Optimal Escape Interdiction on Transportation Networks

Youzhi Zhang[1], **Bo An**[1], **Long Tran-Thanh**[2], **Zhen Wang**[3], **Jiarui Gan**[4], **Nicholas R. Jennings**[5]
[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]Department of Electronics and Computer Science, University of Southampton, UK
[3]School of Cyberspace, Hangzhou Dianzi University, China
[4]Department of Computer Science, University of Oxford, UK
[5]Departments of Computing and Electrical and Electronic Engineering, Imperial College, UK
[1]{yzhang137,boan}@ntu.edu.sg, [2]ltt08r@ecs.soton.ac.uk, [3]wangzhen@hdu.edu.cn, [4]jiarui.gan@cs.ox.ac.uk, [5]n.jennings@imperial.ac.uk

Abstract

Preventing crimes or terrorist attacks in urban areas is challenging. Law enforcement officers need to respond quickly to catch the attacker on his escape route, which is subject to time-dependent traffic conditions on transportation networks. The attacker can strategically choose his escape path and driving speed to avoid being captured. Existing work on security resource allocation has not considered such scenarios with time-dependent strategies for both players. Therefore, in this paper, we study the problem of efficiently scheduling security resources for interdicting the escaping attacker. We propose: 1) a new defender-attacker security game model for escape interdiction on transportation networks; and 2) an efficient double oracle algorithm to compute the optimal defender strategy, which combines mixed-integer linear programming formulations for best response problems and effective approximation algorithms for improving the scalability of the algorithms. Experimental evaluation shows that our approach significantly outperforms baselines in solution quality and scales up to realistic-sized transportation networks with hundreds of intersections.

1 Introduction

Preventing crimes or terrorist attacks in urban areas is challenging, since the number of potential targets in cities and large towns is huge. For example, the large number of bank branches, especially those offering many escape routes, could be highly profitable targets for bank robbers. In case of such an event, e.g., bank robbery or terrorist attack on buildings, it is critical for police officers to respond quickly and catch the attacker on his escape route. In this paper, we aim to help law enforcement agencies (defender) with efficiently scheduling security resources to interdict the escaping attacker.

It is a significant challenge to develop an effective response plan given the limited security resources and the attacker's strategic actions. Given the initial locations of the security resources, the defender needs to consider traffic-dependent travel time in planning the schedules. In addition, one defender resource may capture the attacker at different intersections on his possible escape routes. Therefore, to make the best use of her limited resources, the defender needs to dynamically relocate them among potential intersections during the event. Moreover, the attacker may strategically choose his escape path and dynamically change his driving speed on different roads along the path to avoid being captured, which makes the attacker's strategy space continuous. This type of problems is typically referred to as pursuit-evasion games (PEGs) within the game theory literature, where the evader tries to minimize the probability of encountering the pursuer but the pursuer wants to thwart the evader's plans by capturing him [Adler *et al.*, 2002; Nowakowski and Winkler, 1983]. However, PEGs do not consider realistic traffic dynamics, where the travel time is influenced by the traffic flow. Furthermore, the evader in PEGs cannot escape to the external world, thus the goal of the pursuer is to minimize the number of rounds needed to catch the evader. In contrast, our domain consists of a defender who aims to maximize the probability of capturing the attacker before he escapes. While there has been a variety of research work on applying game theoretic approaches to security domains [Tambe, 2011; Shieh *et al.*, 2012; Letchford and Conitzer, 2013; Blum *et al.*, 2014; Vorobeychik *et al.*, 2014; Yin *et al.*, 2014; 2015; Zhao *et al.*, 2016; Guo *et al.*, 2016], most of these unrealistically assume that at most one player takes paths [Basilico *et al.*, 2009; Fang *et al.*, 2016], and the time dynamics are irrelevant [Tsai *et al.*, 2010; Jain *et al.*, 2013]. As such, these models are not suitable for our problem. In particular, the attacker can exploit the fact that traffic is slower in some places, and then can choose a path, which could have been covered by the police in current models with no time dynamics, but not in our setting.

To fill this gap, we introduce a novel Escape Interdiction Game (EIG) to model time-dependent strategies for both players, where the defender chooses a sequence of intersections for each police officer to protect, while the attacker chooses an escape path and his travel time on different edges along the path. However, due to these extensions, both the defender strategy space and the continuous attacker strategy space suffer an exponential growth. These lead to major challenges in the computation of the optimal solution, as we will prove this problem is NP-hard. In particular, state-of-the-art security game solutions, such as the current double oracle type algorithms [Jain *et al.*, 2013;

Wang *et al.*, 2016], can only be applied to our problem at very small scales. To overcome this computational challenge, we first propose our solution, *EIGS*, which incorporates the following key components: 1) a new double oracle structure to avoid calling the hard oracle, i.e., the best response attacker oracle, frequently; 2) novel mixed-integer linear programming (MILP) formulations to compute the best response strategies for both players; and 3) a greedy algorithm and an efficient modified Dijkstra algorithm to efficiently generate improving strategies. To further improve the scalability to solve large-scale scenarios with hundreds of intersections, we introduce a novel defender algorithm *RedAD* that reduces the strategy space by deploying each defender resource to protect only one node, assuming that the attacker travels with maximum speed, and using a novel algorithm to contract the graph before using the MILP to compute the attacker response strategy. We conduct extensive experiments showing that *EIGS* can efficiently obtain a robust solution significantly outperforming existing approaches in problems of related small scales and *RedAD* can scale up to realistic-sized transportation networks of hundreds of intersections with good solution quality.

2 Modelling Traffic Network

In this section, we introduce a realistic macroscopic traffic model to illustrate how the traffic flow influences the travel time in a transportation network [Daganzo, 1994; 1995; Skabardonis and Dowling, 1997; Coogan and Arcak, 2015].

Network Structure: The urban road network is modeled as a directed graph $G = (V, E)$ consisting of a set of directed links E to represent the roads, and a set of nodes V to represent the intersections. For each link $e = (\sigma_e, \tau_e)$, the traffic flows from its start node σ_e to its end node τ_e. An extra node $v_\infty \in V$ represents the external world, which is both the source of the flow entering the network and the sink of flow exiting it. In particular, the sets of entry and exit links are denoted by $E_{entry} = \{e|\sigma_e = v_\infty\}$ and $E_{exit} = \{e|\tau_e = v_\infty\}$, respectively. Accordingly, the remaining subset of the links is the set of internal links $E_{inter} = \{e|\sigma_e, \tau_e \in V \setminus \{v_\infty\}\}$. It assumes that there are no self-loops, i.e., $\sigma_e \neq \tau_e$ for each link $e \in E$. For each link $e \in E$, T_e is the minimal travel time when the road is empty, C_e is the link capacity, and the sets of upstream links and downstream links are denoted by $E_e^- = \{e'|\tau_{e'} = \sigma_e\}$ and $E_e^+ = \{e'|\sigma_{e'} = \tau_e\}$, respectively.

Traffic Flow: Each entry link $e \in E_{entry}$ is associated with a constant traffic demand $d_e \geq 0$ modelling the rate of vehicles entering the node τ_e from the external world[1]. Traffic flows among consecutive links according to a static turning preference matrix $R \in \mathbb{R}_+^{E\times E}$ whose entry $R_{ee'}$ represents the fraction of vehicles flowing out of link e that are routed to link e'. These turning ratios model the route choice behavior of drivers, justified by empirical observations [Lebacque, 2005]. Conservation of mass implies that $\sum_{e'\in E_e^+} R_{ee'} = 1$ for all $e \in E$. Moreover, the natural topological constraints imply that $R_{ee'} = 0$ if $\tau_e \neq \sigma_{e'}$. To avoid trivial cases, we assume that for every node v, there exists at least one directed path from v to the node v_∞, so that every vehicle will eventually exit. The rate of vehicles passing link e is denoted by traffic flow f_e. According to [Coogan and Arcak, 2015], there exists a unique *equilibrium flow* $\mathbf{f} = \{f_e, e \in E\}$ such that:

$$f_e = d_e \qquad \forall e \in E_{entry} \tag{1}$$

$$f_e = \sum\nolimits_{e'\in E_e^-} f_{e'} R(e', e) \qquad \forall e \in E_{inter} \cup E_{exit} \tag{2}$$

For any given demand vector $\mathbf{d}$ and turning preference matrix R, the induced equilibrium flow $\mathbf{f}$ is used to calculate the minimal travel time t_e on link e. This solution can be easily extended to handle more complex traffic models [Aboudolas *et al.*, 2009; Daganzo, 1994; 1995].

