

# Urban rail patrolling: a game theoretic approach

Abdolmajid Yolmeh<sup>1</sup>  · Melike Baykal-Gürsoy<sup>2</sup>

Received: 8 September 2017 / Accepted: 16 January 2018 / Published online: 10 February 2018  
© Springer Science+Business Media, LLC, part of Springer Nature 2018

**Abstract** Patrol scheduling is a critical operational decision in protecting urban rail networks against terrorist activities. Designing patrols to protect such systems poses many challenges that have not been comprehensively addressed in the literature of patrol scheduling so far. These challenges include strategic attackers, dynamically changing station occupancy levels and human resource related limitations. In this paper, we develop a game theoretic model for the problem of scheduling security teams to patrol an urban mass transit rail network. Our main objective is to minimize the expected potential damage caused by terrorist activities while observing scheduling constraints. We model this problem as a non-cooperative simultaneous move game between a defender and an attacker. We then develop column generation based algorithms to find a Nash equilibrium for this game. We also present a lower bound for the value of the game which can be used to terminate the column generation algorithm when a desired solution quality is reached. We then run computational experiments to investigate the efficiency of the proposed algorithms and to gain insight about the value of the patrolling game. Our results show the efficiency of the proposed algorithms. Finally, we present results for the case of a real urban rail network.

**Keywords** Patrolling game · Column generation · Zero sum games · Urban rail

---

✉ Abdolmajid Yolmeh  
majid\_yolmeh@yahoo.com

Melike Baykal-Gürsoy  
gursoy@soe.rutgers.edu

<sup>1</sup> Industrial and Systems Engineering Department, Rutgers University, 96 Frelinghuysen Rd, Piscataway, NJ 08854, USA

<sup>2</sup> Industrial and Systems Engineering Department, RUTCOR, CAIT, Rutgers University, 96 Frelinghuysen Rd, Piscataway, NJ 08854, USA

## Introduction

Protecting critical infrastructures against terrorism is one of the top priorities in homeland security (Moteff 2005). Among these critical infrastructures, transportation systems, which serve 32 million passengers every day in the United States, are critical for supporting the national security and economic well-being. For decades, public transit systems around the world have been considered as a principal target for terrorist acts (Strandberg 2013; Shvetsov and Shvetsova 2017; Shvetsov et al. 2017). Among these systems, airliners are considered to be hard targets due to implementation of security checkpoints and increased security measures. Over the years, the number of attempted hijackings and bombings has declined gradually (although the public areas of airports still remain vulnerable). Unlike airlines, where security checkpoints screen passengers and luggage, mass transit options like subways, passenger trains, and buses, are designed to be easily accessible and are therefore harder to protect. Ground transportation systems, which often include enclosed spaces packed with people, could prove attractive targets for terrorists. Therefore, such open transit systems are considered to be soft targets for the terrorists. The attacks in Brussels and Istanbul along with many other incidents indicate that terrorists tend to target such large crowds to cause mass human casualties in addition to panic and chaos. These incidents along with many others highlight the importance of protecting such infrastructures. The threat to these infrastructures could be substantially reduced by analyzing the risk associated with attack to each infrastructure component, mitigation planning and designing efficient response policies. This includes assigning security teams and designing efficient patrol schedules to protect vulnerable areas. Patrol scheduling involves the process of constructing optimized work timetables for security staff in order to minimize the potential damage of possible attacks. Designing patrols to protect public transport systems and other soft targets poses unique challenges that have not been properly addressed in the literature of patrol scheduling so far. One of these challenges is the dynamic nature of crowd size inside these systems. Because the adversary's primary objective is to inflict human casualties, the attacker's payoff value for each station depends on the number of people residing in the station. These numbers may change over time. Another challenge is to develop schedules that observe the constraints regarding human resources, for example, the generated schedules may be required to include breaks for the security teams and these breaks should not be consecutive. Moreover, efficient methods are needed to design patrols for a general network. In this paper, we address these challenges in a patrolling game setting.

Patrol scheduling has many applications in real life. Police officers patrol cities; security officers patrol terminals at airports and transportation centers; security guards patrol museums and shopping malls. The first studies on the patrolling problems were conducted in 1970s, when several studies focused on allocating patrols to different areas to optimize performance measures such as patrol delays, average waiting time and total response time (Larson 1972; Olson and Wright 1975; Chaiken and Dormont 1978; Chelst 1978). These studies assume that crime frequency in different regions remain fixed and known to the patroller. However, since the adversaries can be strategic in their attacks, they may change the location and timing of their attacks in response to the patroller's strategy. Therefore, game theoretic analysis of such attacks yields more realistic results. To this end, several researchers developed game theoretic models for the patrolling problem. One of the first models applying game theory in patrol scheduling, named ARMOR (Paruchuri et al. 2005,

2006, 2007, 2008; Pita et al. 2008), casts the patrolling/monitoring problem as a Bayesian Stackelberg game in order to aid the security agent decide where to allocate inspection gates in an airport. ARMOR was successfully deployed at the Los Angeles International Airport (LAX) to randomize checkpoints on the roadways entering the airport. Another stream of research focuses on protecting the urban rail systems against fare evasion (Jiang et al. 2012, 2013; Yin et al. 2012; Delle Fave et al. 2014, 2015). For example, Jiang et al. (2012) developed a Stackelberg game to generate patrolling strategies to inspect passenger tickets in an urban rail transportation system. They use this model to randomly deploy security personnel for fare inspection in the Los Angeles Metro Rail System. The obtained strategy seems to yield high levels of revenue by deterring fare evasion. Another model, named IRIS (Intelligent Randomization In Scheduling) is developed by Tsai et al. (2009) to assist Federal Air Marshal Service (FAMS) with randomly scheduling air marshals on flights. FAMS deployed IRIS in limited use since October 2009. There have been some studies on designing patrol strategies for transit systems. Varakantham et al. (2013) study the problem of designing randomized patrols to enhance security of mass rapid transit systems. They formulate this problem as a Stackelberg security game to optimize the patrols in terms of travelled distance. They then apply their model on a real life case of rail network in Singapore. To solve this problem, they apply a two-stage solution algorithm; in the first stage, they obtain coverage probabilities using a Stackelberg security game. Then in the second stage, they use these coverage probabilities to derive an actual patrol. They assume that every station is accessible from every other station; therefore, it is possible to generate feasible patrols for every coverage vector. However, the accessibility assumption may not always be valid; the distance between stations, especially the ones on different lines, can be significantly high. Therefore, some of the coverage vectors may lead to patrols that cannot be conducted within the given time horizon. Delle Fave et al. (2014) develop another model for patrolling transit systems. They propose a Multi-Operation Patrol Scheduling System (MOPSS), a new system to generate patrols for transit system to tackle multiple types of illegal activities including fare evasion, terrorism and crime. However, they do not consider human resource considerations such as break times etc. Moreover, the counter terrorism module in their model assumes that the targets have fixed values. This may not be a realistic assumption due to the dynamic nature of crowd sizes inside transportation systems.

