Optimal Escape Interdiction on Transportation Networks

Youzhi Zhang\textsuperscript{1}, Bo An\textsuperscript{1}, Long Tran-Thanh\textsuperscript{2}, Zhen Wang\textsuperscript{3}, Jiarui Gan\textsuperscript{4}, Nicholas R. Jennings\textsuperscript{5}

\textsuperscript{1}School of Computer Science and Engineering, Nanyang Technological University, Singapore
\textsuperscript{2}Department of Electronics and Computer Science, University of Southampton, UK
\textsuperscript{3}School of Cyberspace, Hangzhou Dianzi University, China
\textsuperscript{4}Department of Computer Science, University of Oxford, UK
\textsuperscript{5}Departments of Computing and Electrical and Electronic Engineering, Imperial College, UK

\textsuperscript{1}\{yzhang137,boan\}@ntu.edu.sg, \textsuperscript{2}\texttt{ltt08r@ecs.soton.ac.uk}, \textsuperscript{3}\texttt{wangzhen@hdu.edu.cn}, \textsuperscript{4}jiarui.gan@cs.ox.ac.uk, \textsuperscript{5}\texttt{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 extension[s, 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 \( e \) in the start node \( \sigma_e \) to its end node \( \tau_e \). An entry 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 \( E_{\text{entry}} = \{ e | \sigma_e = v_{\infty} \} \) and \( E_{\text{exit}} = \{ e | \tau_e = v_{\infty} \} \), respectively. Accordingly, the remaining subset of the links is the set of internal links \( E_{\text{inter}} = \{ e | \sigma_e, \tau_e \in V \setminus \{ v_{\infty} \} \} \). 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 \( E^+_e = \{ e'| \sigma_{e'} = \sigma_e \} \) and \( E^-_e = \{ e'| \sigma_{e'} = \tau_e \} \), respectively.

**Traffic Flow:** Each entry link \( e \in E_{\text{entry}} \) is associated with a constant traffic demand \( d_e \geq 0 \) modelling the rate of vehicles entering the node \( \tau_e \) 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 \( v_{\infty} \), 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 \( f = \{ f_e, e \in E \} \) such that:

\[
 f_\varepsilon = d_\varepsilon \quad \forall e \in E_{\text{entry}} \quad (1)
\]

\[
 f_e = \sum_{e' \in E_e} f_{e'} R(e', e) \quad \forall e \in E_{\text{inter}} \cup E_{\text{exit}} \quad (2)
\]

For any given demand vector \( d \) and turning preference matrix \( R \), the induced equilibrium flow \( f \) 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]:

\[
 t_e = T_e(1 + 0.15 \times (f_e/C_e)^4) \quad (3)
\]

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 \( v_{\infty} \) via any exit node \( d \in D \subset \{ v | \exists e \in (v, v_{\infty}) \} \) after conducting some criminal activity at node \( v_0 \) in \( V \). There are \( m \) defender resources (police officers/teams), initially located at intersections \( v_0^1, \ldots, v_0^m \in V \). To only consider nontrivial problems, we assume that \( v_0^i \neq v_0^j \) for all \( r \in R = \{ 1, \ldots, m \} \). W.l.o.g., we assume that the event starts at time 0 and ends at time \( t_{\text{max}} > 0 \). 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 \( A = (a_1 = (v_0^1, 0), \ldots, a_j = (v_j, t_j^a), \ldots, a_{k-1} = (d \in D, t_{k-1}^a), a_k = (v_{\infty}, t_{k}^a) \leq t_{\text{max}}) \), 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 \( v_j \in N(v_{j+1}) \), where \( N(v_{j+1}) = \{ u \in V | (v_{j+1}, u) \in E \} \), and \( t_{j+1}^a \geq t_{j+1} + t(v_j, v_{j+1}) \). 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 \( A \) continuous. A mixed attacker strategy is denoted by \( y = (y_A) \), with \( y_A \) representing the probability that \( A \) is played.