Travel Time: t_e on a link e, strictly increasing with f_e, is defined by the commonly used Bureau of Public Roads (BPR) function [Manual, 1964; Skabardonis and Dowling, 1997]:

$$t_e = T_e(1 + 0.15 \times (f_e/C_e)^4) \tag{3}$$

3 The Escape Interdiction Game

The EIG is played between the escapee (attacker), and the police officers (defender). The attacker tries to escape to the external world v_∞ via any exit node $d \in D \subset \{v|\exists e = (v, v_\infty) \in E\}$ after conducting some criminal activity at node $v_0^a \in V$. There are m defender resources (police officers/teams), initially located at intersections $v_0^1, \dots, v_0^m \in V$. To only consider nontrivial problems, we assume that $v_0^a \neq v_0^r$, for all $r \in \mathcal{R} = \{1, \dots, m\}$. W.l.o.g., we assume that the event starts at time 0 and ends at time $t_{max} > 0$. The traffic flow on each edge e is likely to affect the corresponding travel time t_e of players, which is measured by Eq.(3). The traffic network, the initial positions of the bank and the police officers are common knowledge, but both players cannot see each other until the attacker gets caught.

Strategies: A pure attacker strategy is a sequence of states $A = \langle a_1 = (v_0^a, 0), \dots, a_j = (v_j, t_j^a), \dots, a_{k-1} = (d \in D, t_{k-1}^a), a_k = (v_\infty, t_k^a \leq t_{max})\rangle$, where each state $a_j = (v_j, t_j^a)$ represents the situation that the attacker arrives at node v_j at time t_j^a. For every two consecutive states $a_j = (v_j, t_j^a)$ and $a_{j+1} = (v_{j+1}, t_{j+1}^a)$ in an attacker strategy A, it is satisfied that $v_{j+1} \in N(v_j)$, where $N(v_j) = \{u \in V|(v_j, u) \in E\}$, and $t_{j+1}^a \geq t_j^a + t_{(v_j,v_{j+1})}$. This means the attacker can spend any time longer or equal to the minimal travel time $t_{(v_j,v_{j+1})}$ and makes the attacker's strategy space $\mathcal{A}$ continuous. A mixed attacker strategy is denoted by $\mathbf{y} = \langle y_A\rangle$, with y_A representing the probability that A is played.

A state of defender resource d_r is a tuple $s^r = (v^r, t^{r,in}, t^{r,out})$, representing the situation that d_r is protecting node v^r during $[t^{r,in}, t^{r,out}]$. We assume that d_r can only stay at a node for a multiple of a constant time interval δ (e.g., 10 seconds), i.e., $t^{r,out} - t^{r,in} = \kappa\delta$, where κ is a nonnegative integer. To simplify our analysis, we assume that d_r captures the attacker at a node. Thus, for every two consecutive states $s_i^r = (v_i^r, t_i^{r,in}, t_i^{r,out})$ and $s_{i+1}^r = (v_{i+1}^r, t_{i+1}^{r,in}, t_{i+1}^{r,out})$, $t_{i+1}^{r,in} - t_i^{r,out} = dist(v_i^r, v_{i+1}^r)$, which is the minimal travel time between v_i^r and v_{i+1}^r. This can easily be computed using Dijkstra's shortest path algorithm.

[1]This rate is possibly time-varying. For simplicity we consider it as constant during the relatively short time period of interest.

A pure strategy for the defender is a set of m schedules, i.e., $S = \{S^r : r \in \mathcal{R}\}$. The schedule for d_r is a sequence of states $S^r = \langle s_1^r, \ldots, s_i^r, \ldots, s_k^r \rangle$, where $s_1^r = (v_0^r, 0, t_1^{r,out})$ and $s_k^r = (v_k^r, t_k^{r,in}, t_{max})$. Moreover, d_r cannot drive to the external world v_∞ to capture the attacker, i.e., $v_i^r \in V \setminus \{v_\infty\}$, $\forall s_i^r \in S^r$. The defender's pure strategy space is denoted by $\mathcal{S}$. A defender's mixed strategy is $\mathbf{x} = \langle x_S \rangle$, with x_S representing the probability that S is played.

Utility: We are only concerned with whether the attacker can be captured or not, so we assume a zero-sum game. For $a_j = (v_j, t_j^a)$ and $s_i^r = (v_i^r, t_i^{r,in}, t_i^{r,out})$, we say the attacker is captured by d_r at node v_i^r if $v_i^r = v_j$ and $t_i^{r,in} \leq t_j^a \leq t_i^{r,out}$. In such a case, $z_{s_i^r, a_j} = 1$, otherwise $z_{s_i^r, a_j} = 0$. If an attacker successfully escapes to the external world, he gains a utility of 1 and the defender gets -1; otherwise both players get a utility of 0. Formally, the defender utility is given by:

$$U_d(S, A) = \begin{cases} 0 & \text{if } \exists z_{s_i^r, a_j} = 1, a_j \in A, s_i^r \in S^r, r \in \mathcal{R} \\ -1 & \text{otherwise} \end{cases} \quad (4)$$

Given $\mathbf{x}$ and A, the expected utility of the defender is $U_d(\mathbf{x}, A) = \sum_{S \in \mathcal{S}} U_d(S, A) x_S$. Accordingly, we can define $U_d(\mathbf{x}, \mathbf{y}) = \sum_{A \in \mathcal{A}} y_A U_d(\mathbf{x}, A)$. Note that $U_a = -U_d$.

Equilibrium: We use Nash equilibrium (NE) as the solution concept of this game as both players decide their strategies simultaneously. The NE is the same as the maxmin equilibrium given the zero-sum assumption. Thus, the optimal mixed strategy $\mathbf{x}$ of the defender can be computed by solving the following linear program (*LP*).

$$\max \quad U^* \quad (5)$$

$$\text{s.t.} \quad U^* \leq U_d(\mathbf{x}, A) \qquad \forall A \in \mathcal{A} \quad (6)$$

$$\sum_{S \in \mathcal{S}} x_S = 1, x_S \geq 0 \qquad \forall S \in \mathcal{S} \quad (7)$$

4 Solution Approach

Solving the *LP* in Eqs.(5)–(7) under the exponentially large $\mathcal{S}$ and the continuous $\mathcal{A}$ is challenging. To handle this challenge, we develop a novel double oracle algorithm *EIGS* (EIG Solver). Now, we first show that solving EIG is NP-hard.

Proposition 1. *Computing the NE of EIG is NP-hard.*

Proof. We reduce 3-SAT to the computation of NE of EIG. 3-SAT asks if there exists an assignment of values to a set of boolean variables that satisfies a given boolean formula in 3CNF (i.e., a conjunctive normal form where each clause is limited to at most three literals). Let the variables of the 3-SAT be $x_1, \ldots, x_n$, and the clauses be $C_1 \wedge C_2 \wedge \cdots \wedge C_k$. We construct an EIG on a road network shown in Figure 1, such that, as we will show later, the 3CNF is satisfiable if and only if (denoted as $\Leftrightarrow^1$) there exists a defender pure strategy intercepting *all* attacker pure strategies if and only if (denoted as $\Leftrightarrow^2$) the attacker gets caught with probability 1 in NE (under the mixed strategy setting). Particularly, we will focus the proof on "$\Leftrightarrow^1$" as "$\Leftrightarrow^2$" is trivial. If there exists such a defender pure strategy, then the defender can simply play a mixed strategy incorporating only this pure strategy regardless of which strategy the attacker plays. Otherwise, for any

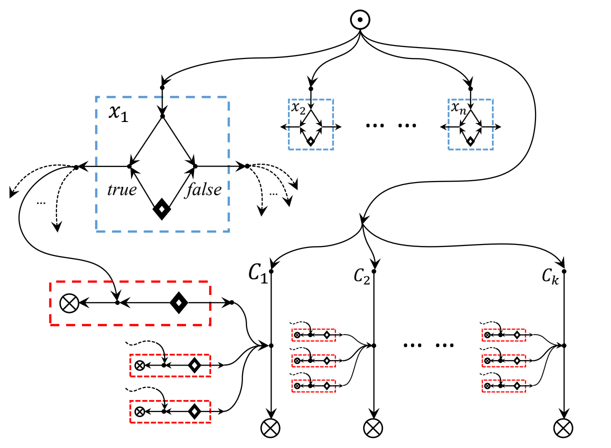


Figure 1: Reduce 3-SAT to EIG.

defender strategy, the attacker can always find a pure strategy not being intercepted by at least one pure strategy in the support of the defender strategy, which gives the attacker a non-zero probability of escaping successfully.

