Simplifying Urban Network Security Games with Cut-Based Graph Contraction

Hiroaki Iwashita[1], Kotaro Ohori[1], Hirokazu Anai[1], and Atsushi Iwasaki[2]
[1]Fujitsu Laboratories Ltd., Kawasaki 211-8588, Japan
{iwashita.hiroak,ohori.kotaro,anai}@jp.fujitsu.com
[2]University of Electro-Communications, Tokyo 182-8585, Japan
iwasaki@is.uec.ac.jp

ABSTRACT

The scalability of the algorithm for solving urban network security games, which is an important challenge concerning security game problems, was improved. State-of-the-art solvers have been scaled up to handle real-world networks with tens of thousands of edges; however, it can take days or more when the inputs are varied. Since they do not essentially overcome exponential growth of the strategy space with increasing graph size, an approach, which can be combined with previous ones, is proposed. In particular, a practical approach of simplifying the graphs so that they can be handled within a realistic time is devised and tested. The key idea behind this approach is to restrict the defender's pure strategies to potential ones before calculating an equilibrium solution. The restriction can be tightened for faster computation and loosened for better solution. The following three techniques for computing an optimal solution to the restricted game are proposed and evaluated: (i) contraction of the network based on the restriction, (ii) compact formulation of the optimization problem using weighted edges in place of multiple edges, and (iii) efficient solution using a mixed-integer quadratic programming oracle. They can naturally cope with an extension of the game to one taking width of the roads into account. Furthermore, a heuristic algorithm of finding effective restriction of the game is also proposed.

Keywords

Game Theory; Optimization; Security; Graph Contraction

1. INTRODUCTION

With the deteriorating security situation around the world, effective safety policies are required to prevent terrorism and drug dealing. However, all possible security checkpoints cannot be covered at every moment because resources are limited. Over recent years, to plan effective resource allocation strategies for patrolling and inspection, game-theoretic approaches have been applied. Security games modeled as Stackelberg games [18] have provided various applications in real-world domains, for example, at LA International Airport [12], the US Federal Air Marshals Service [16], and the US Coast Guard [14, 4].

As one of the important types of game, a security game for urban networks models an attacker and a defender who take decisions on a network consisting of nodes and edges. The defender's strategies represent allocations of their limited resources to edges of the network, while the attacker's strategies represent paths from any source node to any target node. In this study, improving the scalability of the algorithms, which an important challenge concerning security games [15], is focused on. In particular, the strategy spaces for the defender and the attacker increase exponentially with the number of security resources and the size of the network, respectively.

Developed for simple security games in which targets are directly defended or attacked on the basis of pure strategies of each player, a compact security-game model [9] improved the scalability of those games by several orders of magnitude. Instead of enumerating the defender strategies as k-combinations from n targets, this model introduces a "coverage vector" representing the probability that each target is covered. The feasible space of coverage vectors is characterized simply by the sum of its elements being not more than the number of defender resources. A counterpart of that model for urban network security games has not been invented yet; in particular, a coverage vector cannot be defined easily because a defender's pure strategy covers targets indirectly by blocking a huge number of attacker's pure strategies aimed at the target, which is discrete and complicated relation.

Scaling up urban-network security games to handle realistic problem sizes has been attempted in several studies. RANGER [17] obtains an approximate solution for the defender by optimizing marginal probability of selecting individual edges without considering how to combine multiple edges. RUGGED [7] and SNARES [6] are based on a double-oracle approach [10], which does not enumerate all pure strategies for either of the attacker and the defender. SNARES has successfully computed an optimal strategy for the road network of Mumbai comprising 9,503 nodes and 20,416 edges.

However, the above-mentioned algorithms cannot always be applied to real problems without any conditions; that is, the runtime of an algorithm varies significantly with changing conditions such as numbers of targets and resources [8]. In our experience, computation time to obtain a solution increases dramatically as the number of targets and/or resources is increased. Moreover, it was experimentally shown that varying minor conditions (such as source and target placement) significantly affected the runtime of every run.

In light of the above-described issues, a robust method for obtaining effective defender strategies for large-scale networks is proposed and evaluated. After the problem is defined and related works are introduced in Section 2, the key idea behind the proposed method, namely, reducing the problem by contracting the graph, is presented in Section 3. Techniques for efficiently solving the reduced problems, including a column-generation method using a mixed-integer quadratic programming oracle, are shown in Section 4. A heuris-

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016),*
J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.),
May 9–13, 2016, Singapore.

tic algorithm utilizing the proposed method, which incrementally adds candidate edges and improves the solution, is introduced in Section 5. Experimental results in Section 6 show the high quality of the determined solutions as well as the robustness of the proposed method.

2. URBAN NETWORK SECURITY GAMES

Urban network security games [17] are zero-sum games with two players: a defender and an attacker. They are defined by 6-tuple $\mathscr{G} = (V, E, S, T, U, k)$, where $G = (V, E)$ is a graph describing a road network, $S \subset V$ is a set of source nodes, $T \subset V$ is a set of target nodes, $U : V \to \mathbb{R}_{\geq 0}$ is a payoff function (value of each node) satisfying $U(v) \neq 0$ if and only if $v \in T$, and k is the number of security resources.

The defender selects at most k edges from E and allocates the resources to them. The attacker moves along E from any one of the sources, $s \in S$, to any one of the targets, $t \in T$. It is assumed that only one target can be attacked at a time; no targets but the last one (t) can be attacked even when there are multiple targets along the attacker path. If one or more defender resources are located on the path, the attack fails and both players get nothing; otherwise the defender and the attacker gain $-U(t)$ and $U(t)$ respectively.

The objective of the game is to find an optimal mixed strategy for the defender, namely, the most-effective plan for allocating the security resources, in terms of minimizing the maximum expected damage. Since the game is a zero-sum game, its optimal solution is a Nash equilibrium as well as a strong Stackelberg equilibrium [20].