A state of defender resource \( d_e \) is a tuple \( s^r = (v^r, t^r, i^r, t^r, o^r) \), representing the situation that \( d_e \) is protecting node \( v^r \) during \([r^i, r^o]) \). We assume that \( d_e \) can only stay at a node for a multiple of a constant time interval \( \delta \) (e.g., 10 seconds), i.e., \( r^o - r^i = k \delta \), where \( k \) is a nonnegative integer. To simplify our analysis, we assume that \( d_e \) captures the attacker at a node. Thus, for every two consecutive states \( s_i^r = (v_i^r, t_i^r, i^r, t^r, o^r) \) and \( s_{i+1}^r = (v_{i+1}^r, t_{i+1}^r, i_{i+1}^r, t_{i+1}^r, o_{i+1}^r) \), the minimal travel time between \( v_i^r \) and \( v_{i+1}^r \) can be easily 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., \( S = \{ S_r : r \in R \} \). The schedule for \( d_r \) is a sequence of states \( S^r = (s^r_1, \ldots, s^r_i, \ldots, s^r_k) \), where \( s^r_i = (v^r_0, 0, t^{r,\text{out}}_i) \) and \( s^r_k = (v^r_k, t^{r,\text{in}}_k, t^{r,\text{max}}) \). Moreover, \( d_r \) cannot drive to the external world \( v_\infty \) to capture the attacker, i.e., \( v^r_i \in V \setminus \{ v_\infty \} \), \( \forall s^r_i \in S^r \). The defender’s pure strategy space is denoted by \( S \). A defender’s mixed strategy is \( x = \langle x_S \rangle \), 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 \( a_j = (v_j, t^j) \) and \( s^r_i = (v^r_i, t^{r,\text{in}}_i, t^{r,\text{out}}) \), we say the attacker is captured by \( d_r \) at node \( v^r_i \) if \( v^r_i = v_j \) and \( t^{r,\text{in}}_i \leq t^a_j \leq t^{r,\text{out}} \). In such a case, \( x_{s^r_i, a_j} = 1 \), otherwise \( x_{s^r_i, a_j} = 0 \). 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:

\[
U_d(S, A) = \begin{cases} 
0 & \text{if } \exists 2 x_{s^r_i, a_j} = 1, a_j \in A, s^r_i \in S^r, r \in R \\
-1 & \text{otherwise}
\end{cases}
\] (4)

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

**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 maximin equilibrium given the zero-sum assumption. Thus, the optimal mixed strategy \( x \) of the defender can be computed by solving the following linear program (LP).

\[
\begin{align*}
\text{max} & \quad U^* \\
\text{s.t.} & \quad U^* \leq U_d(x, A) \quad \forall A \in A \\
& \quad \sum_{S \in S} x_S = 1, x_S \geq 0 \quad \forall S \in S
\end{align*}
\] (5) (6) (7)

### 4 Solution Approach

Solving the LP in Eqs. (5)–(7) under the exponentially large \( S \) and the continuous \( A \) is challenging. To handle this challenge, we develop a novel double oracle algorithm \( \text{EIGS} \) (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 \( C_1 \land C_2 \land \cdots \land C_k \). 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 \( \iff \)) there exists a defender pure strategy intercepting all attacker pure strategies if and only if (denoted as \( \iff \)) the attacker gets caught with probability 1 in NE (under the mixed strategy setting). Particularly, we will focus the proof on \( \iff \), as \( \iff \) 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 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 \( 0 \) on the top represents the source node \( v^0 \) 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 \( x_i = \text{true} \) or \( \text{false} \), 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 \( v^0 \) 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: \( x_i = \text{true} \) if the literal is \( \neg x_i \); and \( x_i = \text{false} \) if the literal is \( x_i \). Namely, for example, if in the blue box \( x_i \) is chosen to be \( \text{true} \), the \( \text{false} \) 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 attacker pure strategy exiting from the red boxes) only when one of its literals is made \( \text{true} \) by choices of directions in the blue boxes.

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

1. The \( \implies \) direction. Suppose there exists a satisfying assignment for the 3-SAT problem. What the defender can do...
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 "⇐" 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.

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 $S'$ and $A'$, which are small sets of pure strategies for both players. Then, CoreLP computes the maximin strategies $\bar{x}$ and $\bar{y}$ for the restricted game $(S', A')$ (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 $(S', A')$ 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.

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 $(x, y)$ provided by CoreLP, $S$ is added to $S'$ if $U_d(S, y) > U_d(x, y)$. 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 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 stop time.

