresponding edge out of s for every singleton subset in $\mathcal{C}$. □

Formulation: The defender oracle problem, described below, can be formulated as a mixed integer linear program (*MILP*). The objective of the MILP is to identify the allocation that *covers* as many attacker paths as possible, where paths are weighted by the product of the payoff of the target attacked by the path and probability of attacker choosing it. (In this formulation, probabilities a_j are not variables; they are provided by *CoreLP*.) In the formulation, $\lambda_e = 1$ indicates that we assign a resource to edge e, and $z_j = 1$ indicates that path A_j (refer Table 1) is blocked by the allocation.

$$\max_{z,\lambda} \quad -\textstyle\sum_j (1 - z_j) a_j \mathcal{T}_{t_j} \tag{12}$$

$$\text{s.t.} \quad z_j \leq \sum_e A_{je} \lambda_e \tag{13}$$

$$\textstyle\sum_e \lambda_e \leq k \tag{14}$$

$$\lambda_e \in \{0,1\} \tag{15}$$

$$z_j \in [0,1] \tag{16}$$

THEOREM 2. *The MILP described above correctly computes a best-response allocation for the defender.*

PROOF. The defender receives a payoff of $-\mathcal{T}(t_j)a_j$ if the attacker successfully attacks target t_j using path A_j, and 0 in the case of an unsuccessful attack. Hence, if we make sure that $1 - z_j = 1$ if path A_j is not blocked, and 0 otherwise, then the objective function (12) correctly models the defender's expected utility. Inequality (13) ensures this: its right-hand side will be at least 1 if there exists an edge on the path A_j that defender is covering, and 0 otherwise. z_j need not be restricted to take an integer value because the objective is increasing with z_j and if the solver can push it above 0, it will choose to push it all the way up to 1. Therefore, if we let Λ correspond to the set of edges covered by the defender, z_j will be set by the solver so that:

$$z_j = \begin{cases} 1 & \text{if } \Lambda \cap A_j \neq \varnothing \Leftrightarrow \exists e \,|\lambda_e = A_{je} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Inequality (14) enforces that the defender covers at most as many edges as the number of available resources k, and thus ensures feasibility. Hence, the above MILP correctly captures the best-response oracle problem for the defender. □

PROPOSITION 1. *For any attacker mixed strategy, the defender's expected utility from the best response provided by the defender oracle is no worse than the defender's equilibrium utility in the full zero-sum game.*

PROOF. In any equilibrium, the attacker plays a mixed strategy that minimizes the defender's best-response utility; therefore, if the attacker plays any other mixed strategy, the defender's best-response utility can be no worse. □

5.4 Attacker Oracle

This section concerns the best-response oracle problem for the attacker. The ***Attacker Oracle*** problem is to generate the attacker pure strategy (path) Γ from some source $s \in S$ to some target $t \in T$ that maximizes the attacker expected utility given the defender mixed strategy $\mathbf{x}$ over defender allocations $\mathbf{X}$.

Attacker Oracle is NP-hard: We show that the attacker oracle problem is also NP-hard by reducing 3-SAT to it.

THEOREM 3. *The Attacker Oracle problem is NP-hard, even if there is only a single source and a single target.*

PROOF. **Reduction from 3-SAT to *Attacker Oracle*:** We convert an arbitrary instance of 3-SAT to an instance of the *attacker oracle* problem as follows. Suppose the 3-SAT instance contains n variables x_i, $i = 1,\ldots,n$, and k clauses. Each clause is a disjunction of three literals, where each *literal* is either a variable or the negation of the variable. Consider the following example:

$$E = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4) \tag{18}$$

The formula E contains $n = 4$ variables and $k = 2$ clauses.