In this graph, the $\odot$ on the top represents the source node v_0^a where the attacker begins to escape; each of the $\otimes$'s on the bottom and in the red boxes represent an exit node through which the attacker can escape to the external world; each diamond represents a police station, where one police car is available.

For each variable x_i of the 3-SAT problem, we create a gadget shown in the blue box, in which there is a police station with two outbound roads. The police car from this station can choose either to move left or right, corresponding to $x_i = true$ or $false$, respectively. The choice of directions of all the police cars in these blue boxes corresponds to an assignment of values to $x_1, \ldots, x_n$ in the 3-SAT problem.

For each clause C_j, we create a gadget consisting of a road, call it the clause road, from v_0^a to an exit node and three components each corresponding to a literal of the clause. In each of the red boxes, there is a police station with two outbound roads. The one to the right links to the road of the clause, so that choosing this direction, the police car can intercept attacker pure strategies passing through the clause road. The one to the left leads to an exit node, and joints in the middle with another road coming from one of the blue boxes and corresponding to: $x_i = true$ if the literal is $\neg x_i$; and $x_i = false$ if the literal is x_i. Namely, for example, if in the blue box x_i is chosen to be $true$, the $false$ direction is left open for the attacker to pass through, so that to the police cars in the red boxes corresponding to $\neg x_i$ have to move left to intercept the attacker. In such a way, attacker pure strategies passing through a clause road can be intercepted (without missing any attack pure strategy exiting from the red boxes) only when one of its literals is made $true$ by choices of directions in the blue boxes.

We show that: the 3CNF is satisfiable $\Leftrightarrow$ there exists a defender pure strategy intercepting all attacker pure strategies.

1. The "$\Rightarrow$" direction. Suppose there exists a satisfying assignment for the 3-SAT problem. What the defender can do

Algorithm 1: *EIGS*

```
1  Initialize $\mathcal{S}'$, $\mathcal{A}'$.
2  repeat
3      $(\mathbf{x}, \mathbf{y}) \leftarrow CoreLP(\mathcal{S}', \mathcal{A}')$;
4      $\mathcal{S}^* \leftarrow \{betterDO(\mathbf{y})\}$;
5      if $\mathcal{S}^* = \emptyset$ then $\mathcal{S}^* \leftarrow \{bestDO(\mathbf{y})\}$;
6      $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathcal{S}^*$;
7      $\mathcal{A}^* \leftarrow \{betterAO(\mathbf{x})\}$;
8      if $\mathcal{A}^* = \emptyset \wedge \mathcal{S}^* = \emptyset$ then $\mathcal{A}^* \leftarrow \{bestAO(\mathbf{x})\}$;
9      $\mathcal{A}' \leftarrow \mathcal{A}' \cup \mathcal{A}^*$;
10 until convergence;
11 return $(\mathbf{x}, \mathbf{y})$.
```

is let the police car in each blue box move to the direction that is the same as the value of x_i in the satisfying assignment and stay at the first intersection. The other direction in each blue box is left open, but the roads passing through it eventually lead to the red boxes and can be intercepted by moving police cars in the red boxes to the left. Finally, since each clause contains at least one literal which is made true, the police cars in the corresponding red box do not need to move left to intercept the attacker as the attacker is already intercepted in the blue boxes; therefore, letting these police cars move to the clause road, all attacker pure strategies passing through the clause roads get intercepted as well.

2. The "$\Leftarrow$" direction. If there is not a satisfying assignment, then no matter how the defender moves the resources, there exists at least one clause such that the police cars in the red boxes of its literals all have to move left to intercept the incoming attacker. The clause road is left open. Then the attacker strategy passing through the clause road reaches the exit node successfully. $\square$

4.1 A Novel Double Oracle Framework

Now we develop *EIGS*, which first computes the equilibrium strategy for a significantly smaller restricted game, then iteratively computes improving strategies for both players, and finally converges to a global equilibrium.

EIGS is sketched in Algorithm 1. Line 1 initializes $\mathcal{S}'$ and $\mathcal{A}'$, which are small sets of pure strategies for both players. Then, *CoreLP* computes the maximin strategies $\mathbf{x}$ and $\mathbf{y}$ for the restricted game $\langle \mathcal{S}', \mathcal{A}' \rangle$ (Line 3) by solving the *LP* in Eqs.(5)–(7). To improve both players' utilities, the defender oracle (DO) (Lines 4–6) and the attacker oracle (AO) (Lines 7–9) are repeatedly used to find other strategies out of $\langle \mathcal{S}', \mathcal{A}' \rangle$ until no improving strategy can be found (Line 10). DO first calls efficient *betterDO* (better response DO) trying to find an improving strategy that may be suboptimal. If *betterDO* fails, it proceeds to *bestDO* (best response DO) that is optimal. Similarly, AO first calls efficient *betterAO* (better response AO), and if *betterAO* and DO both fail, it proceeds to *bestAO* (best response AO). This structure at Line 8 is different from the classic double oracle employed in security games [Jain *et al.*, 2013; Wang *et al.*, 2016], where the best oracle will be called if the corresponding better oracle fails. However, it is not efficient if one runs much slower than the other between best oracles.

Algorithm 2: *betterDO* $(\mathbf{y})$

```
1  $\overline{\mathcal{A}} \leftarrow \mathcal{A}'$; $\forall r \in \mathcal{R}:\ S^r \leftarrow \langle (v_0^r, 0, 0) \rangle$, $L^r \leftarrow 1$, $t_1^{r,out} \leftarrow 0$;
2  while $\overline{\mathcal{A}} \neq \emptyset$ do
3      for $r \in \mathcal{R}$ do
4          for $v \in V \setminus \{v_\infty\}$ do
5              $t[r][v] \leftarrow t_{L^r}^{r,out} + dist(v_{L^r}, v)$, $\Delta y[r][v] \leftarrow 0$;
6              for $A \in \overline{\mathcal{A}}$ do
7                  if $\exists a_j = (v, t_j^a) \in A, t[r][v] \le t_j^a$ then
8                      $\Delta y[r][v] \leftarrow \Delta y[r][v] + y_A$;
9          $v^{r,*} \leftarrow \arg\max_v \Delta y[r][v]$;
10     $r^* \leftarrow \arg\max_r \Delta y[r][v^{r,*}]$;
11     if $\Delta y[r^*][v^{r,*}] > 0$ then
12         $L^{r^*} \leftarrow L^{r^*} + 1$, $v_{L^{r^*}} = v^{r,*}$;
13         for $A \in \overline{\mathcal{A}}$ do
14             if $\exists a_j = (v_{L^{r^*}}, t_j^a) \in A$ then
15                 $t_{L^{r^*}}^{r^*,out} \leftarrow \max\{t[r^*][v_{L^{r^*}}], t_j^a\}$, $\overline{\mathcal{A}} \leftarrow \overline{\mathcal{A}} \setminus \{A\}$;
16         $\kappa \leftarrow \lceil t_{L^{r^*}}^{r^*,out} - t[r^*][v_{L^{r^*}}]/\delta \rceil$, $t_{L^{r^*}}^{r^*,out} \leftarrow t[r^*][v_{L^{r^*}}] + \delta \cdot \kappa$;
17         $S^{r^*} \leftarrow S^{r^*} \cup \langle (v_{L^{r^*}}, t[r^*][v_{L^{r^*}}], t_{L^{r^*}}^{r^*,out}) \rangle$;
18     else break;
19 return $\{S^r, r \in \mathcal{R}\}$.
```