Urban network security games can be formulated as follows. Let D be a set of all defender allocations and A be a set of all attacker paths. The defender's mixed strategy is denoted as $\mathbf{x} : D \to [0, 1]$, and expected damage is denoted as z. A successful attacker path ($a \in A$) causes damage $U(t(a))$, where $t(a)$ represents the target of a. The probability that attacker path $a \in A$ becomes successful against defender allocation $d \in D$ is denoted as $P(d, a)$, which, in this problem setting, is either 0 or 1. An urban network security game is formulated as the following minimax linear programming (LP) problem:

$$\min_{\mathbf{x}, z} \quad z \tag{1}$$

$$\text{s.t.} \quad z \geq U(t(a)) \sum_{d \in D} P(d, a)\, \mathbf{x}(d), \quad \forall a \in A \tag{2}$$

$$\sum_{d \in D} \mathbf{x}(d) = 1 \tag{3}$$

$$0 \leq \mathbf{x}(d) \leq 1, \quad \forall d \in D. \tag{4}$$

Solving formulas (1)–(4) gives defender's optimal mixed strategy and expected damage corresponding to it. Moreover, the solution to its dual LP problem provides attacker's optimal mixed strategy, $\mathbf{y} : A \to [0, 1]$. This simple formulation, however, cannot be applied directly to real-world problems because $|D|$ and $|A|$ increase exponentially with problem size.

2.1 Min-Cut Method

If the number of resources, k, is sufficient, the defender can protect all targets perfectly by allocating them to all minimum cut-sets separating the targets from the sources. Even in the cases that the size of a min-cut is larger than k, it is known that the defender's optimal strategy is to allocate resources to k edges randomly chosen from the cut-set if only one target exists or all targets have the same value [19].

The min-cut method reduces complexity of the problem to polynomial time of finding a minimum *s-t* cut [2, 1]. However, it does not generally provide a good strategy in the case that the targets' payoff values are not uniform.

2.2 Double-Oracle Method

To solve the minimax LP problem, RUGGED [7] uses a *double-oracle* method [10], which is a kind of constraint and column generation algorithm for solving linear programming problems. It reaches an optimal solution asymptotically by adding pure strategies one by one instead of computing the solution after enumerating all pure strategies.

Algorithm 1: Double-oracle method

```
1 Initialize D by arbitrary defender allocations
2 Initialize A by arbitrary attacker paths
3 repeat
4     Solve the minimax LP problem of equations (1)–(4), and
      let x and y be mixed strategies of the defender and the
      attacker over D and A, respectively
5     Find defender's best response to y and add it to D
6     Find attacker's best response to x and add it to A
7 until convergence
8 return x
```

The double-oracle method is outlined in Algorithm 1. The best-response oracles in lines 5 and 6 are formulated as mixed-integer linear programming problems, which often dominate computation time in RUGGED.

Instead of using the best-response oracles every time, SNARES uses fast heuristic algorithms whenever they give a better response. A method called *mincut-fanout* is used to improve the initial members of D and A. Mincut-fanout first constructs a min-cut that separates the target with the highest value from the sources; it then initializes D by defender allocations, such that each allocation covers k edges in the cut-set, and A by the attacker's best responses to them.

3. PROBLEM REDUCTION

Although the double-oracle method definitely improves the scalability of the solvers, it does not essentially overcome exponential growth of the strategy space with the increasing problem size. In other words, problem reduction can exponentially improve performance of the state-of-the-art solvers.

Generally, a set of defender's pure strategies appearing in an optimal mixed strategy is a fraction of the huge entire space. For example, only edges in $C = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$ are used in an optimal solution to the problem shown in Figure 1(a). If $C \subset E$ is determined in advance by some other algorithm or suggested by a security expert, it becomes possible to reduce the problem.

EXAMPLE 1. *To reduce $G = (V, E)$ in Figure 1(a), unused edges $e \in E \setminus C$ are contracted, and the problem is reduced to a compact one defined on $G_C = (V_C, E_C)$, as illustrated in Figure 1(b). Furthermore, multiple edges in the graph are replaced with a single edge with the capacity to make a more-compact problem on $\hat{G}_C = (\hat{V}_C, \hat{E}_C)$, as shown in Figure 1(c). In later sections, it is shown that optimal solutions computed on both G_C and $\hat{G}_C$ are also optimal on G under the restriction that the defender allocates resources to only C. In this example, C is not the only one best set of candidate edges. If $C' = \{e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$ instead of C is selected, another solution with the same quality is obtained because G_C and $G_{C'}$ are isomorphic.*

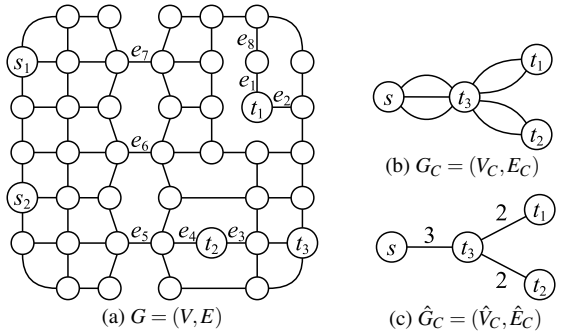


Figure 1: An example of reduction.

Algorithm 2 is a possible manual procedure for finding realistic solutions to large problems. Although quality of the solution depends on the choice of C, the computed security level for the reduced problem is guaranteed also for the original one. Security agencies can therefore determine if the solution is acceptable.

Algorithm 2: Manual task of security scheduling

```
Determine an initial set of candidate edges C
loop
    Reduce and solve the problem based on C
    if the problem cannot be solved then
        Shrink C
    else if the security level is not sufficient then
        Extend C or increase the number of defender resources
    else return
```

3.1 Contracting Unused Edges

An urban network security game with a restriction on the defender's pure strategies is considered as follows. Here, $C \subset E$ is a set of candidate edges to which the defender can allocate resources. A contracted graph, $G_C = (V_C, E_C)$, is made from an original graph, $G = (V, E)$, by *edge contraction* for all $e \in E \setminus C$, where edge contraction of e is an operation in graph theory that removes e from a graph and merges its two incident nodes. When the correspondence of those nodes is represented by $\varphi_C : V \to V_C$, the original game, $\mathscr{G} = (V, E, S, T, U, k)$, is reduced to $\mathscr{G}_C = (V_C, E_C, S_C, T_C, U_C, k)$ where

$$S_C = \{\varphi_C(s) \mid s \in S\},$$
$$T_C = \{\varphi_C(t) \mid t \in T\},$$
$$U_C(v') = \max\{U(v) \mid \varphi_C(v) = v',\ v \in V\}.$$

Additional two functions are defined for later use. First, $\widetilde{\varphi}_C : V_C \to V$ is the function that maps $v' \in V_C$ to a set of the most valuable nodes in V represented by v', that is,

$$\widetilde{\varphi}_C(v') = \arg\max_{v \in V}\{U(v) \mid \varphi_C(v) = v'\}.$$

Secondly, $\widetilde{\psi}_{E_C,E} : E_C \to C$ maps an edge in the contracted graph to the one in the original graph.

