Optimal Interdiction of Urban Criminals with the Aid of Real-Time Information

Youzhi Zhang,[1] Qingyu Guo,[1] Bo An,[1] Long Tran-Thanh,[2] Nicholas R. Jennings[3]
[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]Department of Electronics and Computer Science, University of Southampton, UK
[3]Departments of Computing and Electrical and Electronic Engineering, Imperial College, UK
[1]{yzhang137, qguo005, boan}@ntu.edu.sg, [2]ltt08r@ecs.soton.ac.uk, [3]n.jennings@imperial.ac.uk

Abstract

Most violent crimes happen in urban and suburban cities. With emerging tracking techniques, law enforcement officers can have real-time location information of the escaping criminals and dynamically adjust the security resource allocation to interdict them. Unfortunately, existing work on urban network security games largely ignores such information. This paper addresses this omission. First, we show that ignoring the real-time information can cause an arbitrarily large loss of efficiency. To mitigate this loss, we propose a novel *NEtwork purSuiT game (NEST)* model that captures the interaction between an escaping adversary and a defender with multiple resources and real-time information available. Second, solving NEST is proven to be NP-hard. Third, after transforming the non-convex program of solving NEST to a linear program, we propose our incremental strategy generation algorithm, including: (i) novel pruning techniques in our best response oracle; and (ii) novel techniques for mapping strategies between subgames and adding multiple best response strategies at one iteration to solve extremely large problems. Finally, extensive experiments show the effectiveness of our approach, which scales up to realistic problem sizes with hundreds of nodes on networks including the real network of Manhattan.

1 Introduction

In 2016, there were nearly 5.36 million violent crimes across the United States, most of which happened in the urban and suburban cities (NCVRW 2016). Considering just the commercial banks, 4,185 robberies happened (FBI 2017). When an urban crime occurs, the top priority of law enforcement officers is to dispatch the limited security resources to capture the criminal. Nowadays, novel pursuit devices such as the StarChase GPS-based system (Gaither et al. 2017), UAVs and helicopters can provide the police with the real-time location of the criminal. Equipped with the up to date information, the police can dynamically adjust the resource allocation plan to interdict the adversary.

Unfortunately, existing work on resource allocation in network security ignores the real-time information about the adversary's location (Jain et al. 2011; McCarthy et al. 2016; Wang et al. 2018). Specifically, in existing urban network security games (NSG), the defender usually deploys a time-independent strategy without considering the dynamic relocation of security resources (Jain et al. 2011). Although a time-dependent policy is discussed in (Zhang et al. 2017), they do not take the real-time adversarial information into consideration, which can cause a huge loss of effectiveness as we will show later. Other branches of related work also suffer from similar myopia, including pursuit-evasion games (Parsons 1978) and patrolling security games (Vorobeychik et al. 2014) (see the next section).

To incorporate such real-time information, we model the urban network security problem as a zero-sum *NEtwork purSuiT game (NEST)*. In NEST, the adversary picks one escape path to the outside world, while the defender, tracking the adversary's previous moves, decides on the relocation of her security resources (modeled as a finite-horizon Markov Decision Process (MDP)). Specifically, in the MDP, each state includes information about the adversary's previous moves and the locations of defender resources, and each action is the next location of the defender's resources. This results in a game with a combinatorial action space of each state in the MDP for the defender, where the defender has multiple resources with several actions each. For example, for just ten resources with four actions for each, there are more than one million joint actions of resources in a given state. This large strategy space makes the existing exact solution approaches unable to solve NEST efficiently. These approaches include: i) the approach for the imperfect recall game, i.e., normal-form game with sequential strategies (NFGSS), because their incremental strategy generation (*ISG*) algorithm does not set bounds to cut branches in its best-response (*BR*) oracle that uses a depth-first search through the whole state space to compute the best response (Bošanskỳ et al. 2015); and ii) the approach for the perfect recall game (Von Stengel 1996) because it will lead to a significantly larger strategy space than the imperfect recall approach (see the discussion in (Bošanskỳ et al. 2015)), and the bounds are very loose in their branch-and-bound *BR* (Bošanskỳ et al. 2014).

In this context, after modeling NEST as an imperfect-recall game to reduce the strategy space, this paper makes four additional key contributions. First, we show that the ignorance of real-time information in existing work can lead to an arbitrarily large loss of efficiency. Second, we show that solving NEST is NP-hard. Third, after transforming the non-convex program of solving NEST to a linear program (*LP*), we propose our *ISG* with the following novelty: (i) novel pruning techniques in our *BR*, including: (1) providing tight lower and upper bounds by exploiting the combinatorial structure of each state's action space; and (2) handling imperfect recall by developing an effective lower bound transition method that causes an evaluated state to be reevaluated only if a prefix state with a smaller lower bound transits to it; and (ii) novel techniques to speed up our *ISG* to solve extremely large-scale problems, which include: (1) mapping the computed defender strategy by *BR* between similar subgames to speed up *BR*, which also speeds up *LP* if this strategy is optimal; and (2) adding multiple best response strategies to the restricted game at one iteration against different sampled adversary strategies to reduce the number of iterations for convergence. Finally, experiments show that our approach can scale up to problem sizes with hundreds of nodes on networks including the real network of Manhattan. Thus, for the first time, this paper shows that *ISG* techniques can also be scaled for dynamic games when best-response oracles are tuned with domain-specific pruning techniques, and other domain-specific improvements are employed.

2 Related Work

The pursuit-evasion game (PEG) typically assumes that the evader knows the locations of the pursuer's units, but the pursuer does not know the location of the evader (Parsons 1978; Horák and Bošanskỳ 2016). While our model is a variant of PEG, we focus on a realistic setting in our motivating scenario where the defender has the adversary's real-time location information. Most of the work on PEGs cannot capture the special structure of specific problems. For example, although the partially observable stochastic game (Horák, Bosanskỳ, and Pechoucek 2017) is domain-independent, their model focuses on infinite-horizon games. Similarly, the patrolling security game (PSG) (Basilico, Gatti, and Amigoni 2009; Vorobeychik et al. 2014), where the defender defends an environment against an unseen intruder who needs multiple turns to perform the attack, is typically modeled as a stochastic game with infinite horizon. Recently, PSGs have been extended to cover the situation that the defender receives a spatially uncertain signal after being attacked (Basilico, De Nittis, and Gatti 2017; Basilico et al. 2017), and has to reach the attacked target to catch the attacker. However, this signal response game does not consider real-time information after receiving the signal due to the fact that no more information is available for the defender after receiving the signal.