4.2 Defender Oracle

The defender needs to find an improving path with a stop time such that it can intercept as many attacker paths as possible, i.e., increase the probability of catching the attacker. Formally, given $(\mathbf{x}, \mathbf{y})$ provided by *CoreLP*, S is added to $\mathcal{S}'$ if $U_d(S, \mathbf{y}) > U_d(\mathbf{x}, \mathbf{y})$. This can be formulated as an MILP, i.e., ***bestDO***. It first constructs a path attaching travel time and stop time, and then checks if it can intercept the given attacker path, i.e., both players meet at the same time and the same node. If there are more attacker paths being intercepted by this new defender path, the defender will get a higher utility. This procedure is represented by the following MILP:

$$\max -\sum\nolimits_{A \in \mathcal{A}'} (1 - z_A) y_A \tag{8}$$

$$\text{s.t. } s_{1,v_0^r}^r = 1, \sum\nolimits_{v \in V \setminus \{v_\infty\}} s_{i,v}^r = 1 \quad \forall r, i \tag{9}$$

$$\omega_{r,i,(v,u)} \le \min(s_{i,v}^r, s_{i+1,u}^r) \quad \forall r, i, v, u \tag{10}$$

$$\omega_{r,i,(v,u)} \ge s_{i,v}^r + s_{i+1,u}^r - 1 \quad \forall r, i, v, u \tag{11}$$

$$t_1^{r,in} = 0, t_{L_{max}^d}^{r,out} = t_{max}, t_i^{r,out} = t_i^{r,in} + \kappa_{r,i}\delta \ \ \forall r, i \tag{12}$$

$$t_{i+1}^{r,in} = t_i^{r,out} + \sum\nolimits_{v,u \in V \setminus \{v_\infty\}} dist(v,u)\omega_{r,i,(v,u)} \ \forall r, i \tag{13}$$

$$-M\alpha_{r,i}^{A,j} \le t_i^{r,in} - t_j^A \le M(1 - \alpha_{r,i}^{A,j}) \ \ \forall r, i, A, j \tag{14}$$

$$-M\beta_{r,i}^{A,j} \le t_j^A - t_i^{r,out} \le M(1 - \beta_{r,i}^{A,j}) \ \ \forall r, i, A, j \tag{15}$$

$$\gamma_{r,i}^{A,j} \le (\alpha_{r,i}^{A,j} + \beta_{r,i}^{A,j} + s_{i,v_j^A}^r)/3 \quad \forall r, i, A, j \tag{16}$$

$$\gamma_{r,i}^{A,j} \ge \alpha_{r,i}^{A,j} + \beta_{r,i}^{A,j} + s_{i,v_j^A}^r - 2 \quad \forall r, i, A, j \tag{17}$$

$$z_A \le \sum\nolimits_{j,r,i} \gamma_{r,i}^{A,j} \quad \forall A \tag{18}$$

$$s_{i,v}^r, \omega_{r,i,(v,u)}, \alpha_{r,i}^{A,j}, \beta_{r,i}^{A,j}, \gamma_{r,i}^{A,j}, z_A \in \{0, 1\} \tag{19}$$

Algorithm 3: *betterAO* ($\mathbf{x}$)

```
1  $\forall v \in V \setminus \{v_0^a\}$, $\Delta x[v] \leftarrow \infty$, $t^a[v] \leftarrow \infty$;
2  $\Delta x[v_0^a] \leftarrow 0$, $t^a[v_0^a] \leftarrow 0$, $\overline{\mathcal{S}}_{v_0^a} \leftarrow \mathcal{S}'$, $Q \leftarrow V$;
3  while $Q \neq \emptyset$ do
4      $v \leftarrow \arg\min_{u \in Q} \Delta x[u]$;
5      if $v = v_\infty$ then break;
6      $Q \leftarrow Q \setminus \{v\}$;
7      for each neighbor $u$ of $v$ do
8          $t \leftarrow \arg\min_{t' \geq t^a[v]+t_{(v,u)}} \Phi(t'):=\sum_{S \in \overline{\mathcal{S}}_v} x_S \cdot \phi_u(S,t')$;
9          if $\Delta x[v] + \Phi(t) < \Delta x[u]$ then
10             $\Delta x[u] \leftarrow \Delta x[v] + \Phi(t)$;
11             $t^a[u] \leftarrow t^a[v] + t$;
12             $prev[u] \leftarrow v$;
13             $\overline{\mathcal{S}}_u \leftarrow \overline{\mathcal{S}}_v \setminus \{S | u \in S, t^a[u] \in [t_{S,u}^{in}, t_{S,u}^{out}]\}$;
14 $A \leftarrow \langle (v_0, 0), (v_1, t^a[v_1]), \ldots, (v_\eta, t^a[v_\eta]) \rangle$ such that
   $v_0 = v_0^a$, $v_\eta = v_\infty$, and $v_i = prev[v_{i+1}]\ \forall i = 0, \ldots, \eta - 1$;
15 return $A$.
```

$$\kappa_{r,i} \in \mathbb{Z}_{\geq 0}, t_i^{r,in}, t_i^{r,out} \in [0, t_{max}] \tag{20}$$

Here, $s_{i,v}^r = 1$ indicates that d_r arrives at v by the i-th state of S^r. Eq.(9) indicates that d_r starts at v_0^r and stays on only one node in any state of S^r. In Eqs.(10) and (11), the $0/1$ variable $\omega_{r,i,(v,u)}$ encodes whether there is a path v-u between the i-th state and $(i+1)$-th state of S^r. Eq.(12) forces the strategy to start at time 0 and end at t_{max}, where L_{max} is a sufficiently large number to estimate the maximum length allowed of the defender strategy sequence, and the time that d_r stays on the i-th state is $\kappa_{r,i}\delta$. Eq.(13) ensures that d_r travels one stop to another through the shortest path. With Eqs.(14)–(18), z_A determines whether the attacker using A is caught. Specifically, (v_j^A, t_j^A) represents the attacker's j-th state in A; $\gamma_{r,i}^{A,j}$ indicates whether the attacker using A, meets d_r at his j-th state's position while d_r is at her i-th stop; $\alpha_{r,i}^{A,j}$ (respectively, $\beta_{r,i}^{A,j}$) indicates whether the attacker arrives at his j-th state's position after d_r arrives (respectively, before d_r leaves); and M is a very large number.

To speed up the above MILP, we propose a faster oracle ***betterDO*** as shown in Algorithm 2. It is a greedy algorithm, i.e., at each step, the defender goes to the nodes with the highest marginal value. This marginal value on each node is the probability of intercepting the remaining attacker paths if the defender moves to that node. As shown in Algorithm 2, $\overline{\mathcal{A}}$ stores the attacker's paths that are not caught by any S^r. Initially, d_r is at her initial position v_0^r (Line 1). Lines 2–18 are the main loop, where we repeatedly allocate police officers to interdict the attacker's paths with the highest marginal value. Specifically, Lines 3–9 enumerate all the combinations $\langle r, v \rangle \in \mathcal{R} \times V$, and check how much value d_r can obtain if she moves to v and stays there as long as possible (therefore, in Lines 6–8, all attacker paths in which the attacker arrives at v later than $t[r][v]$ are intercepted, and the corresponding values are added to $\Delta y[r][u]$). d_{r^*} who can obtain the highest value among defender resources is chosen (Line 10). The remaining part is to update the finalized movement (Lines 13–15), where the attacker paths caught by the movement are removed from $\overline{\mathcal{A}}$, and $t_{L^{r^*}}^{r^*,out}$ is updated to the time that this movement can intercept as many attacker paths as possible. d_{r^*} stays at a node for $\kappa\delta$ (Line 16) and then updates her new state to the strategy sequence (Line 17).