This procedure is represented by the following MILP:

\[
\begin{align*}
\max & - \sum_{A \in A'} (1 - z_A) y_A \\
\text{s.t.} & \quad s_{r,v_0}^x = 1, \quad \forall v \in V \setminus \{v_{\text{in}}\} : s_{r,v}^x = 1, \quad \forall r, i \\
& \quad \omega_{r,i,(v,u)} \leq \min(s_{r,i,v}, s_{r+1,i,u}) \quad \forall r, i, v, u \\
& \quad \omega_{r,i,(v,u)} \geq s_{r,i,v}^x + s_{r+1,i,u} - 1 \quad \forall r, i, v, u \\
& \quad t_{r,i}^{\text{in}} = 0, \quad t_{r,i}^{\text{out}} = t_{\text{max}}, \quad t_{r,i}^{\text{out}} = t_{r,i}^{\text{in}} + \kappa_{r,i} \delta \quad \forall r, i \\
& \quad t_{r,i}^{\text{out}} = t_{r,i}^{\text{in}} + \sum_{v,u} \text{dist}(v, u) \omega_{r,i,(v,u)} \quad \forall r, i \\
& \quad -M \alpha_{r,i} \leq t_{r,i}^{\text{in}} - t_{r,i}^{\text{out}} \leq M(1 - \alpha_{r,i}) \quad \forall r, i, A, j \\
& \quad -M \beta_{r,i} \leq t_{r,i}^{\text{out}} - t_{r,i}^{\text{out}} \leq M(1 - \beta_{r,i}) \quad \forall r, i, A, j \\
& \quad \gamma_{r,i} \leq \alpha_{r,i} + \beta_{r,i} + s_{r,i}^x / 3 \quad \forall r, i, A, j \\
& \quad \gamma_{r,i} \leq \alpha_{r,i} + \beta_{r,i} + s_{r,i}^x / 2 \quad \forall r, i, A, j \\
& \quad z_A \leq \sum_{r,i} \gamma_{r,i} \quad \forall A \\
& \quad s_{r,i,v}^x, \omega_{r,i,(v,u)}, \alpha_{r,i}, \beta_{r,i}, \gamma_{r,i}, z_A \in \{0, 1\}
\end{align*}
\]

Algorithm 1: EIGS

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

Algorithm 2: betterDO ($y$)

1. $A' \leftarrow A'$; \quad $\forall r \in R : S' \leftarrow \{(v_0, 0, 0)\}$, $L' \leftarrow 1$, $t_{r_{\text{in}}}^I = 0$;
2. while $A' \neq \emptyset$
   3. for $r \in R$ do
      4. for $v \in V \setminus \{v_{\text{in}}\}$ do
         5. $t_{r,i}^{\text{in}} = t_{r,i}^{\text{out}} = t_{\text{max}}, t_{r,i}^{\text{out}} = t_{r,i}^{\text{in}} + \kappa_{r,i} \delta \quad \forall r, i$
         6. $t_{r,i}^{\text{in}} = t_{r,i}^{\text{in}} + \sum_{v,u} \text{dist}(v, u) \omega_{r,i,(v,u)} \quad \forall r, i$
         7. $-M \alpha_{r,i} \leq t_{r,i}^{\text{in}} - t_{r,i}^{\text{out}} \leq M(1 - \alpha_{r,i}) \quad \forall r, i, A, j$
         8. $-M \beta_{r,i} \leq t_{r,i}^{\text{out}} - t_{r,i}^{\text{out}} \leq M(1 - \beta_{r,i}) \quad \forall r, i, A, j$
         9. $\gamma_{r,i} \leq \alpha_{r,i} + \beta_{r,i} + s_{r,i}^x / 3 \quad \forall r, i, A, j$
        10. $\gamma_{r,i} \leq \alpha_{r,i} + \beta_{r,i} + s_{r,i}^x / 2 \quad \forall r, i, A, j$
      11. $z_A \leq \sum_{r,i} \gamma_{r,i} \quad \forall A$
      12. $s_{r,i,v}^x, \omega_{r,i,(v,u)}, \alpha_{r,i}, \beta_{r,i}, \gamma_{r,i}, z_A \in \{0, 1\}$
9. return $(S', r, r \in R)$. 
Algorithm 3: betterAO (x)