ISG has been successfully used to exactly solve many games (Jain et al. 2011; Bošanskỳ et al. 2014; 2015; Wang et al. 2018), whose core part is to compute the best response strategy. However, the existing exact solution approaches for both imperfect recall games and perfect recall games are unable to solve NEST efficiently. Firstly, NEST is similar to an imperfect recall game NFGSS (Bošanskỳ et al. 2015), but NFGSS does not consider real-time information. Note that the general form of imperfect recall games cannot be exactly solved by a linear program (Čermák et al. 2018). Although NFGSS is solved by combining *ISG* with the network-flow representation of strategies (Jiang et al. 2013), its algorithm is impractical for the large NEST as their *ISG*'s *BR* with a depth-first search does not set bounds to cut branches and then evaluates all related states to compute the best response. Secondly, if NEST is modeled as a perfect recall game (Von Stengel 1996), it will lead to a significantly larger strategy space than the imperfect recall approach (see the discussion in (Bošanskỳ et al. 2015)). Moreover, the branch-and-bound *BR* in the latest exact algorithm for perfect recall games (Bošanskỳ et al. 2014) is still impractical for the large NEST because its bounds are very loose. More specifically, in their *BR*, the upper bound for each node always equals the maximum possible value of the game, and the lower bound for each succeeding node of the first node in the information set for the searching player always equals the minimum possible value of the game. The immediate result is that at least one succeeding state of each action will be evaluated. However, it is impractical to evaluate at least one succeeding state for each action in each state in NEST because NEST has an extremely large state space and a large action space for each state in the MDP for the defender.

3 Motivating Scenario

To illustrate the underlying issues and motivation for our model, we analyze the problem of police vehicle pursuits. Recently, a new pursuit technology (Fischbach, Hadsdy, and McCal 2015; Gaither et al. 2017) based on GPS has been tested and deployed in many cities, e.g., New York (Shook 2017) in America and Delta (Baker 2017) in Canada. This technology developed by StarChase (see *http://www.starchase.com*) provides police officers with real-time (every 3 to 5 seconds (Gaither et al. 2017)) GPS locations of the fleeing vehicle. However, it does not mean that the evader will always be captured. For example, in the urban scenario discussed in (Gaither et al. 2017), even obtaining the real-time information provided by StarChase GPS-based system, police officers only achieved 61% apprehension rate. Outside the cities, i.e., in the open area, it may be even harder to capture the evader. Cities, however, have more advanced facilities and security resources, which make it much easier to locate the evader's location in real time and capture him. Therefore, we can set the roads to the outside of cities as the exit points. In addition, based on the data in (Reaves 2017), about 10% of 5,568 pursuits ended because the vehicle crossed into another jurisdiction. In such a context, we can set the roads to other jurisdictions as the exit points. Meanwhile, about 50% of cases discussed in (Fischbach, Hadsdy, and McCal 2015) lasted more than 5 minutes, and the maximum time is 50 minutes. Therefore, police officers have limited but sufficient time to deploy security resources to interdict the evader with the aid of the real-time information about the evader.

Our approach is also suitable for security problems in other domains including interdicting poachers in the field, where UAVs can provide poachers' location information in real time (Bondi et al. 2018); and combating the threat of piracy or fighting against illegal fishing on the sea, where a new system can track ships or boats in real time (Reid 2018).

4 Problem Description

The *NEtwork purSuiT game (NEST)* is played between an evader (adversary) and police officers (defender) on an urban road network $G = (V, E)$ consisting of a set of directed edges E representing roads and a set of nodes V representing intersections. W.l.o.g., we assume that the time it takes to travel through each road segment is one time unit, e.g., 1 minute, since we can add dummy nodes to separate each road into several segments with equal length. We denote by V_e the set of exit nodes, through which the adversary can escape to the external world. The adversary, initially located at node v_0^a, tries to escape to the external world, while the defender deploys m identical resources (police officers/teams), initially located at intersections $v_0^1, \ldots, v_0^m \in V$, to catch him. The adversary is caught if he and one of the defender resources reach the same node at the same time[1]. We assume that $v_0^a \neq v_0^r$ for all $r \in \mathcal{R} = \{1, \ldots, m\}$. The adversary cannot see the defender until he gets caught (Jain et al. 2011; Zhang et al. 2017). Motivated by the fact that new pursuit technology can be used to provide real-time information about the adversary's whereabouts, the defender can observe the adversary's locations in real time. The initial positions of both players are common knowledge. For example, police officers are notified about the location of the crime, while the locations of the police stations are public information.

Adversary's Strategies: A pure adversary strategy is a path from his initial node v_0^a to an exit node $v_e \in V_e$, represented by a sequence of nodes o visited by the adversary with a length no longer than $t_{\max}$[2] (time horizon), where any pair of consecutive nodes are connected by an edge in E. The set of adversary pure strategies is denoted by O. The mixed strategy is a probability distribution over O, denoted by $\mathbf{y} = \langle y_o \rangle$ where y_o represents the probability of escaping through path o. An observed history h is a sequence of nodes where the adversary is spotted. We say that h' is a descendant of h or $h \sqsubset h'$ if h is any strict prefix of h'. We denote by $O_h = \{o \mid o \in O, h \sqsubseteq o\}$ the set of paths generating history h, $H = \{h | \exists o \in O, h \sqsubseteq o\}$ the set of valid histories, and $CH_h = \{h' \mid h' \in H, \exists v \in V, h' = h \cdot v :$ v will be reached at the next step after generating history $h\}$ the set of child histories of h.

Defender's Strategies: The defender's decision problem is modeled as an MDP. Let $l = (v_l^1, ..., v_l^m)$ denote the location of m resources, where v_l^r denotes the location of resource r. Let L represent the set of all possible locations. We say two locations l and l' are adjacent if l' can be reached from l with a one-step move. That is, for any $r \in R$, either $v_l^r = v_{l'}^r$ or $(v_l^r, v_{l'}^r) \in E$. Let $l_0 = (v_0^1, \ldots, v_0^m)$ be the initial location of m resources. A pure strategy for the defender is a deterministic policy $\pi : S \to A$. Each state $s \in S$ consists of two components, the current location $l_s \in L$ and the observed history $h_s \in H$ of the adversary's moves from the initial state to state s, i.e., $s = (l_s, h_s)$. Let $\eta : H \to V$ return the last node in history $h \in H$. A state s is called a capture state if there exists a resource $r \in R$ such that $v_{l_s}^r = \eta(h_s)$. We denote by S_c the set of capture states. On the other hand, if $\eta(h_s) \in V_e$ and there exists no resource allocated on $\eta(h_s)$, the adversary successfully escapes, and such a state is called an escape state. Let S_e denote the set of all escape states. Let the terminal state set be $S_t = S_c \cup S_e$ and the initial state be $s_0 = (l_0, \langle v_0^a \rangle)$. Each action $a \in A$ corresponds to a one-step move from the current location l to the adjacent location l'. Thus, we abuse the notation slightly and let A and L be exchangeable, such that each action is a location to which the defender plans to transfer her resources. We denote by A_s the set of actions in $s \in S \backslash S_t$. Specifically, we denote by $S_{s,l}$ the set of states reached from s by taking action $l \in A_s$, i.e., $S_{s,l} = \{s' \mid s' = (l, h), h \in CH_{h_s}\}$, and we say that $s' \in S_{s,l}$ is a succeeding state of s. We denote by $\mathbf{x}$ the behavior strategy of the defender, where in each non-terminal state s, $\mathbf{x}$ defines a probability distribution over A_s.