4.3 Attacker Oracle

The attacker needs to find an improving path with a travel time such that it can minimize the probability of being caught. Formally, given $(\mathbf{x}, \mathbf{y})$ provided by *CoreLP*, A is added to $\mathcal{A}'$ if $U_a(\mathbf{x}, A) > U_a(\mathbf{x}, \mathbf{y})$. This can be formulated as an MILP, i.e., ***bestAO***. It first constructs a path attaching travel time, and then checks if it is intercepted by the given defender path, i.e., both players meet at the same time and the same node. If there are more defender paths intercepting this new attacker path, the attacker will get a lower utility. This procedure is represented by the following MILP:

$$\max \sum\nolimits_{S \in \mathcal{S}'} (1 - z_S) x_S \tag{21}$$

$$\text{s.t. } A_{1,v_0^a} = 1, A_{L_{max}^a, v_\infty} = 1, \sum\nolimits_{v \in V} A_{j,v} = 1 \quad \forall j \tag{22}$$

$$A_{j+1,v_\infty} \geq A_{j,v_\infty} \quad \forall j \tag{23}$$

$$\sum\nolimits_{u \in N(v)} A_{j+1,u} \geq A_{j,v} \ \forall v \in V, j \tag{24}$$

$$\omega_{j,(v,u)} \leq \min(A_{j,v}, A_{j+1,u}) \quad \forall (v,u) \in E, j \tag{25}$$

$$\omega_{j,(v,u)} \geq A_{j,v} + A_{j+1,u} - 1 \quad \forall (v,u) \in E, j \tag{26}$$

$$t_1^a = 0 \tag{27}$$

$$t_{j+1}^a \geq t_j^a + \sum\nolimits_{(v,u) \in E} t_{(v,u)} \omega_{j,(v,u)} \quad \forall j \tag{28}$$

$$-M\alpha_{S,r,i}^j \leq t_i^{r,in} - t_j^a \leq M(1 - \alpha_{S,r,i}^j) \ \forall S, r, i, j \tag{29}$$

$$-M\beta_{S,r,i}^j \leq t_j^a - t_i^{r,out} \leq M(1 - \beta_{S,r,i}^j) \ \forall S, r, i, j \tag{30}$$

$$z_S \geq \alpha_{S,r,i}^j + \beta_{S,r,i}^j + A_{j,v_i^{Sr}} - 2 \quad \forall S, r, i, j \tag{31}$$

$$A_{j,v}, \alpha_{S,r,i}^j, \beta_{S,r,i}^j, \omega_{j,(v,u)}, z_S \in \{0,1\}, t_j^a \in [0, t_{max}] \tag{32}$$

Here $A_{j,v}$ indicates whether the attacker arrives at v in the j-th state. Eq.(22) ensures that the attacker starts at v_0^a, terminates at v_∞, and only arrives at one node in each state. Eqs.(23)–(24) fix the attacker at the sink node v_∞ in the following states after he reaches v_∞ and ensure that the attacker's strategy is a path; namely, the attacker can visit some node u in state $j+1$ only if it is a neighbour of the node v in state j recorded by $\omega_{j,(v,u)}$ in Eqs.(25)–(26). Eqs.(27)–(28) initialize the time and update it at the following states. Finally, with Eqs.(29)–(31), z_S captures whether the attacker is caught by S.

Now we propose ***betterAO*** to efficiently achieve a better response. This algorithm is similar to Dijkstra's algorithm (see Algorithm 3). Let's say the *length* of an attacker path is the probability of being caught by the given $\mathbf{x}$. Variable $\Delta x[v]$ maintains, for all $v \in V$, the shortest so-far-known acyclic path from v_0^a to v. In the main loop, the node with minimum Δx is extracted from Q (Line 4). Then in Lines 7–13, Δx of each neighbor of v is updated with $\Delta x[v]$, where in Line 8 a best time point t is calculated such that reaching u at t minimizes the probability of attacker being caught ($\phi_u(S, t) = 1$ if in S at least one defender resource stays at u at time t, and 0 otherwise). $prev[u]$ maintains the previous node of u in the so-far-known shortest path, with which an attacker path is constructed in the end (Line 14).

5 Improving the Scalability

EIGS has improved its scalability by deploying efficient better oracles. However, it it cannot efficiently solve the best oracle of large scale games due to the large strategy spaces. In this section, we further improve scalability without significantly sacrificing the defender's utility much. Our approach is motivated by the intuition that the attacker wants to escape as soon as possible and, correspondingly, each police officer goes to one certain node with maximum speed and then waits for the attacker.

5.1 A Reduced Game of EIG

$\mathcal{A}$ is reduced to $\underline{\mathcal{A}}$, which satisfies the condition that for every two consecutive states $a_j = (v_j, t^a_j)$ and $a_{j+1} = (v_{j+1}, t^a_{j+1})$ of each attacker strategy $A \in \underline{\mathcal{A}}$, $t^a_{j+1} = t^a_j + t_{(v_j,v_{j+1})}$. That is, the attacker travels with the maximum speed. $\mathcal{S}$ is reduced to $\underline{\mathcal{S}}$, which satisfies the condition that for each defender strategy $S \in \underline{\mathcal{S}}$, the schedule S^r has the following form: $S^r = \langle(v^r_0, 0, 0), (v^r_1, dist(v^r_0, v^r_1), t_{max})\rangle$, where $v^r_1 \in V \setminus \{v_\infty\}$. That is, the defender goes to some node with maximum speed and waits there. Obviously, $\underline{\mathcal{A}} \subseteq \mathcal{A}$ and $\underline{\mathcal{S}} \subseteq \mathcal{S}$. Now we have the following property about the relation of $\underline{\mathcal{A}}$ and $\underline{\mathcal{S}}$.

Proposition 2. *Given a defender strategy* $\mathbf{x}$ *with its support set* $\mathcal{S}_\mathbf{x} \subseteq \underline{\mathcal{S}}$, *there is an attacker optimal strategy* $A^* \in \underline{\mathcal{A}}$.

Proof. Suppose that there is not an optimal strategy $A^* \in \underline{\mathcal{A}}$. This means that if A is an optimal strategy, then $A \in \mathcal{A} \setminus \underline{\mathcal{A}}$ and $U_a(\mathbf{x}, A^*) < U_a(\mathbf{x}, A)$.

In any strategy $S \in \underline{\mathcal{S}}$, the schedule S^r has the following form: $S^r = \langle(v^r_0, 0, 0), (v^r, t^{r,in}, t_{max})\rangle$, where $0 \leq t^{r,in} = dist(v^r_0, v^r_1) \leq t_{max}$. Suppose the optimal strategy $A = \langle a_1 = (v^a_0, t^a_0 = 0), \ldots, a_j = (v_j, t^a_j), \ldots, a_{k-1} = (v_{k-1}, t^a_{k-1}), a_k = (v_\infty, t^a_k)\rangle$. Keeping the nodes' sequence order of A , we can obtain $\underline{\mathcal{A}}$'s strategy $A^* = \langle a^*_1 = (v^a_0, t^{a^*}_0 = 0), \ldots, a^*_j = (v_j, t^{a^*}_j), \ldots, a^*_{k-1} = (v_{k-1}, t^{a^*}_{k-1}), a^*_k = (v_\infty, t^{a^*}_k)\rangle$, where $t^{a*}_j = t_{(v_{j-1},v_j)} + t^{a*}_{j-1} (j > 0)$. Then $t^{a^*}_j \leq t^a_j$ $(\forall a^*_j \in A^*, a_j \in A)$ as $t^a_j \geq t_{(v_{j-1},v_j)} + t^a_{j-1} (j > 0)$.

Moreover, we define $z^*_{S^r,a^*_j} \in \{0,1\}$ and $z_{S^r,a_j} \in \{0,1\}$ $(a^*_j \in A^*, a_j \in A)$ as: $z^*_{S^r,a^*_j} = 1$ if $v^r = v_j, t^{r,in} \leq t^{a*}_j$, otherwise $z^*_{S^r,a^*_j} = 0$; and $z_{S^r,a_j} = 1$ if $v^r = v_j, t^{r,in} \leq t^a_j$, otherwise $z_{S^r,a_j} = 0$. Then, $z^*_{S^r,a^*_j} \leq z_{S^r,a_j}$ as $t^{a^*}_j \leq t^a_j$ $(\forall a^*_j \in A^*, a_j \in A)$.