We construct a multi-graph G[6] with $n+k+1$ nodes, $v_0,\ldots,v_{n+k}$ so that the source node is $s = v_0$, and the target node is $t = v_{n+k}$. Every edge connects some pair of nodes with consecutive indices, so that every simple path from s to t contains exactly $n + k$ edges. Each edge corresponds to a literal in the 3-SAT expression (that is, either x_i or $\neg x_i$). There are exactly three edges that connect nodes v_{i-1} and v_i for $i = 1,\ldots,k$. Those three edges correspond to the three literals in the i-th clause. There are exactly two edges that connect nodes v_{k+j-1} and v_{k+j} for $j = 1,\ldots,n$. Those two edges correspond to literals x_j and $\neg x_j$. An example graph that corresponds to the expression (18) is shown in Figure 4.

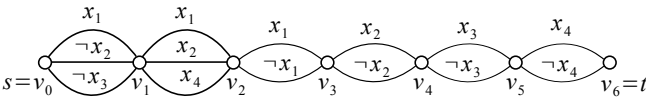

Figure 4: An example graph corresponding to the CNF formula $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4)$

There are $2n$ defender pure strategies (allocations of resources), each played with equal probability of $1/(2n)$. Each defender pure strategy corresponds to a literal, and the edges that correspond to that literal are blocked in that pure strategy. In the example shown in Figure 4, the defender plays 8 pure strategies, each with probability $1/8$. Three edges are blocked in the pure strategy that corresponds to the literal x_1 (namely, the top edge between v_0 and v_1,

[6]We use a multi-graph for simplicity; having a multi-graph is not essential for the NP-hardness reduction.

the top edge between v_1 and v_2, and the top edge between v_2 and v_3); only one edge is blocked in the pure strategy that corresponds to the literal $\neg x_4$ (the bottom edge between v_5 and v_6). (If it is desired that the defender always use the same number of resources, this is easily achieved by adding dummy edges.) We now show that there is an assignment of values to the variables in the 3-SAT instance so that the formula evaluates to *true if and only if* there is a path from s to t in the corresponding attacker oracle problem instance which is blocked with probability at most $1/2$.

The "if" direction: Suppose there is a path Γ from s to t that is blocked with probability at most $1/2$. Note that any path from s to t is blocked by at least one of the strategies $\{x_i, \neg x_i\}$, for all $i = 1, \ldots, n$, so the probability that the path is blocked is at least $n/(2n) = 1/2$. Moreover, if for some i, the path passes through both an edge labeled x_i and one labeled $\neg x_i$, then the probability that the path is blocked is at least $(n+1)/(2n) > 1/2$—so this cannot be the case for Γ. Hence, we can assign the *true* value to the literals that correspond to the edges on the path Γ, and *false* to all the other literals. This must correspond to a solution to the 3-SAT instance, because each clause must contain a literal that corresponds to an edge on the path, and is thus assigned a *true* value.

The "only if" direction: Suppose there is an assignment of values to the variables such that the 3-SAT formula evaluates to *true*. Consider a simple path Γ that goes from s to t through edges that correspond to literals with *true* values in the assignment. Such a path must exist because by assumption the assignment satisfies every clause. Moreover, this path is blocked only by the defender strategies that correspond to *true* literals, of which there are exactly n. So the probability that the path is blocked is $n/(2n) = 1/2$. □

Formulation: The attacker oracle problem can be formulated as a set of mixed integer linear programs, as described below. For every target in T, we solve for the best path to that target; then we take the best solution overall. Below is the formulation when the attacker is attacking target t_m. (In this formulation, probabilities x_i are not variables; they are values produced earlier by *CoreLP*.) In the formulation, $\gamma_e = 1$ indicates that the attacker passes through edge e, and $z_i = 1$ indicates that the allocation X_i blocks the attacker path. Equations (20) to (22) represent the flow constraints for the attacker for every node $n \in \mathbf{N}$.