1. ∀ v ∈ V \ {v₀}, Δx[v] ← ∞, tʰ[v] ← ∞;
2. Δx[v₀] ← 0, tʰ[v₀] ← 0, \overline{S} \overline{v₀} ← S', Q ← V;
3. while Q ≠ ∅ do
   4. v ← arg min\overline{Q} Δx[w];
   5. if v = v∞ then break;
   6. Q ← Q \ {v};
   7. for each neighbor u of v do
      8. t ← arg min\overline{r(u,v)} Δx[w] + \Phi(t'); :=\{S_S,S_F|\Delta x[w] + \Phi(t');
      9. if Δx[w] + \Phi(t') < Δx[v] + \Phi(t) then
         10. Δx[w] ← Δx[w] + \Phi(t);
         11. tʰ[w] ← tʰ[w] + t;
         12. \overline{S}_w ← \overline{S}_w \cup \{S|u ∈ S, tʰ[w] ∈ [tʰ(u,0),tʰ(u,\infty)];
      14. A ← {(v₀,0), (v₁,tʰ[v₁]),..., (vₙ,tʰ[vₙ])} such that
         15. return A.

\[ κ_{r,i} \in \mathbb{Z}_{≥ 0}, tʰ^{in}, tʰ^{out} \in [0, t_{max}] \] (20)

Here, sᵢᵣ, ᵐᵣ is 1 indicates that dᵣ arrives at v by the i-th state of S'. Eq.(9) indicates that dᵣ starts at v₀ and stays on only one node in any state of S'. In Eq.(10) and (11), the 0/1 variable ωᵣᵢ,ᵣ,u,v,ᵣ encodes whether there is a path u-v between the i-th state and (i + 1)-th state of S'. 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 dᵣ stays on the i-th state is κ_{r,i}. Eq.(13) ensures that dᵣ travels one stop to another through the shortest path. With Eqs.(14)–(18), zₐ determines whether the attacker using A is caught. Specifically, (vᵢ, ᵐᵣ,i) represents the attacker’s j-th state in A; γᵣ,i,j indicates whether the attacker using A, meets dᵣ at his j-th state’s position while dᵣ is at her i-th stop; ρᵣ,i,j indicates whether the attacker arrives at his j-th state’s position after dᵣ arrives (respectively, before dᵣ 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{A} stores the attacker’s paths that are not caught by any S’. Initially, dᵣ is at her initial position v₀ (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⟩ ∈ R × V, and check how much value dᵣ 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 tʰ[v] are intercepted, and the corresponding values are added to Δ_{y}[r][v]), dᵣ 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{A}, and tʰ^{in, out} is updated to the time that this movement can intercept as many attacker paths as possible. dᵣ stays at a node for n₀ δ (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 \langle x, y \rangle provided by CoreLP, we are added to A’ if Uₐ (x, y) > Uₐ (x, y). This can be formulated as an MILP, i.e., bestAO. It first constructs a path attaining 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:

\[ \max \sum_{S \in S'} (1 - z_S) x_S \] (21)

s.t. \[ A_1 v₀ = 1, A_{L_{max}} vₙ = 1, \exists v ∈ V A_j v = 1 \ \forall j \] (22)
\[ A_{j+1}, vₙ ≥ A_{j,v} \ \forall j \] (23)
\[ \sum_{u ∈ V(j)} A_{j+1,u} ≥ A_{j,v} \ \forall v ∈ V, j \] (24)
\[ \omega_j (v,u) ≤ \min (A_j,v, A_{j+1,u}) \ \forall (v,u) ∈ E, j \] (25)
\[ \omega_j (v,u) ≥ A_{j,v} + A_{j+1,u} - 1 \ \forall (v,u) ∈ E, j \] (26)
\[ t₀ = 0 \] (27)
\[ t_{j+1} ≥ t₀ + \sum_{(v,u) ∈ E} t(v,u) ω_j (v,u) \ \forall j \] (28)
\[ -M ω_j (r,i,j) ≤ t₀^{in} - t_j^{out} ≤ M(1 - \alpha_{r,i,j}) \ \forall S, r, i, j \] (29)
\[ -M ω_j (r,i,j) ≤ t₀^{in} - t_j^{out} ≤ M(1 - \beta_{r,i,j}) \ \forall S, r, i, j \] (30)
\[ z_S ≥ \alpha_{r,i,j} + \beta_{r,i,j} + A_j vₙ - 2 \ \forall S, r, i, j \] (31)
\[ A_{j,v}, A_{j,r,i}, A_{j,r,i}^{in}, A_{j,r,i}^{out}, ω_j (v,u), z_S ∈ [0,1], t₀^{in} ∈ [0, t_{max}] \] (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 v₀, terminates at vₙ, and only arrives at one node in each state. Eqs.(23)–(24) fix the attacker at the sink node vₙ in the following states after he reaches vₙ 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 ω_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 x. Variable Δx[v] maintains, for all v ∈ V, the shortest so-far-known acyclic path from v₀ to v. In the main loop, the node with minimum Δx[v] is extracted from Q (Line 4). Then in Lines 7–13, Δx of each neighbor of v is updated with Δx[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 (ϕₜ (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 bet-
ter oracles. However, 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 signif-
ically 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