The most relevant paper to our study is conducted by Lau et al. (2016). They study the problem of generating patrolling schedules for security teams to patrol a mass rapid transit rail network of an urban area. Their objective is to deploy patrolling units to the stations in different time units so that some scheduling and security related constraints are satisfied. They develop various mathematical models and apply it to a real rail network. The shortcoming of their model is that, because it is not a game based model, it is not designed to generate randomized schedules. To remedy this, they propose to generate randomized solutions by varying some of the problem parameters such as the start time and break time for each team. However, this may lead to sub-optimal patrol schedules. Moreover, the adversary's attack probabilities are assumed to be fixed and known. Again, this is not a realistic assumption because terrorists can change the location and timing of their attacks in response to the patrol strategy. Game theoretic models are designed to generate randomized strategies and are more suited for such adversarial settings.

In this paper, we develop a game theoretic model to schedule security teams in order to protect an urban rail network against terrorist attacks. We develop column generation

based algorithms to efficiently solve the game under general network structures. The computational results show the efficiency of the proposed algorithms. The rest of this paper is organized as follows. In Section “[Problem description](#)”, the problem under consideration is described. The proposed column generation approach is explained in Section “[Column generation procedure](#)”. In Section “[A heuristic solution approach for the pricing sub-problem](#)”, a heuristic algorithm is presented to efficiently solve the pricing sub-problem. Section “[Numerical experiments](#)” presents the numerical results along with an application of the model on a real case. Finally, the main conclusion of the paper and suggestions for future research are discussed in Section “[Conclusions and future research](#)”.

### Problem description

The patrolling problem considered in this paper involves scheduling a set of security teams  $I$  to protect a set of stations  $J$  on an urban rail network over a time horizon of  $T$  time periods. The time periods can represent the working hours in a day. Figure 1 shows an example of such an urban rail network in Philadelphia.

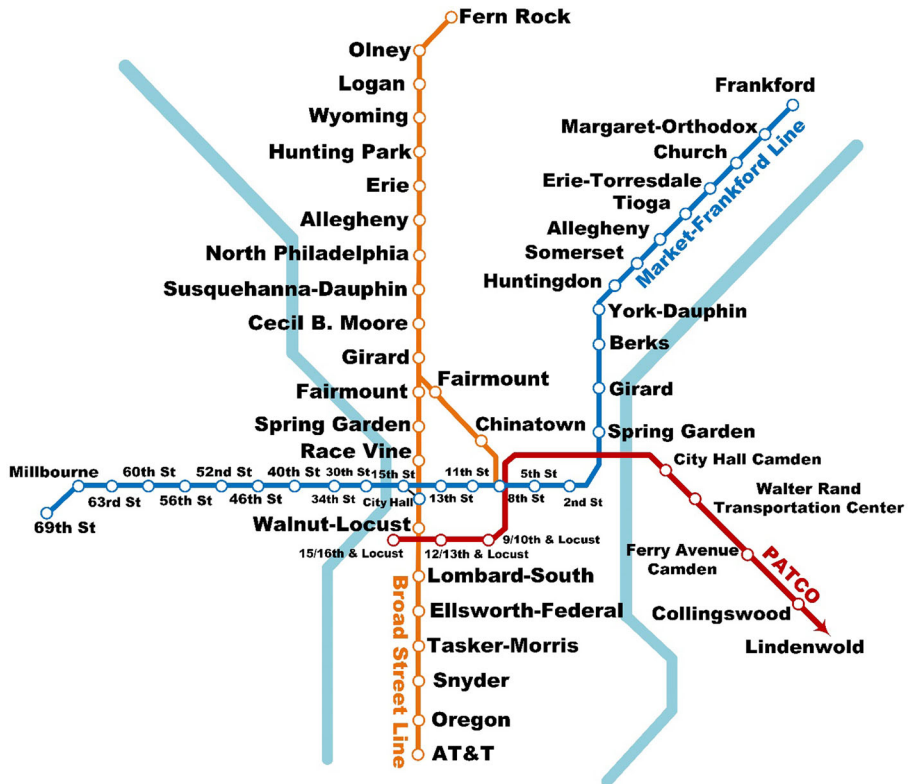


Fig. 1 Urban rail network in Philadelphia

Most of these networks consists of multiple lines that are connected via interchange stations. We model the patrolling problem as a simultaneous game between a defender and a single adversary. The defender controls the security teams and chooses a schedule to minimize the damage from the adversary’s attack, while the adversary chooses the station and the time to attack. A pure strategy for the adversary is represented by a pair  $(j, t)$  which indicates the station  $j$  and time  $t$  to attack. A pure strategy for the defender is a schedule that determines the complete course of actions for all teams throughout the time horizon. These actions include patrolling different stations or taking a break. Each team should have a pre-specified number of breaks and these breaks should not be consecutive or scheduled at the beginning or end of the time horizon. The payoffs to the players are determined by the expected damage to the network. While the defender wants to minimize the expected damage, the adversary wants to maximize it. We denote the value of station  $j$  at time  $t$  by  $c_{jt}$ , this value can represent the number of affected people if a successful attack is launched. If the adversary decides to attack station  $j$  at time  $t$  and the station is not being patrolled by a security team, the adversary wins a payoff of  $c_{jt}$ . On the other hand, if the station is being patrolled by a security team at the time of attack, with some probability  $d_j$  the attack will be thwarted and with probability  $1 - d_j$  the attack will be successful. Therefore, the expected damage is  $(1 - d_j)c_{jt}$ . We represent the set of all possible schedules by  $\mathcal{S}$  and index them by  $s$  with  $s = 1, 2, \dots, |\mathcal{S}|$ .

The players play a zero-sum matrix game where the defender plays as the row player; with the set of all possible schedules constituting the rows of the matrix. The adversary is the column player, with the set of all possible attack pairs  $(j, t)$  constituting the columns of the game matrix. The game can be solved by generating all of the possible strategies for both players. However, for the games of large size, the set of all possible strategies is exponentially large for the defender and generating all of them becomes impractical. In the next section, we develop an efficient column generation approach to obtain the Nash equilibrium for this game.