Utilities: Assume that the game is zero-sum as we are only concerned with whether the defender can capture the adversary or not. If it is a capture state, the defender receives a unit reward, and the adversary suffers a unit loss; otherwise, both players receive a zero payoff. That is, $u_d(s) = -u_a(s) = 1$ if $s \in S_c$ and $u_d(s) = u_a(s) = 0$ if $s \in S_e$. Given strategy profile $(\mathbf{x}, o)$, we denote by $P_{\mathbf{x},o}(s)$ the probability that state s is reached. Notice that: i) if $h_s \not\sqsubseteq o$, i.e., the escaping path o does not generate history h_s, $P_{\mathbf{x},o}(s) = 0$; and ii) otherwise, $P_{\mathbf{x},o}(s)$ only depends on $\mathbf{x}$ and h_s, and is independent of the adversary's moves after h_s. Therefore, we can denote by $P_{\mathbf{x}}(s)$ the probability of reaching state s under $(\mathbf{x}, o)$ with $h_s \sqsubseteq o$. $P_{\mathbf{x}}(s)$ is determined by the following recursive equation:

$$\begin{aligned} &P_{\mathbf{x}}(s_0) = 1, \\ &P_{\mathbf{x}}(s) = \textstyle\sum_{s' \in S \backslash S_t : s \in S_{s', l_s}} P_{\mathbf{x}}(s') x_{s', l_s} \quad \forall s \in S \backslash \{s_0\}, \end{aligned} \tag{1}$$

where the summation is over states that can transit to s.

Given profile $(\mathbf{x}, o)$, the defender's expected utility is: $U_d(\mathbf{x}, o) = \sum_{s \in S_t : h_s \sqsubseteq o} P_{\mathbf{x}}(s) u_d(s) = \sum_{s \in S_c : h_s \sqsubseteq o} P_{\mathbf{x}}(s)$. Accordingly, we have $U_d(\mathbf{x}, \mathbf{y}) = \sum_{o \in O} y_o U_d(\mathbf{x}, o)$. Since the game is zero-sum, $U_a(\mathbf{x}, \mathbf{y}) = -U_d(\mathbf{x}, \mathbf{y})$.

Equilibrium: We use Nash equilibrium (NE) as the solution concept as both players move simultaneously. Generally, $(\mathbf{x}^\star, \mathbf{y}^\star)$ is NE if and only if: (i) $U_d(\mathbf{x}^\star, \mathbf{y}^\star) \geqslant U_d(\mathbf{x}, \mathbf{y}^\star), \forall \mathbf{x}$; and (ii) $U_a(\mathbf{x}^\star, \mathbf{y}^\star) \geqslant U_a(\mathbf{x}^\star, \mathbf{y}), \forall \mathbf{y}$.

5 Theoretical Analysis

This section first shows the cost of ignoring the real-time information, and then proves that solving NEST is NP-hard.

Cost of Ignoring Information: Here, we denote the strategy that does not consider the up to date information about the adversary as the non-real-time strategy and its counterpart as the real-time strategy.

[1] Our model can be easily extended to cover the case that both players meet on an edge by extending this definition of capture.

[2] Assume that after $t_{\max}$, the defender can deploy sufficient police forces (e.g., police from neighbor cities) to prevent the adversary from escaping.

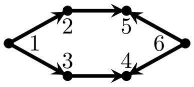


Figure 1: Example

Consider the example in the left figure, where a unique defender resource is at node 6, the adversary is at node 1, and exit nodes are 4 and 5. If the defender ignores the real-time information, in the worst case (under the NE of the zero-sum game), the adversary reaches each exit node with probability 0.5, while the defender's optimal non-real-time strategy moves the single resource to nodes 4 or 5 with equal probability. That is to say, the defender's expected utility is 0.5. However, the defender can ensure the capture of the adversary by doing the following. (i) If the adversary's current location is at the starting node, the defender just stays at node 6. (ii) When the adversary moves to node 2, the defender moves her resource to node 5, and 4 if she observes that the adversary has moved to node 3. As long as the time horizon is finite, the defender will capture the adversary for sure, and the expected utility is 1, much higher than 0.5.

Proposition 1. *The defender's optimal non-real-time strategy under the NE can be arbitrarily worse compared with the optimal real-time strategy under the NE.*[3]

Complexity Analysis

Theorem 1. *Computing the NE of NEST is NP-hard.*

6 Computing the Optimal Solution

This section first transforms the non-convex program of computing the NE to *LP*, then develops our *ISG* algorithm with a novel *BR* oracle, and finally proposes two additional techniques to speed up our *ISG* significantly.

6.1 Transformation to the Linear Program

The NE strategy $\mathbf{x}^\star$ of the defender in NEST can be computed by: $\mathbf{x}^\star \in \arg\max_{\mathbf{x}} \min_{o\in O} U_d(\mathbf{x}, o)$, i.e.,

$$\max_{\mathbf{x}} \quad U_d^\star \tag{2a}$$
$$\text{s.t.} \quad U_d^\star \leqslant U_d(\mathbf{x}, o), \quad \forall o \in O \tag{2b}$$
$$\textstyle\sum_{l\in A_s} x_{s,l} = 1, \quad \forall s \in S\backslash S_t \tag{2c}$$
$$x_{s,l} \in [0,1], \quad \forall l \in A_s, \forall s \in S\backslash S_t \tag{2d}$$

Unfortunately, program (2) is non-convex, since the probability $P_{\mathbf{x}}(s)$ of reaching state s involves multiplication of various entities in $\mathbf{x}$ due to the recursive property defined in Eq.(1). To transform it to an *LP*, based on the propagation of the probability starting from s_0 towards terminal states, we develop a flow representation of the defender's strategy.

Definition 1. $\mathbf{f}$ *is the flow representation of the defender's strategy where* $f_{s,l}$ *specifies the probability that the defender reaches state* $s \in S\backslash S_t$ *and takes the action* $l \in A_s$.

Given $\mathbf{f}$ and o, the defender's expected utility is: $U_d(\mathbf{f}, o) = \sum_{s\in S_c: h_s \sqsubseteq o} \sum_{s'\in S\backslash S_t: s\in S_{s',l_s}} f_{s',l_s}$, where,

$$\textstyle\sum_{l\in A_{s_0}} f_{s_0,l} = 1 \tag{3a}$$
$$\textstyle\sum_{s'\in S\backslash S_t: s\in S_{s',l_s}} f_{s',l_s} = \sum_{l\in A_s} f_{s,l}, \forall s\in S\backslash(\{s_0\}\cup S_t) \tag{3b}$$
$$f_{s,l} \geqslant 0, \quad \forall l \in A_s, \forall s \in S\backslash S_t. \tag{3c}$$

[3] All proofs in this paper are in Section A of the Appendix available at http://www.ntu.edu.sg/home/boan/papers/AAAI19_Urban_Appendix.pdf.