LEMMA 1. *When $e_1 \in E$ and $C = E \setminus \{e_1\}$, $\widetilde{\psi}_{E_C,E}$ maps the defender's optimal mixed strategy for $\mathscr{G}_C$ to the one for $\mathscr{G}$ that has the same quality and is optimal under the restriction that the defender allocates resources only to C.*

PROOF. Any defender allocation, $d' \subseteq E_C$, can be simply mapped to $d = \{\widetilde{\psi}_{E_C,E}(e') \mid e' \in d'\}$. Attacker path $a' \subseteq E_C$ to target $t' \in T_C$ can be mapped to $a = \{\widetilde{\psi}_{E_C,E}(e') \mid e' \in a'\}$ if it forms a path to $\widetilde{\varphi}_C(t')$; otherwise, it can be mapped to $a \cup \{e_1\}$, which forms a path to $\widetilde{\varphi}_C(t')$. These mappings do not change the equilibrium between both player's mixed strategy and the best responses against them. Therefore, quality and optimality do not change. □

THEOREM 1. *For all $C \subseteq E$, $\widetilde{\psi}_{E_C,E}$ maps the defender's optimal mixed strategy for $\mathscr{G}_C$ to the one for $\mathscr{G}$ that has the same quality and is optimal under the restriction that the defender allocates resources only to C.*

PROOF. Let $C_0 = C$ and $C_i = C_{i-1} + \{e_i\}$ for $i = 1, \ldots, m$ where $E \setminus C = \{e_1, \ldots, e_m\}$. From Lemma 1, $\widetilde{\psi}_{E_C,E_{C_1}}$ maps the defender's optimal mixed strategy for $\mathscr{G}_C$ to the one for $\mathscr{G}_{C_1}$. If $\widetilde{\psi}_{E_C,E_{C_i}}$ maps the defender's optimal mixed strategy for $\mathscr{G}_C$ to the one for $\mathscr{G}_{C_i}$, it can be mapped over again by $\widetilde{\psi}_{E_{C_i},E_{C_{i+1}}}$ to the one for $\mathscr{G}_{C_{i+1}}$. By using induction, $\widetilde{\psi}_{E_C,E_{C_m}} = \widetilde{\psi}_{E_C,E}$ maps the defender's optimal mixed strategy for $\mathscr{G}_C$ to the one for $\mathscr{G}_E = \mathscr{G}$. □

The following statements give guidelines for selecting the set of candidate edges.

THEOREM 2. *The defender's mixed strategies for G computed with two different set of candidate edges, C and C', have the same quality if G_C and $G_{C'}$ are isomorphic in consideration of node labels of sources, targets, and payoff values.*

PROOF. Apparently, $\mathscr{G}_C$ and $\mathscr{G}_{C'}$ yield the defender's optimal mixed strategies on G_C and $G_{C'}$ respectively that have the same expected utilities. According to Theorem 1, both strategies can be mapped to the ones on G while the same quality is maintained. □

THEOREM 3. *The set of candidate edges that achieves the defender's mixed strategy with the best possible quality can be composed of cut-sets separating one or more targets from all sources.*

PROOF. Let C be a set of candidate edges achieving the defender's mixed strategy with the best possible quality. If self-loop edges are included in E_C, they can be excluded from C without loss of solution quality because they are useless for attacker paths and thus for defender allocation. If non-target node $v' \in V_C \setminus T_C$ has edges only to single node $u' \in V_C$, no edges between u' and v' are used by either player and can be excluded from C without loss of solution quality. Edges between two sources in S_C can also be excluded from C safely. Consequently, all remaining candidate edges are covered by a collection of cut-sets separating one or more targets from all sources. □

3.2 Unifying Multiple Edges

While the reduced problem can be solved as is in the same way as the original one, more compact problem can be formulated by adapting the proposed model. Contraction of many edges often produces multiple edges, which are two or more edges that are incident to the same two nodes. An optimal strategy of the defender should use those edges uniformly (since they are topologically equivalent). Each pair of nodes connected by m edges do not need to be distinguished, and they can be simplified as a single edge (e) with capacity $w_e = m$, as illustrated in Figure 1. In the example, six attacker paths from s to t_1 on G_C are reduced to only one on $\hat{G}_C$. This conversion effectively suppresses combinatorial explosion of the strategy space.

A graph with multiple edges, $G = (V, E)$, is reduced to a more compact weighted graph without multiple edges, $\hat{G} = (\hat{V}, \hat{E})$, in

which each edge, $e \in \hat{E}$, has capacity w_e. The game is reduced to $\hat{\mathscr{G}} = (\hat{V}, \hat{E}, \hat{S}, \hat{T}, \hat{U}, k)$ where $\hat{S} \subset \hat{V}$ is a set of source nodes, $\hat{T} \subset \hat{V}$ is a set of target nodes, and $\hat{U} : \hat{V} \to \mathbb{R}_{\geq 0}$ is a payoff function.

Let $d_e \in \{0, \ldots, w_e\}$ denote the number of resources on edge e in defender allocation d. When the defender and the attacker play d and a respectively on $\hat{G}$, the attack succeeds at probability $P(d,a) = \prod_{e \in a}(1 - d_e/w_e)$. Therefore, this game is formulated as the following minimax LP problem:

$$\min_{\hat{\mathbf{x}}, z} \quad z \tag{5}$$

$$\text{s.t.} \quad z \geq \hat{U}(t(a)) \sum_{d \in \hat{D}} \prod_{e \in a} \left(1 - \frac{d_e}{w_e}\right) \hat{\mathbf{x}}(d), \ \ \forall a \in \hat{A} \tag{6}$$

$$\sum_{d \in \hat{D}} \hat{\mathbf{x}}(d) = 1 \tag{7}$$

$$0 \leq \hat{\mathbf{x}}(d) \leq 1, \quad \forall d \in \hat{D}, \tag{8}$$

where $\hat{D}$ and $\hat{A}$ are all defender allocations and all attacker paths on $\hat{G}$ respectively, $\hat{\mathbf{x}} : \hat{D} \to [0,1]$ is the defender's mixed strategy, z is an expectation of the damage, and $\hat{U}(t(a))$ is the amount of damage when attacker path $a \in \hat{A}$ succeeds.

In fact, this formulation is usable independently of graph contraction. It is a natural extension of urban network security games that edge e is assumed to have capacity w_e, where w_e resources are required to block e.