### Column generation procedure

In this section, we develop a column generation algorithm to obtain the Nash equilibrium point for the patrolling game described in Section “[Problem description](#)”. We can write the linear program (LP) to obtain the Nash equilibrium of this game as:

$$\text{Minimize } u \tag{1}$$

$$\text{subject to } u \geq \sum_{s \in \mathcal{S}} c_{jt} (1 - w_{jt}^s d_j) x_s, \quad \forall j, t, \tag{2}$$

$$\sum_{s \in \mathcal{S}} x_s = 1, \tag{3}$$

$$x_s \geq 0, \quad \forall s \in \mathcal{S}. \tag{4}$$

In this formulation  $x_s$  is the probability of using schedule  $s \in \mathcal{S}$  in the defender's mixed strategy.  $w_{jt}^s$  is a binary parameter that is equal to 1 if schedule  $s$  interrupts an attack strategy  $(j, t)$ , zero otherwise. In the terminology of column generation, this LP is called the linear programming master problem (LPM). Note that each column in the LPM corresponds to a schedule. In general the set  $\mathcal{S}$ , may be exponentially large; however, the number of non-zero variables (the basic variables) in the LPM is equal to the number of constraints i.e. the total number of  $(j, t)$  pairs:  $T|J|$ . Therefore, even though the number of possible schedules  $\mathcal{S}$  is large, only a small number of them is used in the Nash equilibrium. The column generation algorithm uses this idea to start with a subset  $\mathcal{S}' \subseteq \mathcal{S}$  of columns and generate columns as needed. The starting subset  $\mathcal{S}'$  could be any set of feasible schedules. Using the restricted set of schedules,  $\mathcal{S}'$ , we obtain the following LP:

$$\text{Minimize } u \quad (5)$$

$$\text{subject to } u \geq \sum_{s \in \mathcal{S}'} c_{jt} \left(1 - w_{jt}^s d_j\right) x_s, \quad \forall j, t, \quad (6)$$

$$\sum_{s \in \mathcal{S}'} x_s = 1, \quad (7)$$

$$x_s \geq 0, \quad \forall s \in \mathcal{S}'. \quad (8)$$

This problem is called Restricted LPM (RLPM). Dual of the RLPM is:

$$\text{Maximize } v \quad (9)$$

$$\text{subject to } v \geq \sum_{j,t} c_{jt} \left(1 - w_{jt}^s d_j\right) q_{jt}, \quad \forall s \in \mathcal{S}', \quad (10)$$

$$\sum_{j,t} q_{jt} = 1, \quad (11)$$

$$q_{jt} \geq 0, \quad \forall j, t. \quad (12)$$

Where  $q_{jt}$  is the dual variable associated with constraint  $(j, t)$  in the RLPM. Next step is to find a column (schedule) in  $\mathcal{S}'$  that could improve the current optimal solution of the RLPM. Given the optimal dual solution  $q_{jt}$  of the RLPM, the reduced cost of column  $s \in \mathcal{S}'$  is  $\sum_{j,t} c_{jt} (1 - w_{jt}^s d_j) q_{jt} - v$ . Based on the concept of duality in linear programming, optimality of the RLPM is equivalent to feasibility of the dual. Therefore, patrols that violate the constraint  $v \leq \sum_{j,t} c_{jt} (1 - w_{jt}^s d_j) q_{jt}$  can improve the current optimal solution. Therefore, we should look for a column (schedule)  $s$  such that:  $\sum_{j,t} c_{jt} (1 - w_{jt}^s d_j) q_{jt} - v < 0$ . Note that  $q_{jt}$  are fixed, and the problem is to find a schedule  $s$  with  $w_{jt}^s$  such that:  $\sum_{j,t} c_{jt} (1 - w_{jt}^s d_j) q_{jt} - v < 0$ . This problem is called the pricing sub-problem. The pricing sub-problem involves finding a column, i.e. a schedule, with a negative reduced cost. Subsection “[Mathematical formulation to solve the pricing sub-problem](#)” develops a mathematical program to solve the pricing sub-problem. Subsection “[Overall column generation procedure and a dual bound](#)” presents the overall column generation algorithm and a lower bound on the value of the game.

### Mathematical formulation to solve the pricing sub-problem

In this section, we develop a mathematical formulation to solve the pricing sub-problem. Here is a list of parameters and variables used to formulate the pricing sub-problem:

- $a_{jj'}$  Binary parameter, is equal to 1 if it is feasible to visit stations  $j$  and  $j'$  consecutively; 0 otherwise.
- $M$  A big enough number.
- $x_{ijt}$  Binary variable, 1 if team  $i$  patrols station  $j$  at time  $t$ ; zero otherwise.
- $y_{ijt}$  Binary variable, 1 if team  $i$  takes a break at station  $j$  at time  $t$ ; zero otherwise.
- $w_{jt}$  Binary variable, 1 if a team patrols station  $j$  at time  $t$ , zero otherwise.

Using this notation, the pricing sub-problem is formulated as follows:

$$\text{Minimize } \sum_j \sum_t c_{jt} w_{jt} q_{jt} d_j \tag{13}$$

$$\text{subject to } \sum_j (x_{ijt} + y_{ijt}) = 1, \quad \forall i, t, \tag{14}$$

$$x_{ijt} + y_{ijt} + x_{ij',t+1} + y_{ij',t+1} \leq a_{jj'} + 1, \quad \forall i, j, j', t, \tag{15}$$