$$\max_{z,\gamma} \quad \mathcal{T}_{t_m} \textstyle\sum_i x_i(1 - z_i) \tag{19}$$

$$\text{s.t.} \quad \textstyle\sum_{e \in \text{out}(n)} \gamma_e = \sum_{e \in \text{in}(n)} \gamma_e \quad n \neq s, t_m \tag{20}$$

$$\textstyle\sum_{e \in \text{out}(s)} \gamma_e = 1 \tag{21}$$

$$\textstyle\sum_{e \in \text{in}(t_m)} \gamma_e = 1 \tag{22}$$

$$z_i \geq \gamma_e + X_{ie} - 1 \quad \forall e \forall i \tag{23}$$

$$z_i \geq 0 \tag{24}$$

$$\gamma_e \in \{0, 1\} \tag{25}$$

THEOREM 4. *The MILP described above correctly computes a best-response path for the attacker.*

PROOF. The flow constrains are represented in Equations (20) to (22). The sink for the flow is the target t_m that we are currently considering for attack. To deal with the case where there is more than one possible source node, we can add a virtual source (s) to G that feeds into all the real sources. $\texttt{in}(n)$ represents the edges coming into n, $\texttt{out}(n)$ represents those going out of n. The flow constraints ensure that the chosen edges indeed constitute a path from the (virtual) source to the sink.

The attacker receives a payoff of $\mathcal{T}(t_m)$ if he attacks target t_m successfully, that is, if the path does not intersect with any defender allocation. Hence, if we make sure that $1 - z_i = 1$ if allocation X_i does not block the path, and 0 otherwise, then the objective function (19) correctly models the attacker's expected utility. Inequality (23) ensures this: if the allocation X_i covers some e for which $\gamma_e = 1$, then it will force z_i to be set at least to 1; otherwise, z_i only needs to be set to at least 0 (and in each case, the solver will push it all the way down to this value, which also explains why the z_i variables do not need to be restricted to take integer values). Therefore, if we let Γ correspond to the path chosen by the attacker, z_i will be set by the solver so that

$$z_i = \begin{cases} 1 & \text{if } X_i \cap \Gamma \neq \varnothing \Leftrightarrow \exists e \,|\gamma_e = X_{ie} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

It follows that the MILP objective is correct. Hence, the above MILP captures the best-response oracle problem for the attacker. □

PROPOSITION 2. *For any defender mixed strategy, the attacker's expected utility from the best response provided by the attacker oracle is no worse than the attacker's equilibrium utility in the full zero-sum game.*

PROOF. In any equilibrium, the defender plays a mixed strategy that minimizes the attacker's best-response utility; therefore, if the defender plays any other mixed strategy, the attacker's best-response utility can be no worse. □

Figure 5: Example graph of Southern Mumbai with 455 nodes. Sources are depicted as green arrows and targets are red bullseyes. Best viewed in color.

6. EVALUATION

In this section, we describe the results we achieved with RUGGED. We conducted experiments on graphs obtained from road network GIS data for the city of Mumbai (inspired by the 2008 Mumbai incidents [5]), as well as on artificially generated graphs. We provide two types of results: (1) Firstly, we compare the solution quality obtained from RUGGED with the solution quality obtained from RANGER. These results are shown in Section 6.1. (2) Secondly, we provide runtime results showing the performance of RUGGED when the input graphs are scaled up.[7] The following three types of graphs were used for the experimental results:

(1) *Weakly fully connected (WFC)* graphs, denoted $G^{\text{WFC}}(N, E)$, are graphs where N is an ordered set of nodes $\{n_1, \ldots, n_m; S = \{n_1\}, T = \{n_m\}\}$. For each node n_i, there exists a set of directed

[7]All experiments were run on standard desktop 2.8GHz machine with 2GB main memory.

edges, $\{(n_i, n_j)|n_i < n_j\}$, in E. These graphs were chosen because of the extreme size of the strategy spaces for both players. Additionally, there are no *bottleneck* edges, so these graphs are designed to be computationally challenging for RUGGED.

(2) *Braid*-type graphs, denoted $G^B(N, E)$, are graphs where N is a sequence of nodes n_1 to n_m such that each pair n_{i-1} and n_i is connected by 2 to 3 edges. Node n_1 is the source node. Any following node is a target node with probability 0.2, with payoff $\mathcal{T}$ randomly chosen between 1 and 100. These graphs have a similar structure as the graph in Figure 1, and were motivated by the counterexample in Section 4.