4. SOLUTION TECHNIQUES

If multiple edges are not unified, the problem after reduction can be still formulated as (1)–(4); conventional techniques are applicable to it as they are. This section presents the solution techniques that can be applied to the problem (5)–(8). Without these techniques, all pure strategies, $\hat{D}$ and $\hat{A}$, must be enumerated beforehand.[1]

4.1 Selecting Pure Strategies

To reduce the cost for solving the minimax LP, the strategy spaces are restricted without sacrificing quality of solutions. First, the defender's strategies that do not affect the quality are identified.

Let $R_e = \{0, \ldots, w_e\}$ and $R_e^- = R_e \setminus \{0, w_e\}$. Edge e is *partially blocked* by defender allocation d when $d_e \in R_e^-$. The following theorem guarantees that even if every defender allocation that partially blocks two or more edges is ignored, a mixed strategy for the defender that yields the optimal expected utility still exists.

THEOREM 4. *At least one optimal mixed strategy for the defender includes no defender allocation by which two or more edges are partially blocked.*

PROOF. Let us assume that defender allocation $\check{d}$ partially blocks edges i and j in an optimal mixed strategy $\hat{\mathbf{x}}$, that is, $1 \leq \check{d}_i \leq w_i - 1$ and $1 \leq \check{d}_j \leq w_j - 1$. New defender allocations $\grave{d}$ and $\acute{d}$ are made by moving as many resources as possible from j to i and i to j respectively:

$$\grave{d}_e = \begin{cases} \check{d}_e + \alpha & \text{if } e = i \\ \check{d}_e - \alpha & \text{if } e = j \\ \check{d}_e & \text{otherwise}, \end{cases} \qquad \acute{d}_e = \begin{cases} \check{d}_e - \beta & \text{if } e = i \\ \check{d}_e + \beta & \text{if } e = j \\ \check{d}_e & \text{otherwise}, \end{cases}$$

$$\alpha = \min(w_i - \check{d}_i, \check{d}_j), \qquad \beta = \min(\check{d}_i, w_j - \check{d}_j),$$

where α and β are the largest possible numbers of moved resources. Note that at most either i or j is partially blocked after the move. A new mixed strategy is defined as:

$$\hat{\mathbf{x}}'(d) = \begin{cases} 0 & \text{if } d = \check{d} \\ \hat{\mathbf{x}}(d) + \frac{\beta}{\alpha+\beta}\hat{\mathbf{x}}(\check{d}) & \text{if } d = \grave{d} \\ \hat{\mathbf{x}}(d) + \frac{\alpha}{\alpha+\beta}\hat{\mathbf{x}}(\check{d}) & \text{if } d = \acute{d} \\ \hat{\mathbf{x}}(d) & \text{otherwise}. \end{cases}$$

The difference of blocking probability of i between $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ is:

$$\begin{aligned} &\sum_{d \in \hat{D}} \frac{d_i}{w_i} \left(\hat{\mathbf{x}}'(d) - \hat{\mathbf{x}}(d)\right) \\ &= -\frac{\check{d}_i}{w_i}\hat{\mathbf{x}}(\check{d}) + \frac{\grave{d}_i}{w_i} \cdot \frac{\beta}{\alpha+\beta}\hat{\mathbf{x}}(\check{d}) + \frac{\acute{d}_i}{w_i} \cdot \frac{\alpha}{\alpha+\beta}\hat{\mathbf{x}}(\check{d}) \\ &= -\frac{\check{d}_i}{w_i}\hat{\mathbf{x}}(\check{d}) + \frac{\check{d}_i + \alpha}{w_i} \cdot \frac{\beta}{\alpha+\beta}\hat{\mathbf{x}}(\check{d}) + \frac{\check{d}_i - \beta}{w_i} \cdot \frac{\alpha}{\alpha+\beta}\hat{\mathbf{x}}(\check{d}) \\ &= 0. \end{aligned}$$

It follows that the probability of blocking i remains unchanged by replacing $\hat{\mathbf{x}}$ with $\hat{\mathbf{x}}'$. Similarly, the probability of blocking j also remains. Therefore, $\hat{\mathbf{x}}'$ is also optimal. By repeating the above modification, it is possible to find an optimal mixed strategy including no defender allocation by which two or more edges are partially blocked. □

Secondly, the way to reduce the attacker's strategy space is rather simple. It can be done by excluding any path passing over target t and attacking t' such that $\hat{U}(t) \leq \hat{U}(t')$. Such a path is worse than its prefix attacking t and is not required by the attacker's optimal strategy. In Figure 1(c), the valid attacker paths are s–t_3 and s–t_3–t_1 if $\hat{U}(t_2) \leq \hat{U}(t_3) < \hat{U}(t_1)$.

4.2 Column Generation

To avoid an explosion of the number of defender's pure strategies to be considered in the case of the minimax LP problem (5)–(8), Algorithm 3, which is a column generation technique using defender's best-response oracle, is introduced.

Algorithm 3: Single-oracle column generation

```
Initialize D by arbitrary defender allocation
Initialize A by all attacker paths
repeat
    Solve the minimax LP problem (5)–(8), and let x̂ and ŷ be
    mixed strategies of the defender and the attacker using D
    and A, respectively
    Find defender's best response to ŷ and add it to D
until convergence
return x̂
```

The oracle finds one of the best defender allocations, d, against the attacker's mixed strategy, $\hat{\mathbf{y}}$. It is formulated as a mixed-integer nonlinear programming (MINLP) problem:

$$\min_{d} \quad \sum_{a \in \hat{A}} \hat{U}(t(a)) \hat{\mathbf{y}}(a) \prod_{e \in a} \left(1 - \frac{d_e}{w_e}\right) \tag{9}$$

$$\text{s.t.} \quad \sum_{e \in \hat{E}} d_e \leq k \tag{10}$$

$$d_e \in R_e, \quad \forall e \in \hat{E}. \tag{11}$$

Unfortunately, MINLP introduces a further level of difficulty and can not be accepted directly by many generic optimization solvers.

[1] Actually, it is possible to some extent because the problem reduction drastically compresses the strategy spaces.