$$\sum_i x_{ijt} \leq M w_{jt}, \quad \forall j, t, \tag{16}$$

$$w_{jt} \leq \sum_i x_{ijt}, \quad \forall j, t, \tag{17}$$

$$\sum_{j,t} y_{ijt} = 2, \forall i, \quad (18)$$

$$\sum_j y_{ijt} + \sum_j y_{ij,t+1} \leq 1, \forall i, t, \quad (19)$$

$$\sum_j y_{ijt} = 0, \forall i \in I, t \in \{i, T\}, \quad (20)$$

$$x_{ijt}, y_{ijt}, w_{jt} \in \{0, 1\}. \quad (21)$$

In this formulation, Eq. (13) is the objective function, which is minimizing the reduced cost. Note that the reduced cost is equal to  $\sum_{j,t} c_{jt} (1 - w_{jt}^s d_j) q_{jt}^{-v}$  which, after removing the fixed terms, is equivalent to maximizing  $\sum_{j \in J} \sum_{t \in T} \tau c_{jt} w_{jt} q_{jt} d_j$ . Equation (14) ensures that each team at each time can be assigned to exactly one job. This job can be patrolling a station or taking a break. Equation (15) ensures for each team that the pairs of jobs undertaken consecutively are feasible, for example, the team cannot consecutively patrol two stations that are far apart from each other, or they cannot take two consecutive breaks. Equation (16) ensures that if any team is patrolling station  $j$  at time  $t$  then  $w_{jt}$  is equal to 1. Equation (17) ensures that if no team is patrolling station  $j$  at time  $t$  then  $w_{jt}$  is equal to 0. Constraint (18) ensures that the number of breaks for each team is exactly equal to 2. Constraint (19) ensures that consecutive breaks do not happen. Constraint (20) ensures that the breaks are not scheduled at the beginning or the end of time horizon. Constraint (21) is the integrality constraint for variables  $w_{jt}$ ,  $y_{ijt}$  and  $x_{ijt}$ .

## Overall column generation procedure and a dual bound

Figure 2 presents the pseudo-code for the overall column generation algorithm. As seen in this figure, the column generation algorithm starts with a randomly generated set of initial columns (schedules). The RLPM is solved using this set of initial columns and the vector of dual values is obtained. Dual values are then used in the pricing sub-problem to generate a new column (schedule). If a new column with a negative reduced cost is obtained, it is added to the RLPM and the process is repeated; otherwise, the procedure terminates.

During column generation, we have access to a dual bound on value of the game so that we can terminate the algorithm when a desired solution quality is reached. The following lemma offers a lower bound on the value of the game, which can be computed in each iteration of the column generation algorithm.

**Lemma 1** (Dual bound) Let  $v(RLPM)$  and  $v(LPM)$  denote the optimum objective function value of the current *RLPM* and *LPM*, respectively. Also, let  $v(PP)$  be the minimum reduced cost obtained by solving the pricing sub-problem to optimality. We have:  $v(LPM) \geq v(RLPM) + v(PP)$ .



---

```

1 Initialize set of schedules  $\mathcal{S}'$ .
2 Solve RLPM. Let  $\bar{q} = (q_{jt})$  and  $v$  be the obtained optimal dual values
  and objective function value, respectively.
3 Solve the pricing sub-problem using  $\bar{q}$  as dual values and let
   $\mathbf{w}^* = (w_{jt}^*)$  be the obtained optimal solution.
4 if  $\sum_{jt} c_{jt}(1 - w_{jt}^*)q_{jt} - v < 0$  then
5 |   Add the new schedule  $\mathbf{w}^*$  to  $\mathcal{S}'$ .
6 |   Go To Line 2.
7 else
8 |   Return  $v$  as the value of the game.
9 |   Terminate the procedure.
10 end

```

---

**Fig. 2** Pseudo-code for the overall column generation algorithm

**Proof** General form of this result can be found in Lübbecke (2011). Because we know that  $\sum_{s \in \mathcal{S}'} x_s = 1$  for an optimal solution of the MP, one cannot improve  $v(RLPM)$  by more than 1 times the smallest reduced cost  $v(PP)$ , hence  $v(LPM) \leq v(RLPM) + v(PP)$ . ■

**Remark 1** Note that in order for the bound in lemma 1 to be valid; the pricing sub-problem should be solved to optimality. In general, the dual bound is not monotone over the iterations, this is called the yo-yo effect.

### A heuristic solution approach for the pricing sub-problem

In this section, we develop a dynamic programming based greedy algorithm to obtain an approximate solution to the pricing sub-problem. This heuristic algorithm can be used inside the column generation procedure to obtain an approximate solution for the patrolling game. To solve the pricing sub-problem, we use a greedy algorithm to generate schedules. We define a patrol as a detailed course of action for one team that determines what to do at each time period  $t$ . Note that a patrol is different from a schedule. While a schedule determines the complete course of action for all teams, a patrol does so for only one team. The definitions match if we have only one security team. For each security team the greedy algorithm assigns a patrol that maximizes  $\sum_{j \in J} \sum_{t \in T} c_{jt} q_{jt} d_j a_{jt}^p (1 - w_{jt})$ , where  $w_{jt}$  is a binary variable equal to 1 if station  $j$  at time  $t$  is already covered. To find such a patrol, we use a dynamic programming algorithm. The details of the dynamic programming approach are presented in Subsection “Dynamic programming procedure”. In Subsection “Overall heuristic procedure”, the overall greedy algorithm is presented and some results on the quality of the proposed algorithm are presented.

#### Dynamic programming procedure

In this section, we develop a dynamic programming (DP) procedure to solve the pricing sub-problem to find an optimal patrol. The aim is to find a patrol  $p$  with  $a_{jt}^p$  such that  $\sum_{j \in J} \sum_{t \in T} c_{jt} q_{jt} d_j a_{jt}^p (1 - w_{jt})$  is maximized.

DP is a method for solving complex problems by breaking them down into smaller problems (Larson and Casti 1978). In order to solve a problem using DP, the problem must be divided into smaller problems called stages. The stages are often solved backward, which is the case in the proposed DP procedure. Each stage has a number of states that are generally the information needed to solve the stage. The decision at a stage updates the state for the current stage to the state for the next stage. Given the current state, the optimal decision for the remaining stages is independent of the decisions made in the previous stages. This is the fundamental principle of optimality in DP. It means that the problem can be broken down into smaller problems, which can be solved independently. Finally, a recursive relationship between the values of decision at the current stage and the optimum decisions at previous stages must be identified. In other words, the optimum decision uses the previously found optimum decision values. Elements of the proposed dynamic programming procedure are as follows:

- $t = 1, 2, \dots, T$ : Stage variable, each time period is considered a stage.
- $x(t) = [b(t), l(t)]$ : State variable at stage  $t$ , consists of two components: number of breaks until time period  $t$ :  $b(t)$  and location at time  $t$ :  $l(t)$ .
- $u(x, t)$ : Decision in state  $x$  and stage  $t$ , take a break at an adjacent station  $j$  or patrol an adjacent station  $j$ .
- $r(x, t, u)$ : Instant reward in state  $x$  at stage  $t$  if action  $u$  is taken. If the action is to patrol an adjacent station  $j$ , then the obtained instant reward is  $r(x, t, u) = c_j q_j d_j (1 - w_{ji})$ . If the action is to take a break at an adjacent station  $j$ , then the reward is  $r(x, t, u) = 0$ .
- $R(x, t)$ : Optimum accumulated reward with state  $x$  at stage  $t$ .
- $u^*(x, t)$ : Optimum action with state  $x$  at stage  $t$ .
- $F(x, u)$ : Transition function, if action  $u$  is taken in state  $x$ , the state in the next stage will be  $F(x, u)$ . If the action is to patrol an adjacent station  $j$ , then the next state is  $x(t+1) = [b(t+1), l(t+1)] = [b(t), j]$ . If the action is to take a break at an adjacent station  $j$ , then the next state is  $x(t+1) = [b(t+1), l(t+1)] = [b(t+1), j]$ .