(3) *City* graphs of different sizes were extracted from the southern part of Mumbai using the GIS data provided by OpenStreetMaps. The placement of 2-4 targets was inspired by the Mumbai incidents from 2008 [5]; 2-4 sources were placed on the border of the graph,[8] simulating an attacker approaching from the sea. We ran the test for graphs with the following numbers of nodes: 45, 129 and 252. Figure 5 shows a sample Mumbai graph with 252 nodes, 4 sources and 3 targets.

6.1 Comparison with RANGER

This section compares the solution quality of RUGGED and RANGER. Although we have already established that RANGER solutions can be arbitrarily bad in general, the objective of these tests is to compare the actual performance of RANGER with RUGGED. The results are given in Table 2, which shows the average and maximum *error* from RANGER. We evaluated RANGER on the three types of graphs — city graphs, braid graphs and weakly fully connected graphs of different sizes, fixing the number of defender resources to 2 and placing 3 targets, with varied values from the interval $[0, 1000]$. The actual defender utility from the solution provided by RANGER[9] is computed by using the best-response oracle for the attacker with the RANGER defender strategy as input. The error of RANGER is then expressed as the difference between the defender utilities in the solutions provided by RANGER and by RUGGED.

Table 2 shows the comparison results between RANGER and RUGGED, summarized over 30 trials. It shows the percentage of trials in which RANGER gave an incorrect solution (denoted "pct"). It also shows the average and maximum error of RANGER (denoted as *avg* and *max* respectively) over these trials. It shows that while RANGER was wrong only about $1/3$ of the time for Braid graphs, it gave the wrong answer in *all* the runs on the fully connected graphs. Furthermore, it was wrong 90% of the time on city graphs, with an average error of 215 units and a maximum error of 721 units. Given an average target value of 500, these are high errors indeed — indicating that RANGER is unsuitable for deployment in real-world domains.

6.2 Scale-up and analysis

This section concerns the performance of RUGGED when the input problem instances are scaled up. The experiments were conducted on graphs derived directly from portions of Mumbai's road network. The runtime results are shown in Table 3, where the rows represent the size of the graph and the columns represent the number of defender resources that need to be scheduled. As an example of the complexity of the graph, the number of attacker paths in the Mumbai graph with 252 nodes is at least a 10^{12}, while the number of defender allocations is approximately 10^{10} for 4 resources.

[8]We placed more sources and targets into larger graphs.

[9]Because RANGER provides a solution in the form of marginal probabilities of defender allocations along edges, we used *Comb sampling* [15] to convert this into a (joint) probability distribution over defender allocations.

	City		Braid		WFC	
nodes	45	129	10	20	10	20
avg error	215	250	210	259	191	80
max error	721	489	472	599	273	117
pct	90%	100%	30%	37%	100%	100%
avg $\mathcal{T}$	500	500	500	500	500	500

Table 2: RANGER average and maximum error and percent of samples where RANGER provided a suboptimal solution. Target values $\mathcal{T}$ were randomly drawn from the interval $[1, 1000]$.

	1	2	3	4
45	0.91	6.43	22.58	33.42
129	6.63	32.55	486.48	3140.23
252	17.19	626.25	2014.14	34344.70

Table 3: Runtime (in seconds) of RUGGED when the input problem instances are scaled up. These tests were done on graphs extracted from the road network of Mumbai. The rows correspond to the number of nodes in the graph whereas the columns correspond to the number of defender resources.

The game matrix for this problem cannot even be represented, let alone solved. The ability of RUGGED to compute optimal solutions in such situations, while overcoming NP-hardness of both oracles, marks a significant advance in the state of the art in deploying game-theoretic techniques.