Algorithm 1: *IGRS*

1 Initialize $S' = \{s_0\}$ and $A' = \varnothing$;
2 **repeat**
3 $\quad(\mathbf{x}, \mathbf{y}) \leftarrow$ solution of Problem (6) under (S', A');
4 $\quad(V, \pi) \leftarrow BR(s_0, U_d(\mathbf{x}, \mathbf{y}))$;
5 $\quad$**for** *each* s *reached from* s_0 *by* π *with* $V(s) > 0$ **do**
6 $\quad\quad$**if** $s \notin S'$ **then** $S' \leftarrow S' \cup \{s\}$, $A' \leftarrow A' \cup \{\pi(s)\}$;
7 $\quad\quad$**else**
8 $\quad\quad\quad\vdash$ **if** $\pi(s) \notin A'_s$ **then** $A'_s \leftarrow A'_s \cup \{\pi(s)\}$;
9 **until** $V(s_0) = U_d(\mathbf{x}, \mathbf{y})$;
10 **return** $(\mathbf{x}, \mathbf{y})$.

Eqs.(3a)–(3c) ensure that $\mathbf{f}$ is feasible, which is initialized by Eq.(3a), and Eq.(3b) is the flow conservation equality.

We now show how to derive the associated flow representation from the behavior strategy and vice versa. Given $\mathbf{x}$, its corresponding flow representation $\mathbf{f}$ is:

$$f_{s,l} = P_{\mathbf{x}}(s) x_{s,l}, \quad \forall l \in A_s, \forall s \in S\backslash S_t. \tag{4}$$

Given $\mathbf{f}$, $P_{\mathbf{x}}(s) = \sum_{s'\in S\backslash S_t: s\in S_{s',l_s}} f_{s',l_s}$ if $s \neq s_0$, $P_{\mathbf{x}}(s)=1$ if $s=s_0$, and its corresponding behavior strategy $\mathbf{x}$ is:

$$x_{s,l} = \begin{cases} \frac{f_{s,l}}{P_{\mathbf{x}}(s)} & \text{if } P_{\mathbf{x}}(s) > 0, \\ \frac{1}{|A_s|} & \text{otherwise,} \end{cases} \quad \forall l \in A_s, \forall s \in S\backslash S_t. \tag{5}$$

We prove that the flow representation is equivalent with the behavior strategy in terms of expected utility in Theorem 2.

Theorem 2. *For any pair of* $\mathbf{x}$ *and* $\mathbf{f}$ *satisfying Eqs.*(4) *and* (5)*,* $U_d(\mathbf{x}, o) = U_d(\mathbf{f}, o)$ $\forall o \in O$.

Finally, we obtain the following *LP* equivalent with (2).

$$\max_{\mathbf{f}} \quad U_d^\star \tag{6a}$$
$$\text{s.t. } U_d^\star \leqslant U_d(\mathbf{f}, o), \quad \forall o \in O \tag{6b}$$
$$\text{Eqs.}(3a) - (3c). \tag{6c}$$

6.2 Incremental Strategy Generation

Although we can compute the optimal solution with *LP* (6), it does not scale up due to the exponentially large state space. To mitigate this issue, we propose the **I**teratively **G**enerated **R**eachable **S**tates (*IGRS*) algorithm shown in Algorithm 1, which works as follows. Starting from the restricted NEST $G(S', A')$ where $S' \subset S$ and $A' \subset A$, *LP* (6) solves the equilibrium $(\mathbf{x}, \mathbf{y})$ on $G(S', A')$ (Line 3).[4] We then solve the defender's best response policy π in the original strategy space (S, A) (Line 4) and expand A' and S' (Lines 5–8).

Defender's Best Response: Our **B**est-**R**esponse (*BR*) algorithm shown in Algorithm 2 computes the best deterministic π against the adversary's strategy $\mathbf{y}$. Due to the large strategy space, to speed up, *BR* adopts a branch-and-bound approach in a depth-first manner. A branch is a state with an expected value representing the probability of catching the adversary starting from this branch. The lower bound of a branch is the minimum value that this branch must obtain in order to be part of the best response, while its upper bound is

[4] Note that, at initialization (Line 1), s_0 is temporarily treated as an escape state in order to compute a solution $(\mathbf{x}, \mathbf{y})$.

Algorithm 2: *BR*(s,b_s): s—current state, b_s—lower bound

1 **if** *s is visited and* $b_s \geqslant b(s)$ **then return** $V(s)$;
2 **if** $s \in S_t$ **then** $V(s) \leftarrow u_d(s) \times \sum_{o \in O_{h_s}} y_o$, **return** $V(s)$;
3 **for** $r \in \mathcal{R}$ **do**
4 **for** $l_r \in A_s(r), o \in O_{h_s}(y_o > 0)$ **do**
5 **if** $dist(l_r, V_e \cap o) + |h_s| + 1 \leqslant |o|$ **then**
6 $O_{r,l_r} \leftarrow O_{r,l_r} \cup \{o\}$, $A'_s(r) \leftarrow A'_s(r) \cup \{l_r\}$;
7 **if** $A'_s(r) = \varnothing$ **then** $A'_s(r) \leftarrow \{v^r_{l_s}\}$;
8 $A'_s \leftarrow \times_{r \in \mathcal{R}} A'_s(r)$: the best joint action candidate set;
9 **for** $l \in A'_s$ **do** $O_l \leftarrow \cup_{r \in \mathcal{R}} O_{r,l_r}$;
10 sort $l \in A'_s$ descending according to value $B_l \leftarrow \sum_{o \in O_l} y_o$;
11 $V(s) \leftarrow \max\{b_s - 2 \times \epsilon, 0\}, b(s) \leftarrow -\infty, Q_l \leftarrow 0 (\forall l \in A'_s)$;
12 **for** $l \in A'_s$ **do**
13 **if** $V(s) + \epsilon \leqslant B_l$ **then**
14 sort (l,h) ($h \in CH_{h_s}$) descending according to value $B_{(l,h)} \leftarrow \sum_{o \in O_h \cap O_l} y_o$, $B' \leftarrow \sum_{h \in CH_{h_s}} B_{(l,h)}$;
15 **for** $h \in CH_{h_s}, B_{(l,h)} > 0$ **do**
16 $b_{(l,h)} \leftarrow V(s) + \epsilon - (Q_l + (B' - B_{(l,h)}))$;
17 **if** $b_{(l,h)} > B_{(l,h)}$ **then break**;
18 **else**
19 $V(l,h) \leftarrow$*BR*$((l,h), b_{(l,h)})$;
20 **if** $b_{(l,h)} > V(l,h)$ **then break**;
21 **else** $Q_l \leftarrow Q_l + V(l,h)$, $B' \leftarrow B' - B_{(l,h)}$;
22 **if** $V(s) < Q_l$ **then** $V(s) \leftarrow Q_l, \pi(s) \leftarrow l$;
23 **if** $V(s) < b_s$ **then** $b(s) \leftarrow b_s$;
24 **return** $V(s)$.

the maximum value. Using these bounds, *BR* cuts branches which will certainly not be part of the best response.