Now using these parameters, a recursive equation can be written for the optimum accumulated reward functions:

$$R(x, t) = \max_{u(x)} \{r(x, t, u) + R(F(x, u), t + 1)\}. \quad (22)$$

This equation describes an iterative relation for determining  $R(x, t)$ , for all feasible  $x$  and  $t$  from the knowledge of  $R(x, t + 1)$  for all feasible  $x$  and  $t + 1$ .  $R(x, T)$  can be easily solved by using  $R(x, T) = \max_{u(x)} \{r(x, T, u)\}$  then  $R(x, T - 1)$  can be determined using Eq. (22). Continuing this backward procedure  $R(x, 1)$  is determined.

### Overall heuristic procedure

Figure 3 presents the pseudo code for the overall greedy approach. As seen in this pseudo code, the algorithm starts with initializing  $w_{ji}$  to indicate that no patrols have

```

1 for j ← 1 to |J| do
2   for t ← 1 to |T| do
3     | wjt ← 0
4   end
5 end
6 for i ← 1 to |I| do
7   Obtain a new patrol using DP and let a* = (a*jt) be the obtained optimal patrol.
8   Assign patrol a* to team i.
9   foreach (j, t) with a*jt = 1 do
10    | wjt ← 1
11  end
12 end
    
```

**Fig. 3** Pseudo-code for the greedy algorithm

been assigned to any security team. Then the DP is used to obtain a patrol with maximum  $\sum_{j \in J} \sum_{t \in T} c_{jt} q_{jt} d_j a_{jt}^p (1 - w_{jt})$ . Next, we assign the obtained patrol to the security team and update  $w_{jt}$  so that it reflects the covered  $(j, t)$  pairs. This process is repeated for all available security teams.

The following lemma shows that the greedy algorithm achieves an approximation ratio of  $1 - \left(1 - \frac{1}{|I|}\right)^{|I|}$ .

**Lemma 2** For fixed values of  $q_{jt}$  Suppose  $w_{jt}^*$  and  $w_{jt}^G$  are the optimal and greedy values of  $w_{jt}$ . Then we have:  $\sum_{j \in J} \sum_{t \in T} c_{jt} q_{jt} d_j w_{jt}^G \geq \left(1 - \left(1 - \frac{1}{|I|}\right)^{|I|}\right) \sum_{j \in J} \sum_{t \in T} c_{jt} q_{jt} d_j w_{jt}^*$

**Proof** The pricing sub-problem is in fact a weighted maximum coverage problem with  $c_{jt} q_{jt} d_j$  acting as weights and  $|I|$  as the maximum number of sets to be selected. The result comes from the fact that for the weighted maximum coverage problem, the greedy algorithm achieves an approximation ratio of  $1 - \left(1 - \frac{1}{|I|}\right)^{|I|}$  (Nemhauser et al. 1978). ■

The next lemma shows the impact of using the greedy algorithm in the column generation procedure instead of solving the sub-problem to optimality.

**Lemma 3** Let  $v(RLPM)$  and  $v(LPM)$  denote the optimum objective function value of the current *RLPM* and *LPM* respectively. Also, let  $v(PP^G)$  be the reduced cost obtained by solving the pricing sub-problem using the greedy algorithm. We have:

$$v(LPM) \geq v(RLPM) + \frac{v(PP^G)}{1 - \left(1 - \frac{1}{|I|}\right)^{|I|}} > v(RLPM) + \frac{v(PP^G)}{1 - \frac{1}{e}} \tag{23}$$

**Proof** Let  $v(PP)$  be the optimal solution of pricing sub-problem. From Lemma 2 we have  $V(PP^G) \geq \left(1 - \left(1 - \frac{1}{|I|}\right)^{|I|}\right) v(PP) < \left(1 - \frac{1}{|I|}\right) v(PP)$ . Thus:

$$v(PP) \geq \frac{V(PP^G)}{1 - \left(1 - \frac{1}{|I|}\right)^{|I|}} > \frac{V(PP^G)}{1 - \frac{1}{e}} \tag{24}$$

Moreover, from Lemma 1 we have:

$$v(LPM) \geq v(RLPM) + v(PP). \quad (25)$$

The result follows from inequalities (24) and (25). ■

**Remark 2** Note that when  $|I| = 1$ , when there is only one security team, the bound in Lemma 2 is tight and the greedy algorithm obtains the optimal solution.

**Remark 3** Even-though we cannot prove a tighter bound for the greedy algorithm, the numerical experiments show that the solutions obtained using the greedy algorithm match the optimal solution in every instance.

## Numerical experiments

In this section, we perform computational experiments to investigate efficiency of the proposed algorithms and gain insight on some properties of the game. The algorithms are coded in C++ and CPLEX 12.6 solver has been used to solve the LPs and the pricing sub-problems. The computational experiments are performed on a computer with 2.4 GH processor and 4 GB of RAM. For all experiments in this section, we assume that the number of security teams is given as a parameter, represented by  $|I|$ . We also assume that these security teams are identical and they cannot be divided into smaller teams. Meaning that each team must be in exactly one station at each time. Moreover, each security team is required to have exactly two breaks in the time horizon.

In our first experiment, we compare the performances of the proposed exact column generation approach (ECG) with the greedy column generation (GCG). Our base set of test instances consists of randomly generated instances with the underlying general graphs. To generate general graphs, the expected edge density (measured as  $\frac{|E|}{|J|(|J|-1)}$ , where  $E$  is the set of edges in the graph. We do not consider self-loop edges in calculating edge density) of 60% is used for the graph, and the number of stations,  $|J|$ , ranges from 20 to 40. In generating the general graphs, we first started with a random tree and added random edges until the edge density reaches 60%. We generated five instances for each problem size, with different values of  $T \in \{10, 11, \dots, 15\}$  and  $|I| \in \{1, 2, \dots, 5\}$ .