Furthermore, we define $z^*_S \in \{0,1\}$ and $z_S \in \{0,1\}$ $(S \in \mathcal{S}_\mathbf{x})$ as: $z^*_S = 1$ if $\exists z^*_{S^r,a^*_j} = 1 (r \in \mathcal{R}, a^*_j \in A^*)$, otherwise $z^*_S = 0$; and $z_S = 1$ if $\exists z_{S^r,a_j} = 1 (r \in \mathcal{R}, a_j \in A)$, otherwise $z_S = 0$. Then, $z^*_S \leq z_S$ as $z^*_{S^r,a^*_j} \leq z_{S^r,a_j}$ $(\forall a^*_j \in A^*, a_j \in A)$.

So the attacker's utility $U_a(\mathbf{x}, A^*) = \sum_{S\in\mathcal{S}_\mathbf{x}}(1-z^*_S)x_S \geq \sum_{S\in\mathcal{S}_\mathbf{x}}(1 - z_S)x_S = U_a(\mathbf{x}, A)$, which causes a contradiction. □

Algorithm 4: *advancedAO*($\mathbf{x}$)

```
1  $V_\mathbf{x} \leftarrow \{v^r_i \mid x_S > 0, (v^r_i, t^{r,in}_i, t^{r,out}_i) \in S^r \in S\}$;
2  $V_f \leftarrow$ DFS$(G, V_\mathbf{x}, v^a_0), V_{\mathbf{x},f} \leftarrow V_\mathbf{x} \cap V_f$;
3  if $v_\infty \in V_f$ then $\mathcal{A}^* \leftarrow \{P(dist_{(V_f,V_{\mathbf{x},f})}(v^a_0, v_\infty))\}$;
4  else
5      $V_b \leftarrow$ DFS$((V, \{(u,v) \mid (v,u) \in E\}), V_\mathbf{x}, v_\infty)$;
6      $V_{\mathbf{x},b} \leftarrow V_\mathbf{x} \cap V_b$;
7      $E \leftarrow E \cup \{(v^a_0, v) \mid v \in V_{\mathbf{x},f}\} \cup \{(v, v_\infty) \mid v \in V_{\mathbf{x},b}\}$;
8      $t_{v^a_0,v} \leftarrow dist_{(V_f,V_{\mathbf{x},f})}(v^a_0, v), \forall v \in V_{\mathbf{x},f}$;
9      $t_{v,v_\infty} \leftarrow dist_{(V_b,V_{\mathbf{x},b})}(v, v_\infty), \forall v \in V_{\mathbf{x},b}$;
10     $V_c \leftarrow (V \setminus (V_f \cup V_b)) \cup (V_{\mathbf{x},f} \cup V_{\mathbf{x},b} \cup \{v^a_0, v_\infty\})$;
11     $E_c \leftarrow \{(u,v) \mid u,v \in V_c, (u,v) \in E\}$;
12     $\mathcal{A} \leftarrow \{RedbestAO(\mathbf{x})\}$;
13     if $\mathcal{A} \neq \emptyset$ then $\mathcal{A}^* \leftarrow \{P(dist_{(V_f,V_{\mathbf{x},f})}(v^a_0, v_1)) \cup$
       $(v^a_2, t^a_2), \cdots, (v^a_{l-2}, t^a_{l-2}) \cup P(dist_{(V_b,V_{\mathbf{x},b})}(v^a_{l-1}, v_\infty)) \mid$
       $(v^a_0, t^a_0), \cdots, (v_\infty, t^a_l) \in \mathcal{A}\}$;
14 return $\mathcal{A}^*$.
```

Therefore, the utility obtained from the optimal defender strategy of the reduced game $(\underline{\mathcal{S}}, \underline{\mathcal{A}})$ is the same as the one in the semi-reduced game $(\underline{\mathcal{S}}, \mathcal{A})$ by Proposition 2. This means that *bestAO* cannot find an improving strategy for the attacker if the defender strategy is an equilibrium of $(\underline{\mathcal{S}}, \underline{\mathcal{A}})$.

5.2 An Approximate Algorithm of EIG

Based on the reduced game $(\underline{\mathcal{S}}, \underline{\mathcal{A}})$, we can redesign the oracles in Algorithm 1. Here the new best DO, *RedbestDO*, has the same objective function as *bestDO* and it has the following constraints:

$$\textstyle\sum_{v\in V\setminus\{v_\infty\}} s^r_v = 1 \quad \forall r \tag{33}$$

$$\textstyle\sum_{r\in\mathcal{R}} s^r_v \leq 1 \quad \forall v \in V \setminus \{v_\infty\} \tag{34}$$

$$z_A \leq \textstyle\sum_{j,r,dist(v^r_0,v^A_j)\leq t^A_j} s^r_{v^A_j} \quad \forall A \tag{35}$$

$$s^r_v, z_A \in \{0,1\} \tag{36}$$

$s^r_v = 1$ indicates that resource r is deployed to node v. Eq.(33) ensures that each resource will be assigned to one node, which is protected by at most one resource (Eq.(34)). z_A indicates whether the attacker's path is intercepted by the defender. $z_A = 1$ only when there is one node on the path where there is one resource reaching there before the attacker (reflected by $dist(v^r_0, v^A_j) \leq t^A_j$) enforced by Eq.(35).

The new better DO, *RedbetterDO*, is similar to *betterDO*, except that *RedbetterDO* only deploys each resource to one node. The new best AO, *RedbestAO*, is obtained from *bestAO* by changing the method for updating time in Eq.(28) to $t^a_{j+1} = t^a_j + \sum_{(v,u)\in E} t_{(v,u)}\omega_{j,(v,u)}$ to reflect the fact that the attacker will travel with maximum speed. Similarly, the new better AO, *RedbetterAO*, is obtained from *betterAO* by modifying Line 8 in Algorithm 3 to: if $\exists S^r s.t. z_{s^r_2,(u,t^a[v]+t_{v,u})} = 1, \forall S \in \overline{\mathcal{S}}_v$, then $\Phi(t) \leftarrow \Phi(t) + x_S$.

In the above new oracles, *RedbestAO* still has to add $|V|$ variables (i.e., $A_{j,v}$) for each state in A, which will dramatically slow it down. To speed it up, we further reduce the attacker strategy space by graph contraction.

***advancedAO*:** The advanced AO is based on the following intuition: for the nodes where no defender will appear, the at-

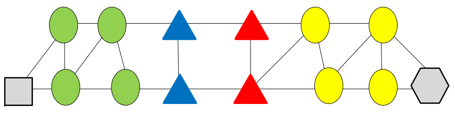


Figure 2: An scenario: the defender will only randomly appear at the triangle nodes to interdict the attacker but will not appear at other nodes; now the attacker needs to find an optimal strategy considering travel time from his current position (the square node on the left) to the external world (the hexagon node on the right).

tacker can travel freely through them without being caught at any time; but, for the node where the defender will appear, he has to find an optimal strategy, e.g., the optimal time, to travel through these nodes to avoid being caught. For example, as shown in Figure 2, to reach the external world, the attacker needs to travel through the circle and triangle nodes. The attacker knows that he will not be caught at the circle nodes but will probably be caught at the triangle nodes. So he can freely take any strategy (path) to travel through the circle nodes but wants to take an optimal strategy (path) to travel through the triangle nodes. To reflect these facts in a simplified network, the circle nodes and their adjacent edges can be deleted. Then four new edges from the square node to each blue triangle node and from each red triangle node to the hexagon node will be added with the travel time of the corresponding special shortest path[2] in the original network.