BR works as follows. Given a state s, if *BR* does not terminate at Lines 1 and 2, *BR* estimates bounds and cuts branches if: (1) the upper bound B_l of action l is less than its lower bound ($V(s) + \epsilon$) (Line 13); (2) the upper bound $B_{(l,h)}$ of succeeding state (l,h) is not more than 0 (Line 15) or is less than its lower bound $b_{(l,h)}$ (Line 17); or (3) the state value $V(l,h)$ is less than $b_{(l,h)}$ (Line 20). Then, *BR* updates action value Q_l and returns state value $V(s) = \max_l Q_l$ (Lines 21–24).

Specifically, *BR* estimates tight bounds as follows. Firstly, given the combinatorial action space, *BR* estimates tight upper bounds (Lines 3–10 and 14). The idea is that, given a joint action l, *BR* can estimate all the possible paths that the defender has a chance to interdict (i.e., the possibly interdicted path set (PIPS)) by taking l. That is, given a resource r's action l_r in l and an adversary path $o \in O_{h_s}$, *BR* can determine if r taking l_r has a chance to interdict o by comparing the time from its current adversary location $\eta(h_s)$ to exit node $v_e \in o$ through o and the shortest time from action (location) l_r to v_e (Line 5). Thus, *BR* obtains l_r's PIPS (O_{r,l_r} (Line 6) initialized as $\varnothing$ with the corresponding best action candidate set $A'_s(r)$), l's PIPS (O_l (Line 9)), and (l,h)'s PIPS ($O_l \cap O_h$ (Line 14)). Secondly, to be the best action, an action's value must be larger than the obtained maximum action value, and then the lower bounds for the remaining actions and succeeding states will adaptively increase. Specifically, $V(s)$ is initialized at Line 11 to guarantee that the best action is computed if s's optimal value $V^\star(s) = b_s$; and $b_{l,h}$ is estimated by assuming that all the remaining succeeding states will return the maximum value (Line 16).

In addition, *BR* effectively handles imperfect recall of NEST where a state may be visited by different prefix states. To avoid repeated computation, *BR* stores the computed state value for state s as $V(s)$ that is immediately returned if s is visited again (Line 1). However, while setting tight bounds, only using this method may cause some wrong cuts. Here, if b_s (that is set for s while being visited by its prefix state s_1) is tight, and $V(s)$ is computed such that $V(s) < b_s$, then $V(s)$ may not be optimal because some cuts may happen. If $V(s)$ is not optimal (i.e., $V(s) < V^\star(s) < b_s$) and s is revisited by another prefix state s_2 with b'_s such that $V(s) < b'_s < V^\star(s)$, then the wrong cut will happen in s_2 after $V(s)$ is returned to s_2. To avoid these wrong cuts, *BR* records the computed state value with the corresponding lower bound (recorded as $b(s)$) if this value is less than this bound (Line 23), and this value will be recomputed only if this state's prefix state with a smaller lower bound transits to it (Line 1).

Theorem 3. IGRS *converges to an NE.*

6.3 Improving the Scalability of *IGRS*

Even though our *IGRS* is far more efficient than *LP*, it still cannot handle the very large NEST. Thus, we further improve the scalability of *IGRS* by developing two techniques[5].

The first technique is to map the computed defender strategy by *BR* in a subgame to its similar subgames to speed up *BR*. Here, NEST is the full game, and part of NEST is the subgame. Moreover, if this computed strategy is optimal in NEST (i.e., part of the equilibrium in NEST), we can also speed up *LP* and map this strategy to more subgames.

Formally, subgame G_s has initial state s, where the defender's strategy space (S_{G_s}, A_{G_s}) includes all states reachable from s (i.e., if s_1 in S_{G_s} transits to s_2, then $s_2 \in S_{G_s}$) with the corresponding action space, and the adversary's strategy space includes all paths generating h_s, i.e., O_{h_s}.[6] Here, the defender has the same action space in states with the same location for her. Given states s_1 and s_2, $O_{h_{s_1}}$ is similar to $O_{h_{s_2}}$ after $\eta(h_{s_1})$ if $\eta(h_{s_1}) = \eta(h_{s_2})$, and a one-to-one correspondence exists between $O_{h_{s_1}}$ and $O_{h_{s_2}}$ such that each such pair $o_1 \in O_{h_{s_1}}$ and $o_2 \in O_{h_{s_2}}$ satisfies $q(h_{s_1}, o_1) = q(h_{s_2}, o_2)$ ($q(h_1, h_2)$ is a subsequence of h_2 after $\eta(h_1)$ such that $h_1 \cup q(h_1, h_2) = h_2$). That is, the adversary has the same move space after $\eta(h_{s_1})$ by both sets.

Two subgames G_{s_1} and G_{s_2} are similar if $l_{s_1} = l_{s_2}$, and $O_{h_{s_1}}$ is similar to $O_{h_{s_2}}$ after $\eta(h_{s_1})$. Then, states $s_i \in S_{s_1}$ and $s_j \in S_{s_2}$ are similar if $l_{s_i} = l_{s_j}$ and $q(h_{s_1}, h_{s_i}) = q(h_{s_2}, h_{s_j})$. The best response strategy π_{s_1} in G_{s_1} against $\mathbf{y}$ is defined by: Given the best response π against $\mathbf{y}$ by *BR*,

$$\pi_{s_1}(s') = \pi(s')(\forall s' \in S_{G_{s_1}}). \tag{7}$$

Here, for $s' \in S_{G_{s_1}}$, if s' is reached from s_1 by π_{s_1}, we define $P_{\pi_{s_1}}(s') = 1$; otherwise, $P_{\pi_{s_1}}(s') = 0$. Therefore,

[5]We illustrate both techniques in Section B of the Appendix

[6]We define the subgame like this because we only compute the defender's strategies in subgames through *BR* given $\mathbf{y}$ over O.

a mapping strategy $\pi_{s_1 \to s_2}$ from G_{s_1} to its similar subgame G_{s_2} is defined by: Given π_{s_1} defined by Eq.(7), $\forall s_j \in S_{G_{s_2}}$,

$$\pi_{s_1 \to s_2}(s_j) = \pi_{s_1}(s_i), \tag{8}$$

where s_i in $S_{G_{s_1}}$ and s_j are similar. We use this mapping only if the adversary takes both paths at each pair of the one-to-one correspondence between $O_{h_{s_1}}$ and $O_{h_{s_2}}$ with the same probability. This mapping could happen at the iteration when π_{s_1} is computed during calling *BR* or at the future iterations due to the loop in *IGRS*.