Our results show that the GCG always finds the same expected damage value as the ECG. One possible explanation for this phenomenon is that our model only indirectly considers the lost time to travel from one station to another. We then compare the algorithms in terms of their run times. Tables 1 and 2 show the obtained run times in seconds. In these tables for each instance, the smaller run time is highlighted in bold. As seen in these tables GCG performs better than ECG for all instances of the problem. Moreover, for both algorithms, the run time generally increases as the number of stations, i.e.  $|J|$ , and the number of time periods, i.e.  $|T|$ , increases.

Figure 4 shows the effect of increasing the number of security teams on the expected damage for different number of stations. As seen in this figure, the expected damage decreases as the number of security teams increases. However, the amount of decrease

**Table 1** Comparison of exact column generation and greedy column generation run times (seconds)

$ J $	$T=10$		$T=11$		$T=12$	
	ECG	GCG	ECG	GCG	ECG	GCG
20	195.01	<b>64.18</b>	228.04	<b>62.37</b>	323.93	<b>119.62</b>
25	354.48	<b>115.77</b>	471.22	<b>153.19</b>	519.32	<b>208.09</b>
30	477.88	<b>146.73</b>	554.75	<b>229.61</b>	776.09	<b>236.04</b>
35	891.87	<b>276.07</b>	1015.9	<b>422.39</b>	1287.66	<b>503.86</b>
40	1147.89	<b>336.41</b>	1855.11	<b>719.51</b>	2830.80	<b>663.60</b>

in expected damage also decreases as the number of security teams increases. This diminishing returns effect is visible for all values of  $|J|$ .

Figure 5 shows convergence of the lower and upper bounds of GCG over iterations for an instance of the problem with  $|J|=1$ ,  $T=10$  and different values of  $|J|$ . In each iteration the lower bound is computed using Lemma 3 and the upper bound is taken as the current objective function value. As seen in this figure, the lower bound is not monotone over the iterations and the yo-yo effect is visible. Moreover, for most cases, the upper bound value stabilizes way before the algorithm terminates. This means that after the upper bound values stabilize, we can terminate the column generation algorithm without undermining the solution quality drastically.

Next, we analyze the effect of detection probabilities on the expected damage, here we assume that the detection probabilities are equal to each other. Figure 6 shows the effect of detection probability on the expected damage for different values of number of security teams  $|J|$ . As seen in this figure, the expected damage is smaller when there are more security teams. Moreover, as the detection probability increases, the expected damage, generally, decreases and this decrease, roughly speaking, behaves linearly for higher values of detection probability.

Next, we consider a real case of an urban rail network with 51 stations. Figure 7 shows the network graph of this case. As seen in this figure, there are two main lines that connect different parts of the city together. There is one free interchange between stations 41 and 16. Occupancy levels in each station are collected based on ridership totals, which are in turn based on turnstile entry and exclude free interchange ridership. A 12-h planning time horizon, starting from 5:00 AM and ending at 5:00 PM, is considered for this problem (i.e.  $T=12$  is considered). Based on this information, we run the GCG algorithm to obtain the best patrolling strategy for  $|J| \in \{1, 2, \dots, 10\}$ .

**Table 2** Comparison of exact column generation and greedy column generation run times (seconds)

$ J $	$T=13$		$T=14$		$T=15$	
	ECG	GCG	ECG	GCG	ECG	GCG
20	382.34	<b>146.01</b>	385.26	<b>199.01</b>	472.00	<b>201.62</b>
25	707.37	<b>383.70</b>	725.17	<b>272.52</b>	987.22	<b>430.63</b>
30	1272.09	<b>619.50</b>	1595.14	<b>829.14</b>	1945.62	<b>1187.35</b>
35	1384.21	<b>578.76</b>	2044.15	<b>932.16</b>	2554.24	<b>1352.95</b>
40	2746.47	<b>1297.65</b>	2897.79	<b>1432.55</b>	3736.70	<b>2273.71</b>

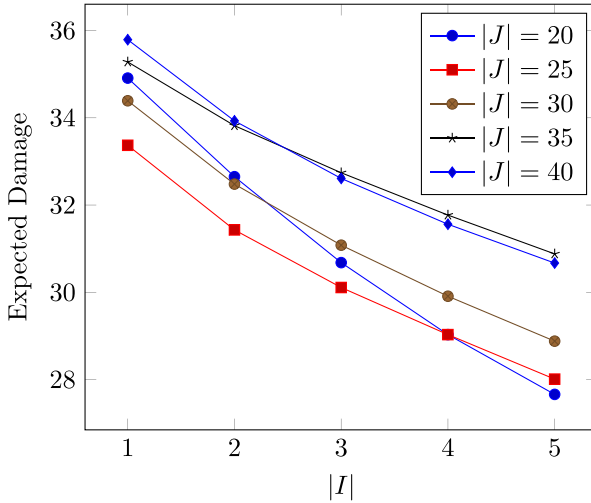


Fig. 4 The effect of number of security teams on expected damage

We first study the effect of the defender’s deviation from the Nash equilibrium on the expected damage. Specifically, we consider the case that the defender, in deriving her strategy, mistakenly thinks that the attack probabilities are the same and are uniformly distributed over stations. She then uses a probabilistic approach (PA) to obtain a single schedule. We compare the expected damage in this case with the expected damage in the Nash Equilibrium (NE). Figure 8 shows the results of this comparison. As seen in this figure, Nash strategy results in smaller expected damage values for all instances. Moreover, as the number of teams increases, the expected damage decreases for both NE and PA. The diminishing returns phenomenon is visible for NE, however this effect does not exist for PA.