It is thus transformed into a mixed-integer quadratic programming (MIQP) problem, which is much easier to solve quickly by state-of-the-art optimization solvers. Let $\lambda(e,r) \in \{0,1\}$ be a variable representing that the number of resources allocated on edge e is r ($d_e = r$). According to Theorem 4, it can be assumed that at most one pair (e,r) of $e \in a$ and $r \in R_e^-$ for each $a \in \hat{A}$ satisfies $\lambda(e,r) = 1$. Equations (9)–(11) are rewritten as follows:

$$\min_{\lambda,\omega} \quad \sum_{a\in\hat{A}} \hat{U}(t(a))\hat{\mathbf{y}}(a)\left(1 - \sum_{e\in a}\sum_{r\in R_e^-} \frac{r}{w_e}\lambda(e,r)\right)(1-\omega(a)) \quad (12)$$

$$\text{s.t.} \quad \sum_{e\in\hat{E}}\sum_{r\in R_e} r\lambda(e,r) \le k \quad (13)$$

$$\sum_{e\in\hat{E}}\sum_{r\in R_e^-} \lambda(e,r) \le 1 \quad (14)$$

$$\sum_{r\in R_e} \lambda(e,r) \le 1, \quad \forall e \in \hat{E} \quad (15)$$

$$\omega(a) \le \sum_{e\in a} \lambda(e,w_e), \quad \forall a \in \hat{A} \quad (16)$$

$$\lambda(e,r) \in \{0,1\}, \ \forall e \in \hat{E}, \ \forall r \in R_e \quad (17)$$

$$\omega(a) \in [0,1], \quad \forall a \in \hat{A}. \quad (18)$$

Equation (13) indicates that the total number of allocated resources does not exceed k. Equations (14) and (15) constrain no two edges to be partially blocked. Equation (16) defines the relation between ω and λ; that is, $\omega(a) = 1$ if attacker path a is completely blocked by the defender at some edge $e \in a$; otherwise, $\omega(a) = 0$.

The attacker's best response oracle is not provided here; the all members of $\hat{A}$ is enumerated in advance and $\hat{D}$ only is generated incrementally. As for the proposed method, it is empirically confirmed that the number of valid attacker paths does not increase explosively.

5. HEURISTIC ALGORITHM

Algorithm 2 is replaced by introducing an automatic method for selecting appropriate candidate edges for defender allocations. The min-cut method for single-target problems [19] and mincut-fanout in SNARES [6] imply that min-cuts are strongly associated with defender strategies. A min-cut separating a subset of targets, $T' \subseteq T$, from a set of all sources, S, gives the best set of edges for uniformly enhancing the security of targets in T'. The idea proposed here, called "min-cut arrangement," is to combine multiple such S-T' min-cuts.

A naive implementation of min-cut arrangement would be one that tries to use all $2^{|T|} - 1$ nonempty subsets of T as T'. It will surely find the best solution among min-cut-arrangement-based algorithms; however, it is not practical in terms of computational complexity when $|T|$ is large.

Algorithm 4: MiCANS(V,E,S,T,U,k)

```
1 C^0 ← ∅,  T^1 ← arg max_t U(t),  i ← 1
2 loop
3     C^i ← C^{i-1} ∪ FindCandidateEdges(V,E,S,T^i)
4     if C^i = C^{i-1} then return x̂^{i-1}
5     (x̂^i, ŷ^i) = ReduceAndSolve(V,E,S,T,U,k,C^i)
6     T^{i+1} ← {v | v ∈ φ̃_{C^i}(t(a)), ŷ^i(a) > 0}
7     i ← i+1
```

Algorithm 4, called MiCANS, is based on min-cut arrangement that aims for both runtime and quality. C^i and T^i represent a set of candidate edges and a set of *critical targets* in the i-th iteration, respectively. Critical targets are the ones that will suffer the highest expectation of damage for the current equilibrium solution. Initially, it is assumed that no target is protected; thus T^1 is a set of the most valuable targets (line 1). Subsequent sets of critical targets are found by extracting targets with non-zero attack probability from $\hat{\mathbf{y}}^i$ and mapping them back to the original graph (line 6).

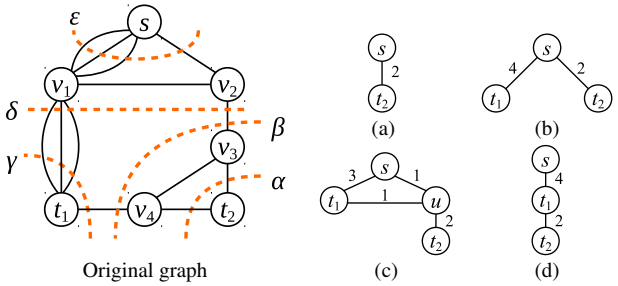


Figure 2: An example of execution.

FindCandidateEdges(V,E,S,T^i) returns a set of edges by computing a min-cut separating T^i from S. When multiple S-T^i min-cuts exist, some heuristics for selecting one or more of them must be devised. One of the simplest heuristics is to choose the min-cut that minimizes the size of the node group containing T^i.

ReduceAndSolve(V,E,S,T,U,k,C^i) stands for the method described in Section 3 and Section 4, which contracts all unused edges, $e \in E \setminus C^i$, of $G = (V,E)$, unifies every bundle of multiple edges into a weighted edge, and solves the minimax LP problem with column generation. The graph is refined in each iteration by adding minimum cut-set separating critical targets. It is repeated until the candidate edges stop growing.

The MiCANS algorithm can be interpreted as a kind of (incomplete) column generation at higher level of abstraction for solving the minimax LP (1)–(4). The defender's strategy space, D, is expanded indirectly by adding candidate edges to E'. Ideally, they should be chosen in such a way that no other choices reduce expected damage if they do not. Although the min-cut edges are efficient candidates for improving the solution, they do not always satisfy such a property; that is, this algorithm does not guarantee that it always converges to an optimal solution.

EXAMPLE 2. *Figure 2 shows an example of execution, in which s is the source, t_1 and t_2 are the targets with values 1 and 2, respectively, and the number of resources is 2. The first critical target set is $T^1 = \{t_2\}$. Two min-cuts separate t_2 from s: α and β. In either case that $C^1 = \alpha$ or $C^1 = \beta$, the first reduced graph is (a), t_2 is completely guarded by two defender resources in the first iteration, expected damage of t_2 changes to zero, and therefore $T^2 = \{t_1\}$. At the second iteration, three min-cuts, γ, δ, and ε, exist between s and t_1. If $C^2 = \alpha \cup \gamma$ is chosen, the reduced graph is (b). As the optimal mixed strategy for the defender balances expected damage of attacks on t_1 and t_2 to 4/5, both targets become critical: $T^3 = \{t_1, t_2\}$. The third reduced graph is (c) and the expected damage is reduced to 2/3, which is the optimal value. The algorithm terminates because $\{t_1, t_2\}$ is critical again. If $C^2 = \beta \cup \varepsilon$ is chosen at the second iteration, reduced graph (d) is obtained, and it yields another solution with the same expected damage, 2/3, even though (c) and (d) are topologically different. Optimal solutions are also obtained from other combinations of min-cuts.*

EXAMPLE 3. *An optimal solution cannot be obtained from the graph in Figure 2(d) if the number of resources is 3. When the de-*

fender completely blocks edge $\{t_1,t_2\}$, the remaining resource must be allocated to edge $\{s,t_1\}$. The security of t_1 is strengthened by blocking one-quarter of attacks. On the other hand, an optimal solution is obtained from the graph in Figure 2(c), in which resource allocation on $\{s,u\}$ and $\{t_1,u\}$ completely protects t_2 and one more resource on $\{s,t_1\}$ blocks one-third of attacks on t_1. In fact, for an optimal solution, it is necessary to choose two out of β, γ, and δ.