Theorem 4. *For any G_{s_1} with π_{s_1} defined by Eq.(7) as the best response in G_{s_1} against $\mathbf{y}$, if G_{s_2} is its similar subgame with $\mathbf{y}'$ satisfying $y_{o_1} = y'_{o_2}$ for each pair $o_1 \in O_{h_{s_1}}$ and $o_2 \in O_{h_{s_2}}$ with $q(h_{s_1}, o_1) = q(h_{s_2}, o_2)$ in the one-to-one correspondence between $O_{h_{s_1}}$ and $O_{h_{s_2}}$, then $\pi_{s_1 \to s_2}$ defined by Eq.(8) is the best response in G_{s_2} against $\mathbf{y}'$.*

A defender strategy $\pi^\star_s$ in G_s is optimal in NEST if given any strategy $\mathbf{y}$ over O and any policy π_s in G_s, $V^{\pi_s}(s) \leqslant V^{\pi^\star_s}(s)$, and we say that this $\pi^\star_s$ is part of the equilibrium in NEST. $\pi^\star_s$ is defined by: Given π_s defined by Eq.(7) as the best response in G_s against some $\mathbf{y}$ over O with $y_o > 0$ ($\forall o \in O_{h_s}$) such that $V^{\pi_s}(s) = \sum_{o \in O_{h_s}} y_o$,

$$\pi^\star_s(s') = \pi_s(s') (\forall s' \in S_{G_s}). \tag{9}$$

Lemma 1. *Given any $\mathbf{y}$ over O, $V^{\pi^\star_s}(s) = \sum_{o \in O_{h_s}} y_o$ under $\pi^\star_s$ defined by Eq.(9).*

Theorem 5. *$\pi^\star_s$ defined by Eq.(9) is optimal in NEST.*[7]

Given $\pi^\star_s$ defined by Eq.(9), the defender will certainly catch the adversary in G_s. Therefore, after computing $\pi^\star_s$ in G_s, if G_s is reached again, the defender's expected utility $\sum_{o \in O_{h_s}} y_o$ is returned immediately. Therefore, *LP* for the restricted game can exclude constraints and variables for the succeeding states of s; and, for any adversary strategy, *BR* does not need to compute the best response starting from s anymore, which will speed up *LP* and *BR*.

We can map $\pi^\star_{s_1}$ defined by Eq.(9) in G_{s_1} to its similar subgames through Eq.(8). Besides, we can map this $\pi^\star_{s_1}$ to more subgames, where the initial location of m resources is not l_{s_1}. In fact, given $\pi^\star_{s_1}$, we can trace the interdiction back to which resource r starting from s_1 contributes to this interdiction by following $\pi^\star_{s_1}$, and we may find that we only need part of m resources to interdict the adversary. Then, we only need to map the actions of these key resources to other subgames to capture the adversary certainly. For example, consider G_{s_1} and G_{s_2}, where $l_{s_1} = (v_1, v_2)$, $l_{s_2} = (v_1, v_3)$, $O_{h_{s_1}}$ is similar to $O_{h_{s_2}}$ after $\eta(h_{s_1})$, and only the first resource in G_{s_1} will contribute to the interdiction by $\pi^\star_{s_1}$. Then, the adversary will be captured certainly in G_{s_2} if the resource initially at v_1 in G_{s_2} takes the action of the resource initially at v_1 in G_{s_1} by $\pi^\star_{s_1}$.

Formally, we define this strategy mapping between semi-similar subgames. Two subgames G_{s_1} and G_{s_2} are semi-similar on $\bar{l}$ if $\bar{l} \subseteq l_{s_1}$, $\bar{l} \subseteq l_{s_2}$, and $O_{h_{s_1}}$ is similar to $O_{h_{s_2}}$ after $\eta(h_{s_1})$. Similar subgames are certainly semi-similar.[8] Here, $s_i \in S_{s_1}$ and $s_j \in S_{s_2}$ are semi-similar if $\underline{l}$, with $\underline{l} \subseteq l_{s_i}$ and $\underline{l} \subseteq l_{s_j}$, is reached from $\bar{l}$ by the corresponding resources in s_1 and s_2, respectively, and $q(h_{s_1}, h_{s_i}) = q(h_{s_2}, h_{s_j})$.

Given $\pi^\star_{s_1}$ defined by Eq.(9), if resource r starting from s_1 by $\pi^\star_{s_1}$ can interdict at least one of the paths in O_{s_1} (i.e., there exists $o \in O_{h_{s_1}}$ such that there is a capture state s_c with $P_{\pi^\star_{s_1}}(s_c) = 1$, $h_{s_c} \sqsubseteq o$, and $v^r_{l_{s_c}} = \eta(h_{s_c})$), we call r's location $v^r_{l_{s_1}}$ in s_1 as the key location. Let $l^*_{s_1} \subseteq l_{s_1}$ be the set of these key locations in s_1. If G_{s_2} and G_{s_1} are semi-similar on $l^*_{s_1}$ with $\pi^\star_{s_1}$ defined by Eq.(9) in G_{s_1}, then its mapping strategy $\pi^\star_{s_1 \to s_2}$ in G_{s_2} is defined by: $\forall s_j \in S_{G_{s_2}}$,

$$\pi^\star_{s_1 \to s_2}(s_j) = (\cdots, l_r, \cdots), \tag{10}$$

where if $v^r_{l_{s_2}} \in l^*_{s_1}$, then $l_r = \pi^\star_{s_1}(s_i)_{r*, v^{r*}_{l_{s_1}} = v^r_{l_{s_2}}}$ (s_i in $S_{G_{s_1}}$ is semi-similar to s_j with $P_{\pi^\star_{s_1}}(s_i) = 1$) which means that resource r in s_j follows $\pi^\star_{s_1}(s_i)$ for resource $r*$ in s_i whose location $v^{r*}_{l_{s_1}}$ in s_1 overlaps r's location $v^r_{l_{s_2}}$ in s_2; otherwise, $l_r = v^r_{l_{s_j}}$, i.e., staying at the current node.

Theorem 6. *$\pi^\star_{s_1 \to s_2}$ defined by Eq.(10) is optimal in NEST.*

The second technique is to add multiple defender best response strategies at one iteration to reduce the number of iterations for the convergence of *IGRS*. More specifically, we sample multiple adversary strategies to add the corresponding defender's best response strategies to the restricted game $G(S', A')$ by calling *BR* in *IGRS* at one iteration.