We now study the distribution of the expected number of visits in important stations. Figure 9 shows the expected number of visits over a 30-day period for the 5 most

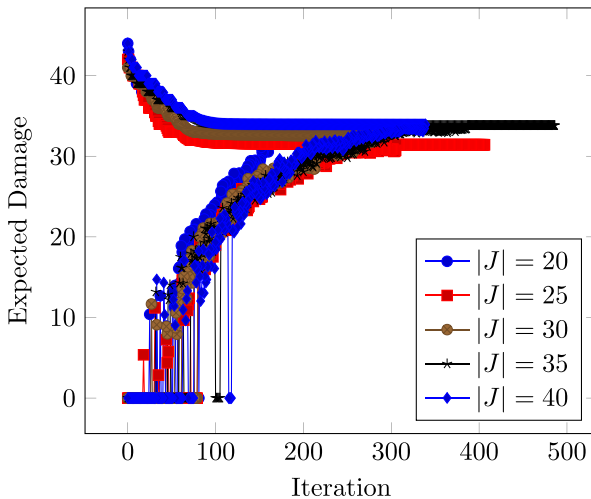


Fig. 5 Convergence of GCG over iterations

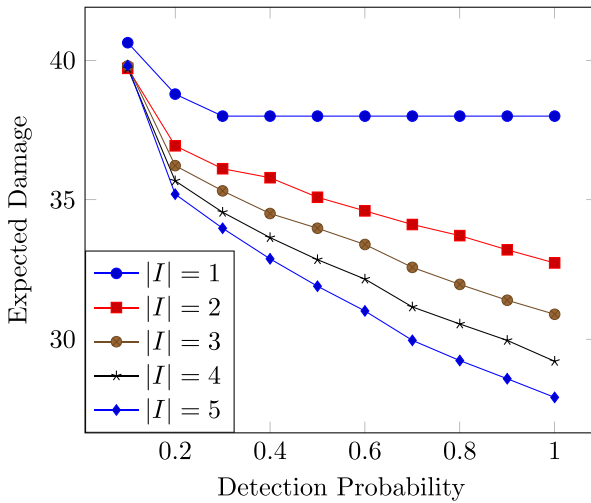


Fig. 6 The effect of detection probability on expected damage

visited stations based on time of the day, for the case of 10 security teams. As seen in this figure, majority of the stations have two peak visit times: one starts around 7:00 AM and end around 9:00 AM, the other one starts around 2:00 PM and end around 4:00 PM. Some stations also have peak visit times at the start and end of times horizon.

Next, we study the distribution of the expected damage among stations based on time of the day. Figure 10 shows the distribution of expected damage for 10 stations

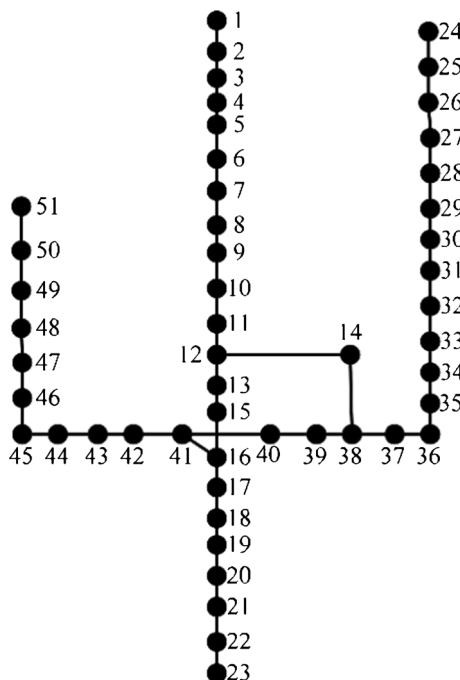


Fig. 7 Case network

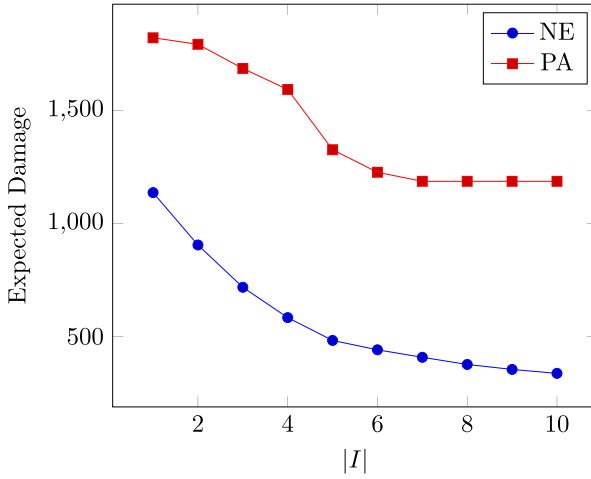


Fig. 8 Comparison of probabilistic approach (PA) and the Nash equilibrium (NE)

with the highest expected damage values, for the case of 10 security teams. As seen in this figure, for the majority of the stations there are two peak times for the expected damage: one starts around 7:00 AM and end around 10:00 AM, the other one starts around 1:00 PM and end around 3:00 PM.

### Conclusions and future research

In this paper, we propose a game theoretic approach for patrolling urban rail networks. The proposed model generates inherently randomized schedules that observe rostering related constraints. Our proposed model is a non-cooperative simultaneous move game

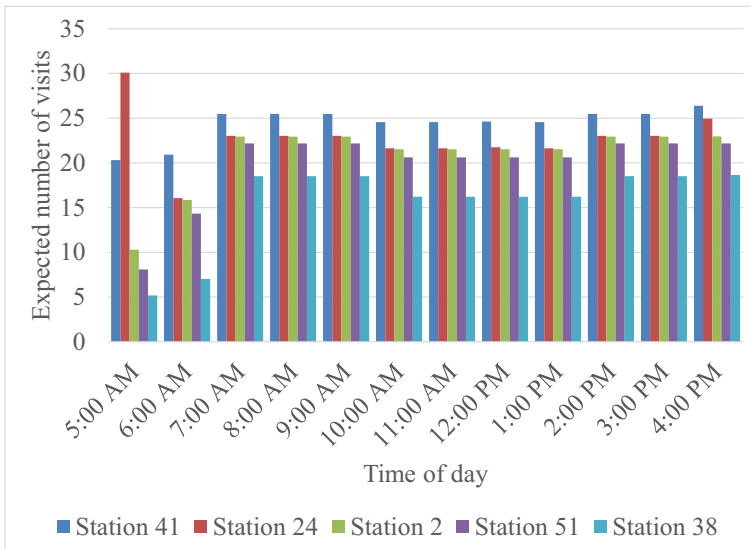
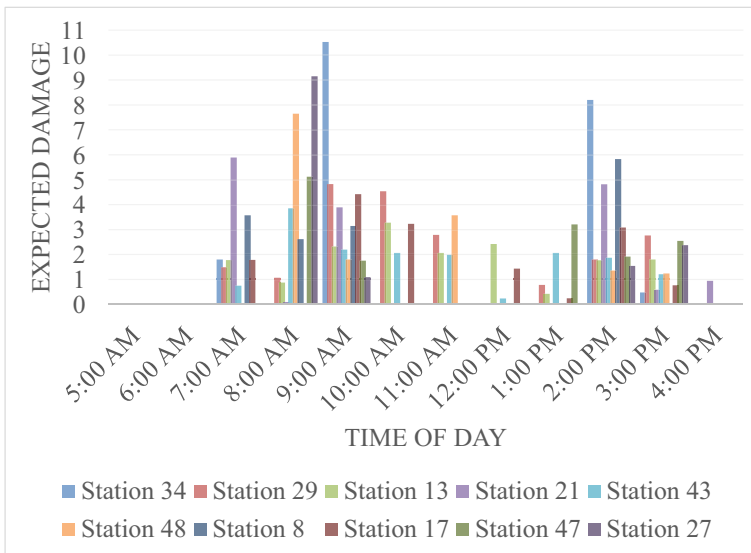