Under the assumption that the min-cut that minimizes the size of the target-side is always chosen, MiCANS overlays the cut-sets that do not cross each other. In that case, the size of the reduced graph increases by one after each iteration. The number of iterations in MiCANS, as well as the final size of the reduced graph, becomes linear in relation to the number of targets with different values, namely, $O(|T|)$. In a large problem in which the reduced graph becomes small enough in comparison with the original graph, $G = (V,E)$, min-cut computation dominates the running time. It takes polynomial time with respect to the size of G, namely, $O(|V| \cdot |E|^2)$. Consequently, the total cost of MiCANS under this assumption is $O(|T| \cdot |V| \cdot |E|^2)$.

6. EXPERIMENTAL RESULTS

We have implemented MiCANS and SNARES in Python with the NetworkX graph library [13] and the Gurobi mathematical programming solver. Experiments were performed on a machine with 2.9 GHz CPU and 10 GB memory. The following three types of network models were used:

1. A random road network model called GRE (grid model with random edges) [11], which is a planar connected graph made of a $l \times w$ square grid of nodes. Horizontal/vertical edges between neighbors are controlled by probability p, and diagonal ones by q. It is reported that the values of p spread in $[0.3,0.9]$ and q in $[0.1,0.7]$ in models that best match the road networks of Europe. Sources were fixed to w nodes at the bottom end, while the target nodes were selected randomly.

2. Random geometric graphs, which have also been shown to mimic some properties of real road networks [3] and were used to evaluate SNARES. n nodes are distributed uniformly at random in a unit square, and node pairs with a distance not more than a given threshold, d, are connected by edges. Source and target nodes were selected randomly.

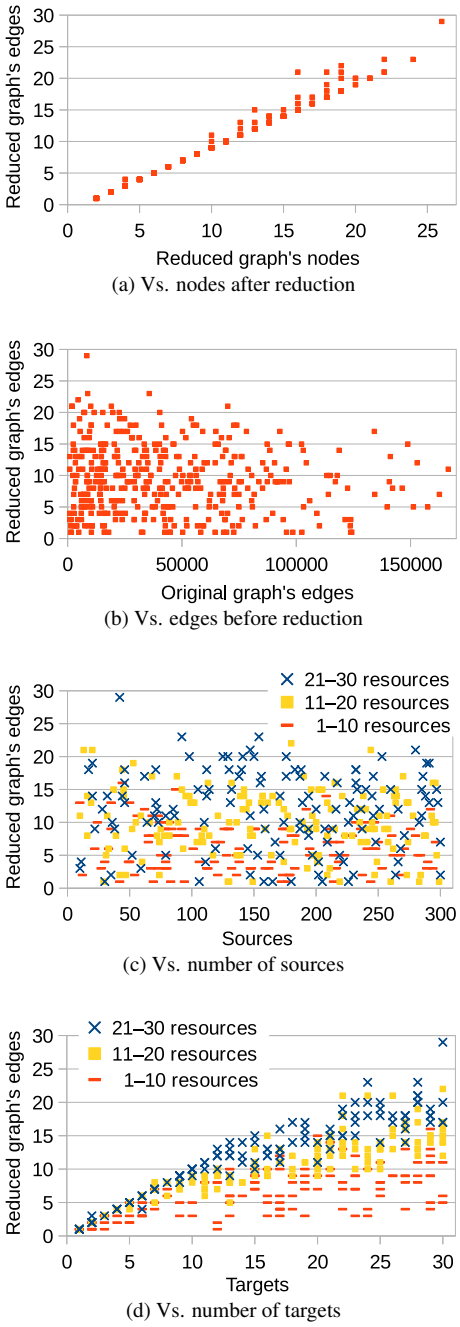
3. Real road network data collected in Mumbai, made of 21,132 nodes and 33,603 edges. It was extracted from a rectangle area of latitude 18.84 to 19.36° and longitude 72.75 to 73.16° of Open Street Map [5]. Source and target nodes were selected randomly.

6.1 Effectiveness of Graph Reduction

400 random instances of the GRE were generated; their horizontal and vertical sizes were chosen randomly from $\{10,\ldots,300\}$, p from $[0.3,0.9]$, and q from $[0.1,0.7]$.

Figure 3(a) shows the relation between numbers of nodes and edges of the final reduced graph made by min-cut arrangement. It is confirmed that the numbers of nodes and edges are about the same in every reduced graph; that is, reduced graphs are generally sparse. That is the reason why the number of attacker paths does not explode in reduced graphs. The actual number of attacker paths enumerated on every reduced graph is less than 200 in these examples.

The comparison between original graphs and reduced ones in Figure 3(b) show that the numbers of edges are reduced by several



(a) Vs. nodes after reduction

(b) Vs. edges before reduction

(c) Vs. number of sources

(d) Vs. number of targets

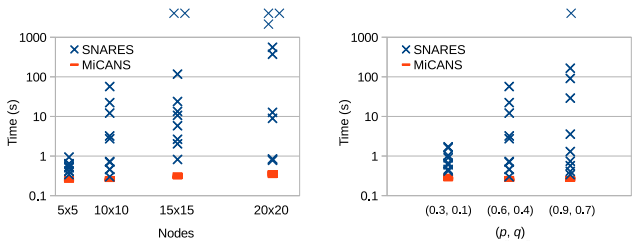
Figure 3: Number of edges after reduction.

orders of magnitude. It should be further emphasized that no direct correlation was found between them.