Figure 6(a) examines the performance of RUGGED when the size of the strategy spaces for both players is increased. These tests were conducted on WFC graphs, since they are designed to have large strategy spaces. These problems have 20 to 100 nodes and up to 5 resources. The x-axis in the figure shows the number of nodes in the graph, while the y-axis shows the runtime in seconds. Different number of defender resources are represented by different curves in the graph. For example for 40 nodes, and 5 defender resources, RUGGED took 108 seconds on average.

To speed up the convergence of RUGGED, we tried to *warm-start* the algorithm with an initial defender allocation such as min-cut-based allocations, target- and source-centric allocations, RANGER allocations and combinations of these. No significant improvement of runtime was measured; in some cases, the runtime increased because of the larger strategy set for the defender.

6.3 Algorithm Dynamics Analysis

This section analyzes the anytime solution quality and the performance of each of the three components of RUGGED: the defender oracle, the attacker oracle, and the *CoreLP*. When we solve the best-response oracle problems, they provide lower and upper bounds on the optimal defender utility, as shown in Propositions 1 and 2. Figure 6(b) shows the progress of the bounds and the *CoreLP* solution for a sample problem instance scheduling 2 defender resources on a fully connected network with 50 nodes. The x-axis shows the number of iterations and the y-axis shows the expected defender utility. The graph shows that a good solution (i.e., one where the difference in the two bounds is less than ϵ) can be computed reasonably quickly, even though the algorithm takes longer to converge to the optimal solution. For example, a solution with an allowed approximation of 10 units[10] can be computed in about 210 iterations, whereas 310 iterations are required to find the optimal solution. The difference between these two bounds gives an upper

[10]10 units is 1% of the maximum target payoff (1000).

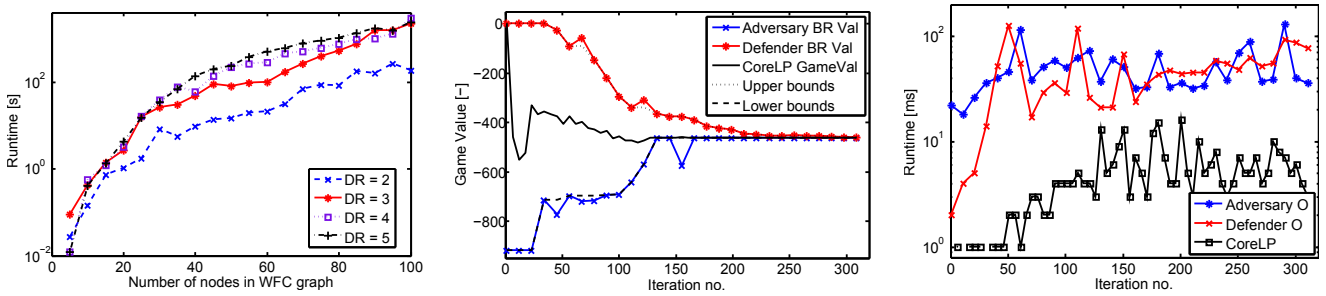

Figure 6: Results. Figure (a) shows the scale-up analysis on WFC graph of different sizes. Figure (b) shows the convergence of oracle values to the final game value and the anytime bounds. Figure (c) compares the runtimes of oracles and the core LP.

bound on the error in the current solution of the *CoreLP*; this also provides us with an *approximation* variant of RUGGED.

Figure 6(c) compares the runtime needed by the three modules in every iteration. The x-axis shows the iteration number and the y-axis shows the runtime in seconds in logarithmic scale. As expected, *CoreLP* — solving a standard linear program — needs considerably less time in each iteration than both the oracles, which solve mixed-integer programs. The figure also shows that the modules scale well as the number of iterations increases.

