CASPT 2015

Security crew scheduling at Netherlands Railways

Hilbert Snijders · Ricardo L. Saldanha

Abstract We address the problem of scheduling work of security guards operating on trains and stations, and explain how OR techniques like column generation and Lagrangian relaxation are suitable for solving a problem that adds new challenges to classical crew scheduling. Planning the work of these teams is challenging due to several reasons. First, the optimization goal in itself is defined in more qualitative than quantitative terms. More precisely, it concerns achieving the highest risk mitigation through a proper choice of work assignments. Second, the choice of work assignments is performed among many possible alternatives. For instance, each security team has millions of possible alternative work assignments, most of which interfere with the work assignments of other teams. We tested the model with a real problem instance supplied by Netherlands Railways. Experimental results show that the prototype not only presents meaningful results in terms of scheduling work, but also can help decide the way security guards will be deployed in the future.

Keywords Crew Scheduling · Operations Research · Railway Planning · Security Patrolling

1 Introduction

Netherlands Railways (NS) is the main Dutch railway operator of passenger trains. It operates about 5.000 trains on a working day, transporting on average more than 1.1 million passengers. The company manages about 380 stations. Each year, approximately 10.000 acts of aggression against passengers and/or NS personnel are reported. Also, more than 380.000 fare evaders are caught

H.H.C.M. Snijders
Netherlands Railways, Laan van Puntenburg 100, Postbus 2025, 3500 HA, Utrecht
E-mail: hilbert.snijders@ns.nl

R.L. Saldanha
SISCOG - Sistemas Cognitivos, SA
E-mail: rsaldanha@siscog.pt

and reported. NS employs around 700 so-called Veiligheid & Service (V&S or Security & Service in English) employees. The primary aim of these employees is to improve the (sense of) social safety of both passengers and NS personnel. The V&S employees always operate in teams of at least two persons in order to increase their own safety and effectiveness during interventions. The teams are deployed for purposes other than improving social safety as well, including checking tickets, helping disabled passengers enter or leave a train, giving travel advice, providing first aid and assisting passengers safely leave a train when it is broken down. Both preventive and reactive measures are part of the V&S teams' workload. Preventive measures include accompanying passenger guards on train trips where a high occurrence of passenger aggression is expected, controlling crowds where they are expected, and patrolling at stations. Some V&S teams can also be called upon to interrupt their current activities to help out in case of emergencies, thus improving safety in a reactive manner. A high correlation between asking fare evaders for their ticket and acts of aggression performed by these fare evaders has been observed. This is the main reason why (assisting with) checking tickets is the most frequently scheduled activity of V&S teams at the time of writing.

This paper deals with scheduling the work of V&S teams, and is a joint effort of NS researchers and software development partner SISCOG (see SISCOG (2015)). The problem is concerned with assigning work (jobs) to a fixed set of duties. Currently, jobs are distributed among duties locally by team managers or roster schedulers. They use generalized (custom-made) recommendations that are based on fare evasion and aggression reports, as well as information about events that require additional security measures, to help decide what jobs to assign to the available duties. The level of detail of these jobs is typically very low (e.g. mentioning that tickets should be checked on trains in a specific time interval without specifying which trains should be checked). NS envisions a planning process in which V&S duties are scheduled centrally and in greater detail than in the current planning process, hypothesizing that a central, detailed approach will ultimately help improve social safety on the Dutch railway network. The aim of the operations research (OR) model described in this paper is to support (and shape) this future planning process. This model was implemented in a software prototype named TUTIS as an extension of CREWS (Morgado and Martins (1998)).

The remainder of this paper is organized as follows. Section 2 provides a short literature review of security patrol scheduling problems, while Section 3 is concerned with the problem description. In Section 4, the solution method we chose to implement is described. Section 5 contains the results of the computational experiments that were performed using this solution method. The paper is concluded in Section 6.