This contraction procedure is shown in Algorithm 4. $V_{\mathbf{x}}$ stores nodes in the support set of $\mathbf{x}$ (Line 1). V_f, a set of nodes that can be reached from v_0^a without traversing the nodes in $V_{\mathbf{x}}$, is given by DFS$(G, V_{\mathbf{x}}, v_0^a)$ (Line 2), where DFS$(G, V_{\mathbf{x}}, v_0^a)$ is obtained from the classic Depth First Search algorithm by ending each path at the nodes in $V_{\mathbf{x}}$. That is, a node v is pushed into the stack only if v is unvisited and $v \notin V_{\mathbf{x}}$. If $v_\infty \in V_f$, then $P(dist_{(V_f,V_{\mathbf{x},f})}(v_0^a, v_\infty))$ is an optimal strategy that will not be intercepted by the defender (Line 3). Here $P(dist_{(V_f,V_{\mathbf{x},f})}(v_0^a, v_\infty))$ is the shortest path from v_0^a to v_∞ on the network $(V_f, \{(u,v) \mid (u,v) \in E, u \in V_f \setminus V_{\mathbf{x},f}, v \in V_f\})$ with corresponding travel time $dist_{(V_f,V_{\mathbf{x},f})}(v_0^a, v_\infty)$. Otherwise, the network G is contracted (Lines 5–11) and *RedbestAO*$(\mathbf{x})$ is called for the contracted network (Line 12). V_b stores nodes reached from v_∞ in the backward network of G (Line 5). After that, new edges are added (Line 7) with corresponding travel time (Lines 8 and 9), and then, the contracted network (V_c, E_c) is generated in Lines 10 and 11. Finally, Line 13 maps the generated strategy (Line 12) to the original network.

Now we can obtain ***RedAD***, an efficient algorithm based on the reduced game of EIG, from Algorithm 1 by replacing *bestDO*, *betterDO*, *bestAO*, and *betterAO* with *RedbestDO*, *RedbetterDO*, *advancedAO*[3], and *RedbetterAO*, respectively.

[2]They are not necessary the shortest paths in the network because they do not contain the triangle nodes, except for the start or end node.

[3]*advancedAO* can be applied to the full game $(\mathcal{S}, \mathcal{A})$ by replacing *RedbestAO* with *bestAO* in Line 12 of Algorithm 4. Actually, we also propose a two-stage approach for the full game $(\mathcal{S}, \mathcal{A})$ which uses *RedAD* to initialize $(\mathcal{S}', \mathcal{A}')$ in Line 1 of Algorithm 1 to obtain algorithm *EIGS-TS*, and then obtain a heuristic algorithm *EIGS-TS-H* where *bestAO* is replaced by *advancedAO*. However, they cannot improve much in solution quality and scalability.

6 Experimental Evaluation

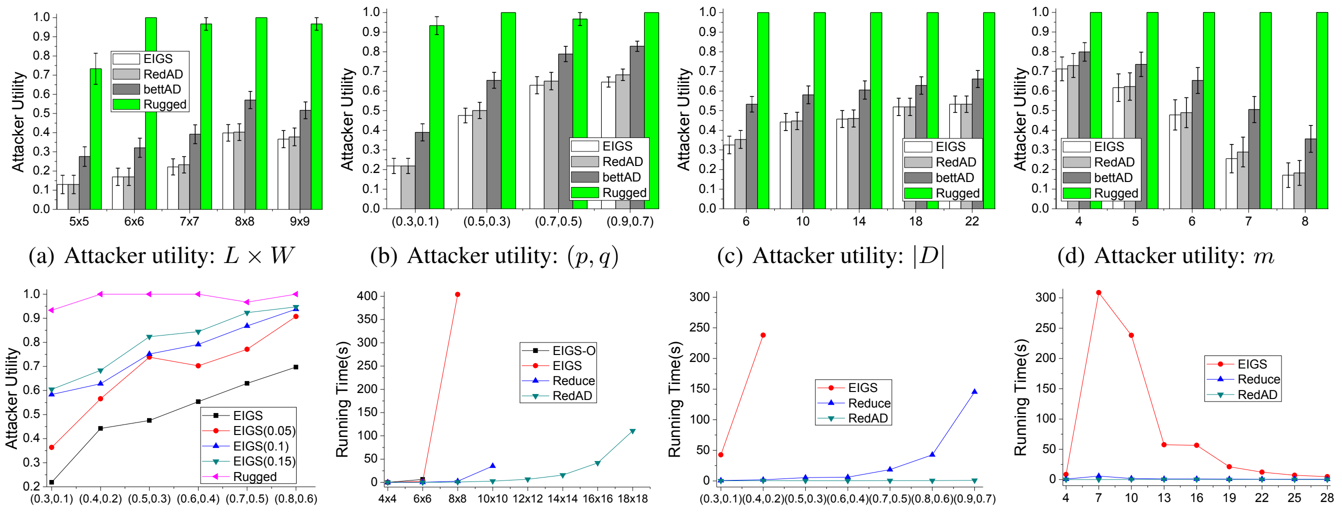
We evaluate our model and algorithms with numerical experiments. All LPs and MILPs are solved with CPLEX (version 12.6). All computations are performed on a machine with a 3.20GHz quad core CPU and 16.00GB memory. All points in the figures are averaged over 30 samples. We use the Grid model with Random Edges (GRE) to generate urban road network topologies [Peng *et al.*, 2014]. GRE is a planar connected graph made of a $L \times W$ square grid of nodes, where horizontal/vertical edges between neighbors are controlled by probability p, and diagonal ones by q. The values of p spread in $[0.3, 0.9]$ and q in $[0.1, 0.7]$ to best match realistic road networks. We add a node v_∞ that is connected to all the nodes at the border of the area to generate the entry and exit links.

In the traffic model, $C_e = 6$, and T_e is randomly chosen in [1,10]. The traffic demand of each entry link is randomly chosen in [0,6], and the turning preference matrix R is uniformly generated. In EIG, v_0^a is fixed to the center node of the area, and all the m defender resources are initially uniformly distributed on the network. The exit nodes D are randomly chosen from the nodes at the border of the area. The defender travels faster than the attacker because the police can circumvent traffic constraints unless otherwise specified. Central parameter values are chosen as $L \times W = 8 \times 8$ nodes, $(p, q) = (0.4, 0.2)$, $|D| = 10$ exit nodes and $m = 4$ resources unless otherwise specified.

6.1 Solution Quality

We compare the solution quality of our approaches with two baselines. The first baseline is *Rugged* [Jain *et al.*, 2011], where the defender resources are deployed to intercept the attacker's path without considering the travel time on paths. *Rugged* is a representative solution in the urban network security domain for the following reasons: 1) it is an extension of the min-cut method that the defender resources are uniformly distributed on the $v_0^a - v_\infty$ min-cut [Washburn and Wood, 1995] without the travel time constraint; 2) it has the same solution quality as its variation of *Snares* [Jain *et al.*, 2013] because *Snares* deploys *Rugged*'s best response oracles, i.e., *Snares* is just an improvement of *Rugged* in scalability; 3) its performance in solution quality is better than *MiCANS* [Iwashita *et al.*, 2016], which is an approximate algorithm for urban network security based on multiple cuts. The second baseline is the approximate algorithm *bettAD* where *EIGS* does not call the best response oracles. Comparing *bettAD* with *RedAD* in solution quality will further show the importance of developing *RedAD*. The solution quality of strategy $\mathbf{x}$ in *Rugged*, *RedAD*, and *bettAD* is assessed in terms of $U_a(\mathbf{x}, A)$, where A is obtained by using *bestAO*. A lower attacker utility indicates a higher defender utility given the zero-sum assumption.

Figures 3(a)–3(d) compare the solution quality obtained from our approaches with those obtained from baselines varying: the number of intersections, denoted by $L \times W$, with



(a) Attacker utility: $L \times W$ (b) Attacker utility: (p, q) (c) Attacker utility: $|D|$ (d) Attacker utility: m

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

Figure 3: Solution quality: (a)–(d); Robustness: (e); Scalability: (f)–(h).