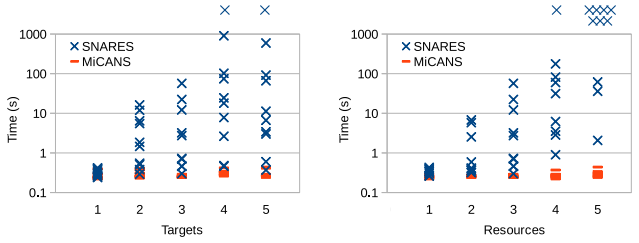
Figure 3(c) and Figure 3(d) shows that the size of reduced graph is strongly related to the number of targets, but not to sources. The number of edges in each reduced graph becomes likely to proportion to the number of targets when sufficient resources are available with respect to the number of targets.

6.2 Comparison with SNARES

For the GRE graphs, central parameter values are chosen as 10×10 nodes, $(p,q) = (0.6,0.4)$, 3 targets, and 3 resources. Figure 4 presents runtime in seconds when varying (a) number of nodes,

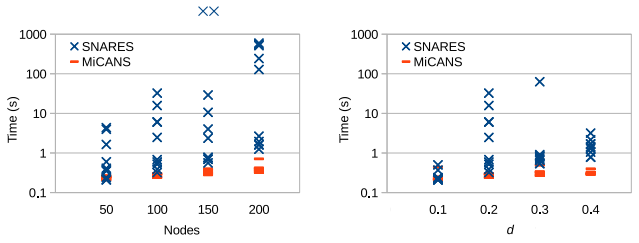


(a) Varying number of nodes (b) Varying edge probabilities



(c) Varying number of targets (d) Varying number of resources

Figure 4: Runtime for small GRE graphs.



(a) Varying number of nodes (b) Varying distance threshold

(c) Varying number of targets (d) Varying number of resources

Figure 5: Runtime for random geometric graphs.

(b) edge probabilities, (c) number of targets, and (d) number of resources. For each parameter setting, 10 graph instances were generated with different random seeds. The marks on the top margin of the figures indicate the number of runs (of SNARES) that did not finish within 1,000 seconds. All the results show that execution time of SNARES varies significantly, while MiCANS is very robust, and all runs are finished within a second.

Figure 5 presents runtime in seconds for random geometric graphs, where central parameter values were selected as 100 nodes, $d = 0.2$, 3 sources, 3 targets, and 3 resources. The same trends as the GRE graphs can be found in these graphs.

For the problem settings of the Mumbai road network in [6], where the number of resources is 1, 5, 10, or 15, the number of targets is 4 or 8, and the number of sources is fixed to 3, every run of MiCANS took 10 to 30 seconds, including 10 seconds of data reading. SNARES often did not end overnight when the number of resources was set to 5.

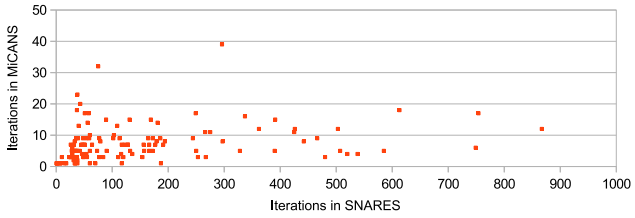
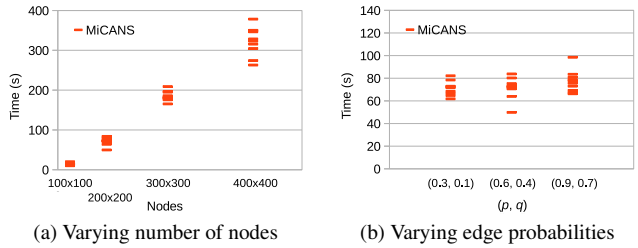
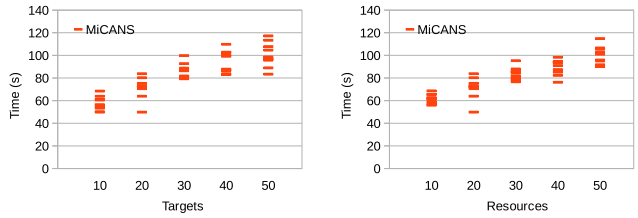


Figure 6: Total number of inner iterations for GRE graphs.



(a) Varying number of nodes (b) Varying edge probabilities



(c) Varying number of targets (d) Varying number of resources

Figure 7: Runtime for large GRE graphs.

Numbers of inner iterations (number of runs of minimax optimizations) for the algorithms are compared in Figure 6. This comparison also shows a significant reduction of iterations by MiCANS and no clear correlation between difficulties concerning the original problem and the reduced one.

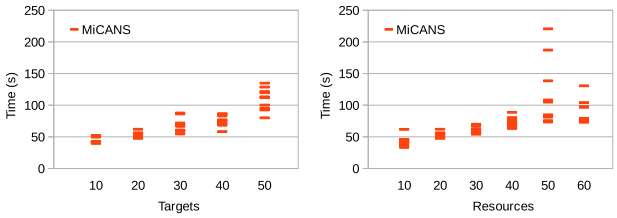
Expected payoff values were also compared for the problems that were solved by both SNARES and MiCANS in these experiments (154 GRE graph problems, 154 random geometric graph problems, and 124 Mumbai road network problems). The comparison confirmed that they are always the same; it thus follows that MiCANS successfully found the optimal solutions for these problems.

6.3 Scalability

Scalability of MiCANS was evaluated by using more-difficult problems, which our implementation of SNARES could not solve. It was performed using 10 random instances for each parameter setting. Note that y-axis is changed from a logarithmic scale to a linear scale.

Figure 7 presents runtime in seconds for larger GRE graphs, where central parameter values were selected as 200×200 nodes, $(p,q) = (0.6, 0.4)$, 20 targets, and 20 resources. It shows that the algorithm is still stable for large problems. When the original graph size is very large (e.g. >100k nodes), runtime is dominated by min-cut computation. These results are similar to those for large random geometric graphs.

Figure 8(a) presents runtime in seconds for the Mumbai road network when the number of targets was varied from 10 to 40 and both numbers of sources and resources were fixed to 20. Figure 8(b) presents runtime in seconds when the number of resources was var-



(a) Varying number of targets (b) Varying number of resources

Figure 8: Runtime for Mumbai road network.

Table 1: Solution quality

(a) GRE graphs

	Optimal solutions	Worst case	Time (sec.)
MiCANS	7342/7412 (99.056%)	+12.92%	433.7
AllComb	7409/7412 (99.960%)	+0.08%	4008.0

(b) Random geometric graphs

	Optimal solutions	Worst case	Time (sec.)
MiCANS	7364/7375 (99.851%)	+4.18%	246.7
AllComb	7375/7375 (100.00%)	+0.00%	1836.4

(c) Random graphs

	Optimal solutions	Worst case	Time (sec.)
MiCANS	8280/8295 (99.819%)	+7.11%	313.1
AllComb	8292/8295 (99.964%)	+0.75%	1904.9

ied from 10 to 60 and both numbers of sources and targets were fixed to 20.