Fig. 9 Expected number of visits for five most visited stations





**Fig. 10** Distribution of expected damage for 10 stations with highest expected damage values

between a defender and an attacker. We developed efficient algorithms to find the Nash equilibrium for this game. We also presented lower bounds on the expected potential damage, which can be used to terminate the column generation algorithm when a desired solution quality is reached. We then run computational experiments to investigate the efficiency of the proposed algorithms and to gain insight about the value of the patrolling game. Our results show the efficiency of the proposed algorithms. Finally, we present results for the case of a real urban rail network.

Our study closes a gap in the literature by extending the existing models so that schedule randomization is done in a strategic way with consideration of possible adversary strategies. However, there is still a need to further study more efficient solution methods to find the Nash equilibrium. Moreover, studying the effect of transparency vs secrecy is an interesting topic for future research in this area. Current model assumes complete secrecy of generated schedule. However, the adversary may perform surveillance to find out the current locations of the patrollers. Transparency could be modelled via a Stackelberg game.

Another way to extend the current research is integrating the current model with long term strategic decisions to obtain a comprehensive model. The current model only studies the operational decision of scheduling the security teams; it assumes that the strategic decisions are fixed and given. For example, the number of available security teams is an input parameter, moreover the detection probabilities are assumed to be fixed and given. However, the number of security teams can be considered as a decision variable, the detection probabilities can be changed by investing on new technologies. Usually the operational and strategic decisions are studied separately, however these decisions affect each other and integrating them to obtain a comprehensive model may lead to significant reductions in the expected potential damage.

**Acknowledgments** This material is based upon work supported by the National Science Foundation under Grant No.1436288

## References

- Chaiken JM, Dormont P (1978) A patrol car allocation model: capabilities and algorithms. *Manag Sci* 24: 1291–1300
- Chelst K (1978) An algorithm for deploying a crime directed (tactical) patrol force. *Manag Sci* 24:1314–1327
- Delle Fave FM, Brown M, Zhang C et al (2014) Security games in the field: deployments on a transit system. In: *International workshop on engineering multi-agent systems*, pp 103–126
- Delle Fave FM, Shieh E, Jain M et al (2015) Efficient solutions for joint activity based security games: fast algorithms, results and a field experiment on a transit system. *Auton Agent Multi-Agent Syst* 29:787–820. <https://doi.org/10.1007/s10458-014-9270-4>
- Jiang AX, Yin Z, Johnson MP et al (2012) Towards optimal patrol strategies for fare inspection in transit systems. In: *AAAI spring symposium: game theory for security, sustainability, and health*
- Jiang AX, Yin Z, Zhang C et al (2013) Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In: *Proceedings of the 2013 international conference on autonomous agents and multi-agent systems*, pp 207–214
- Larson RC (1972) *Urban police patrol analysis*. MIT Press, Cambridge
- Larson RE, Casti JL (1978) *Principles of dynamic programming, part 1: basic analytic and computational methods*. Marcel Dekker, New York
- Lau HC, Yuan Z, Gunawan A (2016) Patrol scheduling in urban rail network. *Ann Oper Res* 239:317–342
- Lübbecke ME (2011) Column generation. *Wiley Encyclopedia of Operations Research and Management Science*. <https://doi.org/10.1002/9780470400531.eorms0158>
- Moteff J (2005) *Risk management and critical infrastructure protection: assessing, integrating, and managing threats, vulnerabilities and consequences*. Library of Congress Washington DC Congressional Research Service, Washington DC
- Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing submodular set functions  $\{1, \dots, n\}$ . *Math Program* 14:265–294
- Olson DG, Wright GP (1975) Models for allocating police preventive patrol effort. *J Oper Res Soc* 26:703–715
- Paruchuri P, Tambe M, Kraus S, et al (2005) Safety in multiagent systems by policy randomization. In: *Second international workshop on safety and security in multiagent systems*, pp 89–101
- Paruchuri P, Tambe M, Ordóñez F, Kraus S (2006) Security in multiagent systems by policy randomization. In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp 273–280
- Paruchuri P, Pearce JP, Tambe M, et al (2007) An efficient heuristic approach for security against multiple adversaries. In: *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, p 181
- Paruchuri P, Pearce JP, Marecki J, et al (2008) Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg games. In: *Proceedings of the 7th international joint conference on autonomous agents and multiagent systems-vol. 2*, pp 895–902
- Pita J, Jain M, Marecki J, et al (2008) Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles international airport. In: *Proceedings of the 7th international joint conference on autonomous agents and multiagent systems: industrial track*, pp 125–132
- Shvetsov A, Shvetsova SV (2017) Protection of high-speed trains against bomb-carrying unmanned aerial vehicles. *J Transp Secur*. <https://doi.org/10.1007/s12198-017-0182-9>
- Shvetsov A, Shvetsova S, Kozyrev VA et al (2017) The “car-bomb” as a terrorist tool at metro stations, railway terminals and airports. *J Transp Secur* 10:31–43. <https://doi.org/10.1007/s12198-016-0177-y>
- Strandberg V (2013) Rail bound traffic—a prime target for contemporary terrorist attacks? *J Transp Secur* 6: 271–286
- Tsai J, Kiekintveld C, Ordóñez F, et al (2009) IRIS—a tool for strategic security allocation in transportation networks. In: *8th international joint conference on autonomous agents and multiagent systems (Industry Track)*, May 2009
- Varakantham P, Lau HC, Yuan Z (2013) Scalable randomized patrolling for securing rapid transit networks. In: *Proceedings of the twenty-fifth innovative applications of artificial intelligence*
- Yin Z, Jiang AX, Johnson MP, et al (2012) TRUSTS: scheduling randomized patrols for fare inspection in transit systems. *IAAI*, pp 2348–2355. <https://doi.org/10.1609/aimag.v33i4.2432>