$|D| = L$ in Figure 3(a); edge probabilities in Figure 3(b); the number of exit nodes in Figure 3(c); and the number of resources in Figure 3(d). Results show that *Rugged* performs the worst in all cases because it is designed without considering players' travel time on roads. Meanwhile, *EIGS* performs the best. *bettAD* performs significantly worse than *EIGS*. However, *RedAD* achieves good solution quality, which has similar performance to *EIGS* and significantly outperforms *bettAD* in most cases.

In terms of robustness, the defender may face execution uncertainty caused by the unpredictable delays associated with congestion, traffic signals, etc. To analyze the performance of *EIGS* in the presence of uncertainty, we add noise θ to the travel time. So the actual travel time $\hat{t_e} = t_e \times (1 + \theta)$. Figure 3(e) shows the attacker utility of *EIGS* under different degrees of uncertainty. Here we can see that while the performance of *EIGS* decreases as θ is increased, it still outperforms *Rugged*.

6.2 Scalability

We compare the scalability of *EIGS* with a baseline, *EIGS-O*, which runs *EIGS* under the double oracle framework adopted by [Jain *et al.*, 2013; Wang *et al.*, 2016]. This shows the effectiveness of our new double oracle framework. In addition, we evaluate the efficiency of *RedAD* with another baseline, *Reduce*, where *RedAD* calls *RedbestAO* instead of *advancedAO*.[4]

Figures 3(f)–3(h) present the runtime in seconds varying different variables similar to the setting of solution quality. From Figure 3(f), we can see that *EIGS-O* does not scale up to the network with 8×8 intersections, so we do not show the result of *EIGS-O* in Figures 3(g) and 3(h). Results show that: 1) Our new double oracle framework outperforms the old one by comparing *EIGS* with *EIGS-O*; 2) *RedAD* outperforms *EIGS* and *Reduce* significantly; 3) The running time generally increases with the size of the network and the edge probabilities, but it reflects the easy-hard-easy pattern in security games [Jain *et al.*, 2012]; 4) *RedAD* can scale up to realistic-sized problems, i.e., an urban road network with 324 intersections. Note that in real world examples, police forces typically focus on patrolling their own wards, which typically have the size of up to hundreds of intersections[5]. Thus, our algorithm can scale up to problems with realistic size.

7 Conclusions

We present a novel game-theoretic model for interdicting the escaping attacker on urban traffic networks. We show finding NE is NP-hard. Therefore, we propose an efficient double oracle framework, *EIGS*, based on novel MILPs for best response oracles and heuristic algorithms for better response oracles. To solve large-scale problems, we develop *RedAD* to significantly reduce the strategy space for both players. Experimental results show that our algorithms obtain a robust solution that significantly outperforms baselines and can scale up to realistic-sized problems.

Acknowledgments

This research is supported by NRF2015 NCR-NCR003-004. Long Tran-Thanh was supported by the EPSRC funded project EP/N02026X/1, Zhen Wang was supported by the NSFC funded project 61403059, and Jiarui Gan was funded by EPSRC International Doctoral Scholars Grant EP/N509711/1 and his department.

[4]We do not show the running time of *Snares* because it has the same solution quality as *Rugged*, which performs significantly worse than our algorithm in solution quality.

[5]E.g., Singapore Police Force has six divisions, and each division has up to dozens of neighbourhood police posts or police centers distributed in the city.

References

[Aboudolas *et al.*, 2009] Konstantinos Aboudolas, Markos Papageorgiou, and Elias B. Kosmatopoulos. Store-and-forward based methods for the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, 17(2):163 – 174, 2009.

[Adler *et al.*, 2002] Micah Adler, Harald Räcke, Naveen Sivadasan, Christian Sohler, and Berthold Vöcking. Randomized pursuit-evasion in graphs. In *ICALP*, pages 901–912, 2002.

[Basilico *et al.*, 2009] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, pages 57–64, 2009.

[Blum *et al.*, 2014] Avrim Blum, Nika Haghtalab, and Ariel D Procaccia. Learning optimal commitment to overcome insecurity. In *NIPS*, pages 1826–1834, 2014.

[Coogan and Arcak, 2015] S. Coogan and M. Arcak. A compartmental model for traffic networks and its dynamical behavior. *IEEE Transactions on Automatic Control*, 60(10):2698–2703, 2015.

[Daganzo, 1994] Carlos F. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269 – 287, 1994.

[Daganzo, 1995] Carlos F. Daganzo. The cell transmission model, part II: Network traffic. *Transportation Research Part B: Methodological*, 29(2):79 – 93, 1995.

[Fang *et al.*, 2016] Fei Fang, Thanh Hong Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Milind Tambe, and Andrew Lemieux. Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *IAAI*, pages 3966–3973, 2016.

[Guo *et al.*, 2016] Qingyu Guo, Bo An, Yair Zick, and Chunyan Miao. Optimal interdiction of illegal network flow. In *IJCAI*, pages 2507–2513, 2016.

[Iwashita *et al.*, 2016] Hiroaki Iwashita, Kotaro Ohori, Hirokazu Anai, and Atsushi Iwasaki. Simplifying urban network security games with cut-based graph contraction. In *AAMAS*, pages 205–213, 2016.

[Jain *et al.*, 2011] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, pages 327–334, 2011.

[Jain *et al.*, 2012] Manish Jain, Kevin Leyton-Brown, and Milind Tambe. The deployment-to-saturation ratio in security games. In *AAAI*, pages 1362–1370, 2012.

[Jain *et al.*, 2013] Manish Jain, Vincent Conitzer, and Milind Tambe. Security scheduling for real-world networks. In *AAMAS*, pages 215–222, 2013.

[Lebacque, 2005] Jean-Patrick Lebacque. Intersection modeling, application to macroscopic network traffic flow models and traffic management. In *Traffic and Granular Flow'03*, pages 261–278. 2005.

[Letchford and Conitzer, 2013] Joshua Letchford and Vincent Conitzer. Solving security games on graphs via marginal probabilities. In *AAAI*, pages 591–597, 2013.

[Manual, 1964] Traffic Assignment Manual. Bureau of public roads. *US Department of Commerce*, 1964.

[Nowakowski and Winkler, 1983] Richard Nowakowski and Peter Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2):235 – 239, 1983.

[Peng *et al.*, 2014] Wei Peng, Guohua Dong, Kun Yang, and Jinshu 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.

[Shieh *et al.*, 2012] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Ben Maule, and Garrett Meyer. PROTECT: An application of computational game theory for the security of the ports of the United States. In *AAAI*, pages 2173–2179, 2012.

[Skabardonis and Dowling, 1997] Alexander Skabardonis and Richard Dowling. Improved speed-flow relationships for planning applications. *Transportation Research Record: Journal of the Transportation Research Board*, 1572:18–23, 1997.

[Tambe, 2011] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned.* Cambridge University Press, 2011.

[Tsai *et al.*, 2010] Jason Tsai, Zhengyu Yin, Jun-young Kwak, David Kempe, Christopher Kiekintveld, and Milind Tambe. Urban security: Game-theoretic resource allocation in networked physical domains. In *AAAI*, pages 881–886, 2010.

[Vorobeychik *et al.*, 2014] Yevgeniy Vorobeychik, Bo An, Milind Tambe, and Satinder P. Singh. Computing solutions in infinite-horizon discounted adversarial patrolling games. In *ICAPS*, pages 314–322, 2014.

[Wang *et al.*, 2016] Zhen Wang, Yue Yin, and Bo An. Computing optimal monitoring strategy for detecting terrorist plots. In *AAAI*, pages 637–643, 2016.

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

[Yin *et al.*, 2014] Yue Yin, Bo An, and Manish Jain. Game-theoretic resource allocation for protecting large public events. In *AAAI*, pages 826–834, 2014.

[Yin *et al.*, 2015] Yue Yin, Haifeng Xu, Jiarui Gan, Bo An, and Albert Xin Jiang. Computing optimal mixed strategies for security games with dynamic payoffs. In *IJCAI*, pages 681–687, 2015.

[Zhao *et al.*, 2016] Mengchen Zhao, Bo An, and Christopher Kiekintveld. Optimizing personalized email filtering thresholds to mitigate sequential spear phishing attacks. In *AAAI*, pages 658–665, 2016.