is reduced to , which satisfies the condition that for every
two consecutive states (or) of each attacker strategy
. That is, the attacker travels with the maximum speed. is reduced to, which satisfies the condition that for each
defender strategy in the schedule has the following form:
, where . That is, the defender goes to some node with maximum speed and waits there. Obviously,
and. Now we have the following property about the relation of and.

Proposition 2. Given a defender strategy with its support set, there is an attacker optimal strategy
.

Proof. Suppose that there is not an optimal strategy. This means that if is an optimal strategy then for all
and. In any strategy in , the schedule has the following form:
, where . Keeping the nodes’ sequence order of and we can obtain an’s strategy as
, where . From the definition of and, we have
. Moreover, we define
 and
. Furthermore, we define
 and
. So the attacker’s utility
, which causes a contradiction.

Algorithm 4: advancedAO(x)

| \text{1} | V_x \leftarrow \{v_i^r \mid x_S > 0, (v_i^r, t^i_{r,\text{in}}, t^i_{r,\text{out}}) \in S' \}; |
| \text{2} | V_f \leftarrow \text{DFS}(G, V_X, v_0^r), V_{f,x} \leftarrow \text{V}_x \cap V_f; |
| \text{3} | \text{if } v_\infty \in V_f \text{ then } A^* \leftarrow \{P(\text{dist}(v_j, v_{\infty}))(v_0^r, v_\infty)) \}; |
| \text{4} | \text{else} |
| \text{5} | V_b \leftarrow \text{DFS}(G, \{u, v \mid (u, v) \in E \}), V_{x,b}; |
| \text{6} | V_{x,b} \leftarrow V_x \cap V_b; |
| \text{7} | E \leftarrow E \cup \{(v_0^r, v) \mid v \in V_{x,b}\}; |
| \text{8} | t_{\infty,v} \leftarrow \text{dist}(v_j, v_{\infty}),(v_0^r, v), \forall v \in V_f; |
| \text{9} | v_{\infty,v} \leftarrow \text{dist}(v_j, v_{\infty}),(v_0^r, v), \forall v \in V_{x,b}; |
| \text{10} | V_c \leftarrow (V \setminus (V_f \cup V_b)) \cup (V_f \cup V_{x,b} \cup (v_0^r, v_{\infty})); |
| \text{11} | E_c \leftarrow \{u, v \mid (u, v) \in E\}; |
| \text{12} | A \leftarrow \{\text{RedbestDO}(x)\}; |
| \text{13} | \text{if } A \neq \emptyset \text{ then } A^* \leftarrow \{P(\text{dist}(v_j, v_{\infty}))(v_0^r, v_\infty)) \cup (v_0^r, t^j_1) \}; |
| \text{14} | \text{return } A^*; |

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

5.2 An Approximate Algorithm of EIG

Based on the reduced game (, ), we can redesign the or-
acles in Algorithm 1. Here the new best DO, RedbestDO, has the same objective function as bestDO and it has the following constraints:

\[ \sum_{v \in V \setminus \{v_\infty\}} s^r_v = 1 \quad \forall r \]  
\[ \sum_{v \in R} s^r_v \leq 1 \quad \forall v \in V \setminus \{v_\infty\} \]  
\[ z_A \leq \sum_{j \in A, t \in R} a^j \leq \sum_{j \in A} s^A_v \quad \forall A \]  
\[ s^r_v, z_A \in \{0, 1\} \]  