We sample a uniform strategy over O_{h_s} for each state s in $G(S', A')$ based on the following intuition. On the one hand, for different adversary strategies, the corresponding best response strategies involve different states and actions that will be added to $G(S', A')$ before the convergence of *IGRS*, and the number of iterations will be enormous if the strategy space is large. At each iteration, *LP* will be solved once to generate the adversary equilibrium strategy in $G(S', A')$. If we can sample some of these strategies, then we can reduce the number of iterations of *IGRS* and times to solve *LP*. On the other hand, if s is added to $G(S', A')$ (i.e., s is part of the best response), at least one of the paths in O_{h_s} is in the support set of the equilibrium strategy in $G(S', A')$. Then, it is possible that the adversary will escape through the paths in O_{h_s}, and the defender needs a corresponding strategy to interdict him. To generate such a strategy immediately, we can sample a uniform strategy over O_{h_s} and then call *BR*.

Finally, *IGRS* embedded with the above two additional techniques (*IGRS++*) is still optimal.

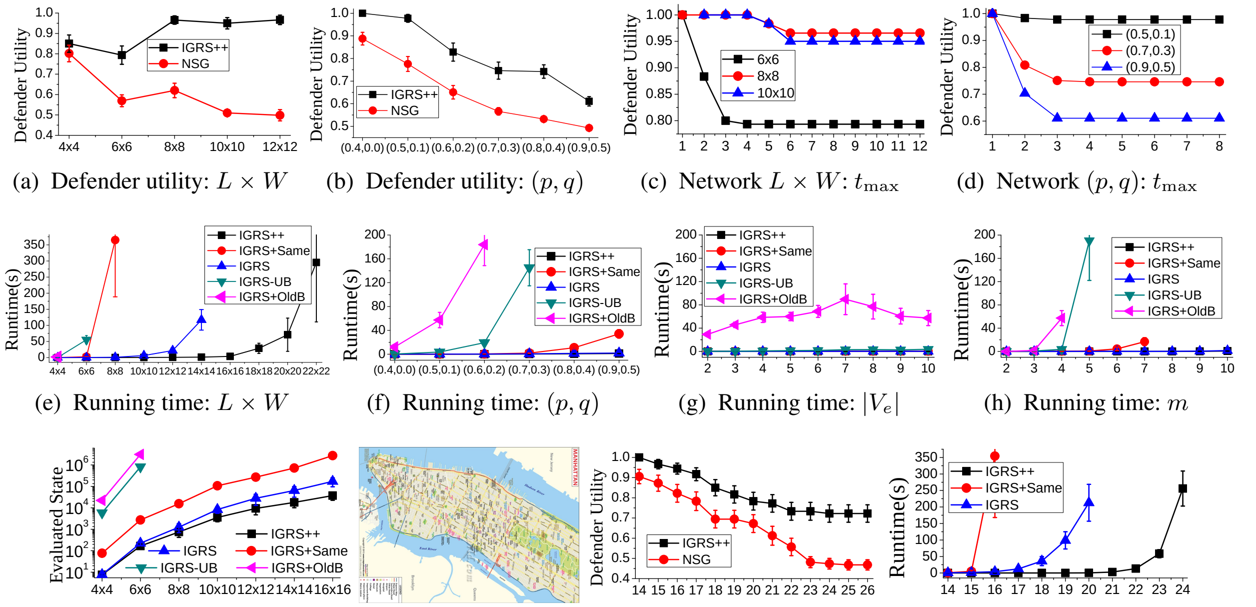
Theorem 7. IGRS++ *converges to an NE.*

7 Experimental Evaluation

We demonstrate the efficiency of our algorithms through numerical experiments. We use CPLEX (version 12.6) to solve the linear program. All computations are performed on a

[7]Note that, for games with imperfect information, the general subgame solving technique cannot guarantee optimality (Brown and Sandholm 2017).

[8]We still need to consider similar subgames because we cannot have the property similar to Theorem 4 between semi-similar subgames (see Section C of the Appendix).



(a) Defender utility: $L \times W$ (b) Defender utility: (p, q) (c) Network $L \times W$: $t_{\max}$ (d) Network (p, q): $t_{\max}$

(e) Running time: $L \times W$ (f) Running time: (p, q) (g) Running time: $|V_e|$ (h) Running time: m

(i) Evaluated states: $L \times W$ (j) Manhattan (k) Defender Utility: $t_{\max}$ (l) Running time: $t_{\max}$

Figure 2: Solution quality: (a)–(b); Influence of $t_{\max}$: (c)–(d); Scalability: (e)–(i) (log y-axis in (i)); Real network: (j)–(l).

machine with a 3.2GHz quad core CPU and 16GB memory. All results are averaged over 30 randomly generated instances. All random planar graphs are generated by the grid model with random edges (Peng et al. 2014), which samples an $L \times W$ square grid where horizontal/vertical edges between neighbors are generated with probability p, and diagonal ones with q. We choose p from $[0.4, 0.9]$ and q from $[0.0, 0.5]$ to match most real-world road networks. In our evaluation, v_0^a is located at the center node, m defender resources are uniformly distributed on the network at initialization, and $|V_e|$ exit nodes are randomly distributed at the border. By default, $L = W = 5$, $(p, q) = (0.5, 0.1)$, horizon $t_{\max} = L$, $|V_e| = 10$, and $m = 4$.

Solution Quality: We compare the solution quality of our approach (represented by *IGRS++*) with one baseline, the algorithm for the discussed network security game model (Jain et al. 2011; Zhang et al. 2017) considering time-dependent strategies, denoted as *NSG*. Figures 2(a) and 2(b) vary parameters: (1) the number of intersections, denoted by $L \times W$, and (2) edge generation probabilities. It can be seen that *IGRS++* significantly outperforms *NSG*. These results are similar to ones for varying the number of exit nodes or resources (not shown here). Here the defender's utility does not decrease with the size of the network because although the adversary has more strategies on larger networks, the defender can exploit more real-time information about him.

We evaluate the influence of the horizon $t_{\max}$ on the solution quality on networks with different scales in Figure 2(c) and networks with different density of edges in Figure 2(d). Results show that the defender's expected utility converges when $t_{\max}$ is large enough, e.g., $t_{\max}$ is almost equal to L.

Scalability: We evaluate the scalability of *IGRS* with two baselines: (1) using the bounds setting in (Bošanskỳ et al. 2014) for *IGRS* (*IGRS+OldB*), and (2) only using our new lower bound setting without the tight upper bound for *IGRS* (*IGRS-UB*). Specifically, *IGRS+OldB* can cut some branches only when the searching player's information set has at least two nodes. To be fair, each state s in *IGRS+OldB* is treated as a defender's information set, and each child history of h_s corresponds to one node in this information set. In addition, to evaluate our second technique at Section 6.3, we have a baseline, *IGRS+Same*, where *IGRS++* uses the method that adds several best response strategies against the same adversary strategy based on (Wang, Yin, and An 2016).