7. CONCLUSION AND FUTURE WORK

Optimally scheduling defender resources in a network-based environment is an important and challenging problem. Security in urban road networks, computer networks, and other transportation networks is of growing concern, requiring the development of novel scalable approaches. These domains have extremely large strategy spaces; a graph with just 20 nodes and 5 resources can have more than 2 billion strategies for both players. In this paper, we presented RUGGED, a novel double-oracle based approach for finding an optimal strategy for scheduling a limited number of defender resources in a network security environment. We showed that previous approaches can lead to arbitrarily bad solutions in such situations, and the error can be very high even in practice. We applied RUGGED to real-city maps generated from GIS data; we presented the results of applying RUGGED to the road network of Mumbai. While enhancements to RUGGED are required for deployment in some real-world domains, optimal solutions even to these problems are now within reach. The scalability of RUGGED opens up new avenues for deploying game-theoretic techniques in real-world applications.

8. ACKNOWLEDGEMENTS

This research is supported by the United States Department of Homeland Security through Center for Risk and Economic Analysis of Terrorism Events (CREATE), the Czech Ministry of Education, Youth and Sports under project number N00014-09-1-0537, the NSF CAREER grant 0953756 and IIS-0812113, ARO 56698-CI, and an Alfred P. Sloan fellowship. We thank Ron Parr, Michal Jakob and Zhengyu Yin for comments and discussions.

9. REFERENCES

[1] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking. Randomized pursuit-evasion in graphs. In *ICALP*, pages 901–912, 2002.

[2] S. Alpern. Infiltration Games on Arbitrary Graphs. *Journal of Mathematical Analysis and Applications*, 163:286–288, 1992.

[3] Y. Bachrach and E. Porat. Path Disruption Games. In *AAMAS*, pages 1123–1130, 2010.

[4] N. Basilico, N. Gatti, and F. Amigoni. Leader-Follower Strategies for Robotic Patrolling in Environments with Arbitrary Topologies. In *AAMAS*, pages 500–503, 2009.

[5] R. Chandran and G. Beitchman. Battle for Mumbai Ends, Death Toll Rises to 195. *Times of India*, 29 November 2008.

[6] J. Dickerson, G. Simari, V. Subrahmanian, and S. Kraus. A Graph-Theoretic Approach to Protect Static and Moving Targets from Adversaries. In *AAMAS*, pages 299–306, 2010.

[7] M. M. Flood. The Hide and Seek Game of Von Neumann. *MANAGEMENT SCIENCE*, 18(5-Part-2):107–109, 1972.

[8] S. Gal. *Search Games*. Academic Press, New York, 1980.

[9] E. Halvorson, V. Conitzer, and R. Parr. Multi-step Multi-sensor Hider-Seeker Games. In *IJCAI*, pages 159–166, 2009.

[10] M. Jain, E. Kardes, C. Kiekintveld, F. Ordóñez, and M. Tambe. Security Games with Arbitrary Schedules: A Branch and Price Approach. In *AAAI*, pages 792–797, 2010.

[11] H. B. McMahan, G. J. Gordon, and A. Blum. Planning in the Presence of Cost Functions Controlled by an Adversary. In *ICML*, pages 536–543, 2003.

[12] J. V. Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[13] J. Pita, M. Jain, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Using Game Theory for Los Angeles Airport Security. *AI Magazine*, 30(1), 2009.

[14] W. Ruckle, R. Fennell, P. T. Holmes, and C. Fennemore. Ambushing Random Walks I: Finite Models. *Operations Research*, 24:314–324, 1976.

[15] J. Tsai, Z. Yin, J. young Kwak, D. Kempe, C. Kiekintveld, and M. Tambe. Urban security: Game-theoretic resource allocation in networked physical domains. In *AAAI*, pages 881–886, 2010.

[16] O. Vaněk, B. Bošanský, M. Jakob, and M. Pěchouček. Transiting Areas Patrolled by a Mobile Adversary. In *IEEE CIG*, pages 9–16, 2010.

[17] A. Washburn and K. Wood. Two-person Zero-sum Games for Network Interdiction. *Operations Research*, 43(2):243–251, 1995.

[18] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in Security Games: Interchangeability, Equivalence, and Uniqueness. In *AAMAS*, pages 1139–1146, 2010.