is 1 indicates that resource is deployed to node . Eq.(33) ensures that each resource will be assigned to one node, which is protected by at most one resource (Eq.(34)). indicates whether the attacker’s path is intercepted by the defender. is 1 only when there is one node on the path where there is one resource reaching there before the attacker (reflected 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 reflect the fact that the attacker will travel with maximum speed. Similarly, the new best AO, RedbetterAO, is obtained from betterAO by modifying Line 8 in Algorithm 3 to: if \( \exists S' s.t. z_{S'}(x,\text{dist}(v_i^r, v_{\infty}))) = 1, \forall S' \in S_\infty, \) which will dramatically slow it down. To speed it up, we further reduce the attacker strategy space by graph contraction.

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

\text{advancedAO:}
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 in the original network.

This contraction procedure is shown in Algorithm 4. \( V_x \) stores nodes in the support set of \( x \) (Line 1). If \( v_f \), a set of nodes that can be reached from \( v_0 \) without traversing the nodes in \( V_{x,v} \), is given by DFS \((G,V_x,v_0)\) (Line 2), where DFS \((G,V_x,v_0)\) is obtained from the classic Depth First Search algorithm by ending each path at the nodes in \( V_x \). That is, a node \( v \) is pushed into the stack only if \( v \) is unvisited and \( v \notin V_x \). If \( v_\infty \in V_f \), then \( P(dist((v_f,v_\infty))((v_0,v_\infty))) \) is an optimal path that will not be intercepted by the defender (Line 3). Here \( P(dist((v_f,v_\infty))((v_0,v_\infty))) \) is the shortest path from \( v_0 \) to \( v_\infty \) on the network \((V_f,\{(u,v) | (u,v) \in E, u \in V_f \wedge v \in V_f \})\) with corresponding travel time \( dist((v_f,v_\infty))((v_0,v_\infty)) \). Otherwise, the network \( G \) is contract-ed (Lines 5–11) and RedbestAO(\( x \)) is called for the contract-ed network (Line 12). \( V_b \) stores nodes reached from \( v_\infty \) 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, 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 \((S,A)\) by replacing RedbestAO with bestAO in Line 12 of Algorithm 4. Actually, we also propose a two-stage approach for the full game \((S,A)\) which uses RedAD to initialize \((S',A')\) 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 \( v_\infty \) 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, \( v_0^f \) 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 path-ways. 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 \( v_0^f - v_\infty \) 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 cut-s. 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 \( x \) in Rugged, RedAD, and bettAD is assessed in terms of \( U_a(x, A) \), 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
\( |D| = L \) 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 \textit{Rugged} performs the worst in all cases because it is designed without considering players’ travel time on roads. Meanwhile, \textit{EIGS} performs the best. \textit{bettAD} performs significantly worse than \textit{EIGS}. However, \textit{RedAD} achieves good solution quality, which has similar performance to \textit{EIGS} and significantly outperforms \textit{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 \textit{EIGS} in the presence of uncertainty, we add noise \( \theta \) to the travel time. So the actual travel time \( t_e = t_e \times (1 + \theta) \). Figure 3(e) shows the attacker utility of \textit{EIGS} under different degrees of uncertainty. Here we can see that while the performance of \textit{EIGS} decreases as \( \theta \) is increased, it still outperforms \textit{Rugged}.

### 6.2 Scarcity

We compare the scalability of \textit{EIGS} with a baseline, \textit{EIGS-O}, which runs \textit{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 \textit{RedAD} with another baseline, \textit{Reduce}, where \textit{RedAD} calls \textit{RedbestAO} instead of \textit{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 \textit{EIGS-O} does not scale up to the network with \( 8 \times 8 \) intersections, so we do not show the result of \textit{EIGS-O} in Figures 3(g) and 3(h). Results show that: 1) Our new double oracle framework outperforms the old one by comparing \textit{EIGS} with \textit{EIGS-O}; 2) \textit{RedAD} outperforms \textit{EIGS} and \textit{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) \textit{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, \textit{EIGS}, based on novel MILPs for best-response oracles and heuristic algorithms for better response oracles. To solve large-scale problems, we develop \textit{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 Jiariu Gan was funded by EPSRC International Doctoral Scholars Grant EP/N09711/1 and his department.

\(^4\)We do not show the running time of \textit{Snare} because it has the same solution quality as \textit{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