Results in Figures 2(e)–2(h) show that: 1) The runtime generally increases except that it reflects the "easy-hard-easy" pattern in security games (Jain, Leyton-Brown, and Tambe 2012) in Figure 2(g). 2) *IGRS++* significantly outperforms *IGRS* and *IGRS+Same*, especially when the size of the problem is extremely large. 3) *IGRS* significantly outperforms *IGRS-UB*, and *IGRS-UB* significantly outperforms *IGRS+OldB*. To further evaluate the effect of our pruning techniques, Figure 2(i) shows the number of evaluated states for computing the best response against the adversary's uniform strategy, i.e., $y_o = 1/|O| (\forall o \in O)$, where results are consistent with the ones in Figures 2(e)–2(h). Here, *IGRS+Same* runs slower than *IGRS* because its *BR* evaluates too many states to compute several best response strategies, and it adds too many strategies to the restricted game.

The Real-World Network: We also evaluate our approach on the road network of the whole Manhattan island of New York as shown in Figure 2(j). In this network, there are 702 nodes with 1,216 links extracted from OSM (*www.openstreetmap.org*). The initial positions of four police officers and the adversary are chosen randomly, and seven exit nodes are chosen based on the connection to the external world of the island. A total of 30 cases with different adversary initial positions are tested. As shown in Figures 2(k) and 2(l) (note that *IGRS-UB* and *IGRS+OldB* cannot run on networks of this size), results are consistent with ones in Figures 2(a)–2(i) on random networks. In particular, *IGRS++* significantly outperforms others and efficiently solves NEST until the defender's expected utility converges.

8 Conclusions

This paper studies the problem of optimal interdiction of urban criminals with the aid of real-time information. We show that ignoring such information can cause an arbitrarily large loss and then propose our NEST. We show that solving NEST is NP-hard, therefore develop an optimal algorithm *IGRS++* to solve it efficiently. Extensive experiments show the effectiveness of our approach in a wide range of settings.

Acknowledgments

This research is supported by NRF2015 NCR-NCR003-004. Long Tran-Thanh was supported by the EPSRC funded project EP/N02026X/1.

References

Baker, P. 2017. Delta Police deploy new tool to help curb dangerous car chases. *Global News. http://globalnews.ca/news/3178224/delta-police-deploy-new-tool-to-help-curb-dangerous-car-chases/.*

Basilico, N.; Celli, A.; De Nittis, G.; and Gatti, N. 2017. Coordinating multiple defensive resources in patrolling games with alarm systems. In *AAMAS*, 678–686.

Basilico, N.; De Nittis, G.; and Gatti, N. 2017. Adversarial patrolling with spatially uncertain alarm signals. *Artificial Intelligence* 246:220–257.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, 57–64.

Bondi, E.; Fang, F.; Hamilton, M.; Kar, D.; Dmello, D.; Choi, J.; Hannaford, R.; Iyer, A.; Joppa, L.; Tambe, M.; et al. 2018. SPOT poachers in action: Augmenting conservation drones with automatic detection in near real time. In *IAAI*.

Bošanskỳ, B.; Kiekintveld, C.; Lisỳ, V.; and Pěchouček, M. 2014. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence Research* 51:829–866.

Bošanskỳ, B.; Jiang, A. X.; Tambe, M.; and Kiekintveld, C. 2015. Combining compact representation and incremental generation in large games with sequential strategies. In *AAAI*, 812–818.

Brown, N., and Sandholm, T. 2017. Safe and nested subgame solving for imperfect-information games. In *NIPS*, 689–699.

Čermák, J.; Bošanskỳ, B.; Horák, K.; Lisỳ, V.; and Pěchouček, M. 2018. Approximating maxmin strategies in imperfect recall games using A-loss recall property. *International Journal of Approximate Reasoning* 93:290–326.

FBI. 2017. *Bank crime statistics 2016*. https://www.fbi.gov/file-repository/bank-crime-statistics-2016.pdf/view.

Fischbach, T. A.; Hadsdy, K.; and McCal, A. 2015. *Pursuit management: Fleeing vehicle tagging and tracking technology*. Final Report to the U.S. Department of Justice.

Gaither, M.; Gabriele, M.; Andersen, N.; Healy, S.; and Hung, V. 2017. *Pursuit technology impact assessment, Version 1.1*. Final Report to the U.S. Department of Justice.

Horák, K., and Bošanskỳ, B. 2016. A point-based approximate algorithm for one-sided partially observable pursuit-evasion games. In *GameSec*, 435–454.

Horák, K.; Bosanskỳ, B.; and Pechoucek, M. 2017. Heuristic search value iteration for one-sided partially observable stochastic games. In *AAAI*, 558–564.

Jain, M.; Korzhyk, D.; Vaněk, O.; Conitzer, V.; Pěchouček, M.; and Tambe, M. 2011. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, 327–334.

Jain, M.; Leyton-Brown, K.; and Tambe, M. 2012. The deployment-to-saturation ratio in security games. In *AAAI*, 1362–1370.

Jiang, A. X.; Yin, Z.; Zhang, C.; Tambe, M.; and Kraus, S. 2013. Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *AAMAS*, 207–214.

McCarthy, S. M.; Tambe, M.; Kiekintveld, C.; Gore, M. L.; and Killion, A. 2016. Preventing illegal logging: Simultaneous optimization of resource teams and tactics for security. In *AAAI*, 3880–3886.

NCVRW. 2016. *2016 NCVRW resource guide: Urban and rural crime fact sheet*. https://ovc.ncjrs.gov/ncvrw2016/content/section-6/PDF/2016NCVRW_6_UrbanRural-508.pdf.

Parsons, T. D. 1978. Pursuit-evasion in a graph. In *Theory and Applications of Graphs*. Springer. 426–441.

Peng, W.; Dong, G.; Yang, K.; and Su, J. 2014. 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.

Reaves, B. A. 2017. *Police vehicle pursuits, 2012-2013*. U.S. Department of Justice, Office of Justice Programs.

Reid, D. 2018. A real-time satellite-based surveillance of ships has gone live. *CNBC. https://www.cnbc.com/2018/07/19/leonardos-satellite-based-surveillance-of-ships-has-gone-live.html*.

Shook, A. 2017. Tracking "darts" take aim at police pursuits. *WWLP. http://wwlp.com/2017/03/22/tracking-darts-take-aim-at-police-pursuits/.*

Von Stengel, B. 1996. Efficient computation of behavior strategies. *Games and Economic Behavior* 14(2):220–246.

Vorobeychik, Y.; An, B.; Tambe, M.; and Singh, S. P. 2014. Computing solutions in infinite-horizon discounted adversarial patrolling games. In *ICAPS*, 314–322.

Wang, X.; An, B.; Strobel, M.; and Kong, F. 2018. Catching Captain Jack: Efficient time and space dependent patrols to combat oil-siphoning in international waters. In *AAAI*, 208–215.

Wang, Z.; Yin, Y.; and An, B. 2016. Computing optimal monitoring strategy for detecting terrorist plots. In *AAAI*, 637–643.

Zhang, Y.; An, B.; Tran-Thanh, L.; Wang, Z.; Gan, J.; and Jennings, N. R. 2017. Optimal escape interdiction on transportation networks. In *IJCAI*, 3936–3944.