Overall, the results for large random geometric graphs, large GRE graphs, and the Mumbai graph are similar. The performance of MiCANS depends on size of the original graph, number of targets, and number of resources. When the original graph size is very large (e.g., $|E| > 100$k), computation time is dominated by min-cut computation. All results show gradual growth in computational cost with respect to size of the problems.

6.4 Solution Quality

Huge number of random tests were performed to evaluate quality of solutions. The number of optimal solutions, the worst-case quality, and the total time for solving all problems are shown in Table 1. For the GRE graphs, horizontal and vertical sizes were chosen randomly from $\{3,\ldots,10\}$, p and q from $[0,1]$, and the numbers of targets and resources from$\{1,\ldots,5\}$. For random geometric graphs, the number of nodes was chosen randomly from $\{10,\ldots,100\}$, d from $[0.1,0.4]$, and the numbers of targets and resources from$\{1,\ldots,5\}$. The number of sources was fixed to 3. Additional tests on random graphs were also performed. 10,000 random instances for each graph types were generated, and the solutions of MiCANS and AllComb were compared with the optimal solutions computed by SNARES, except for the instances that are trivial (perfectly defensible) or unsolvable by SNARES. AllComb enumerates all combinations of targets and selects two min-cuts maximizing and minimizing the size of the node group containing the targets for each combination. MiCANS did not find an optimal solution on rare occasions, and it produced trade-offs between quality and speed.

Figure 9 shows one of the rare cases where both MiCANS and AllComb did not find an optimal solution. The candidate edges selected by MiCANS were $\alpha \cup \beta \cup \gamma$ (expected damage = 35.804); while $\alpha \cup \beta \cup \delta$ were selected by SNARES (expected damage = 35.774). Contrary to expectations, a non-minimum cut, δ, must be selected. δ shares more edges with α and β than γ does, which seems important in this subtle balance of conditions.

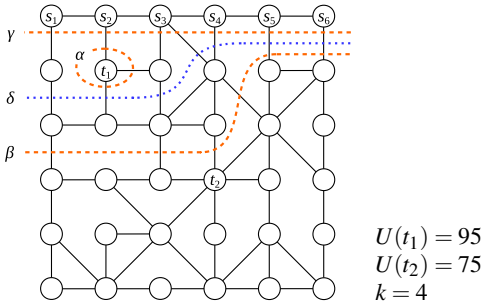


Figure 9: An example of non-optimal solution.

6.5 Warm-Starting SNARES

For some situations in which it is mandatory to guarantee optimality of the solution, MiCANS should still be useful for generating initial sets of the pure strategies for SNARES, which originally used mincut-fanout. Defender allocations on the reduced graph computed by MiCANS can be mapped directly to the original graph. Attacker paths on the reduced graph, however, cannot be restored easily; therefore, we used the shortest paths from the source to the critical targets. They are not optimal and will limit the effectiveness of warm-starting even though defender allocations are optimal. Our preliminary experiments of replacing mincut-fanout with this warm-starting method on small GRE and random geometric graphs showed that computation speed of SNARES is increased five times on average.

7. CONCLUSIONS

A practical approach of simplifying massive urban network security games so that they can be solved within a realistic time was proposed and evaluated. The key idea underpinning this approach is to restrict the defender's pure strategies to potential ones before calculating an equilibrium solution. This restriction can be tightened for faster computation and loosened for better solutions. Output of the proposed method always gives a feasible mixed strategy, which shows an upper bound of the expected damage.

Efficient techniques to solve the problems by using weighted edges to suppress combinatorial explosion of the strategy space were also proposed. These techniques can naturally cope with an extension of the game to one taking width of roads into account, where multiple resources are required for defense.

The weighted edges, however, introduced further level of difficulty in the defender's best-response oracle. It was resolved by identifying the defender's pure strategies that are not essential for optimal solutions and then converting the oracle to an easier problem.

Furthermore, an algorithm called MiCANS, which automatically reduces the original graph to a very small one by abstracting edges that are not essential for the defender's optimal strategy, was proposed. It is shown empirically that MiCANS sometimes fails to find optimal solutions, while the quality of its result is close to the optimal one, even if it is not optimal. Immediate future work will be focused on finding a good bound of the solution quality.

Acknowledgments

This research was partially supported by KAKENHI 26280081.

REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. MIT press, 2009.

[2] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[3] D. Eppstein and M. T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, pages 16:1–16:10, 2008.

[4] F. Fang, A. X. Jiang, and M. Tambe. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 957–964, 2013.

[5] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[6] M. Jain, V. Conitzer, and M. Tambe. Security scheduling for real-world networks. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 215–222, 2013.

[7] M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 327–334, 2011.

[8] M. Jain, K. Leyton-Brown, and M. Tambe. The deployment-to-saturation ratio in security games. In *Conference on Artificial Intelligence (AAAI)*, pages 1362–1370, 2012.

[9] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 689–696, 2009.

[10] H. B. McMahan, G. J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *International Conference on Machine Learning*, pages 536–543, 2003.

[11] W. Peng, G. Dong, K. Yang, and J. Su. A random road network model and its effects on topological characteristics of mobile delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 13(12):2706–2718, 2014.

[12] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 125–132, 2008.

[13] D. A. Schult and P. Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, volume 2008, pages 11–16, 2008.

[14] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. Protect: A deployed game theoretic system to protect the ports of the united states. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 13–20, 2012.

[15] M. Tambe, A. X. Jiang, B. An, and M. Jain. Computational game theory for security: Progress and challenges. In *AAAI Spring Symposium on Applied Computational Game Theory*, 2014.

[16] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordonez, and M. Tambe. Iris-a tool for strategic security allocation in transportation networks. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2009.

[17] J. Tsai, Z. Yin, J.-y. Kwak, D. Kempe, C. Kiekintveld, and M. Tambe. Urban security: Game-theoretic resource allocation in networked physical domains. In *Conference on Artificial Intelligence (AAAI)*, pages 881–886, 2010.

[18] H. Von Stackelberg. *Marktform und Gleichgewicht*. J. Springer, 1934.

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

[20] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: Interchangeability, equivalence, and uniqueness. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1139–1146, 2010.