2 Literature review

In recent years, the most prevalent methods of modelling scheduling of security patrols involve game theory. In this context, security-related problems are often defined as games where attackers attempt to assault one or more targets while defenders attempt to defend these targets, assuming there are more targets than attackers and/or defenders. The strategy of attackers and defenders consists of choosing which targets are to be attacked or defended, respectively. If the assumptions are made that attackers choose their strategies rationally and that attackers can observe the defenders' strategy before having to choose their own, defenders can optimize their strategy (since they can correctly anticipate the way attackers will react to their chosen strategy). Basically, this is the description of an attacker-defender Stackelberg game (see Ordóñez et al (2013)). Randomization can be added to make strategies, and the patrol schedules that are a result of these strategies, less predictable. Stackelberg games can be written as (and solved like) mixed integer linear programming problems. The Teamcore Research Group from the University of Southern California (see Teamcore (2015)) is, to the best of our knowledge, the foremost group of specialists that use Stackelberg games to model security patrolling problems. The group has successfully supported resource allocation problems in the security domain for the Los Angeles Airport (see Pita et al (2008)), the Federal Air Marshals Service (see Tsai et al (2009)), nationwide airports (see Pita et al (2011)), the United States Coast Guard (see Shieh et al (2012)) and the Los Angeles County Sherrifs Department (see Yin et al (2012)).

Other known methods of dealing with security patrol scheduling problems include linking a game theoretical model that estimates the vulnerability of Singapore's subway stations and a linear programming model that schedules security teams that traverse Singapore (see Lau and Gunawan (2012)), and a mixed integer linear program that assists in scheduling Danish conductors by optimizing the income from penalty fares (see Thorlacius et al (2010)). None of the aforementioned models covers the specific requirements of the NS crew scheduling problem for V&S teams. Also, in general, it can be said that the number of research publications in which security patrol scheduling problems in large networks are handled using operations research-related techniques is far from extensive.

Martin van Meerkerk, a former intern at NS, wrote a master thesis about TUTIS (Meerkerk (2014)), which focuses on job generation (see Section 3.1).

3 Problem description

We address the problem of scheduling the daily work of V&S employees in an attempt to reduce risks related with the occurrence of acts of aggression, fare evasion, and lack of support and assistance to train passengers. In the remainder of this document we use the term *Problem* to address this scheduling problem.

V&S employees are organized in teams of two people that perform the same activities. Therefore, we can describe the Problem as the problem of assigning work to teams and not to individuals.

During a working day a team mainly performs two kinds of activities (we call these *jobs* henceforth): platform control and train control.

A *platform control job* consists of blocking off a specific platform of a station for ticket inspection. Platforms with n exits often require n teams. The added value of having $p < n$ teams on such platforms is almost null because aggressors or fare evaders can choose an unsupervised exit. Jobs performed on such platforms are known as *all-or-nothing* jobs because they should be assigned to all required teams or to no teams. Platform control jobs can be decomposed into smaller jobs, which allows for switching of teams.

A *train control job* consists of surveillance and inspection activities on a train. Since fare evaders and potential transgressors usually abandon a train after detecting the presence of V&S teams, and since V&S teams typically sweep a train once after they've entered it, the added value of having a team on a train more or less maxes out at some point. Covering the entire trip does not seem useful in most cases, so, among many possible segments of a train trip, either one is covered, or none at all. We call this the *one-or-nothing* behaviour.

While performing jobs, a team is mitigating *security risks* (i.e. possible aggression) and *fare evasion risks*. The higher these risks are in the location and at the time the job is performed the more added value there is in having a team performing that job. Therefore, we can say that a job has a security value and a fare evasion value, that somehow map the corresponding risks. If we want to make an efficient schedule, we should ensure that the limited number of teams available perform the most valuable jobs. The data that is available to evaluate risks comprises:

- reports of aggression and fare evasion written by train guards and V&S teams specifying the date, time and place of the occurrence (sometimes inside a train running from A to B); these data are important because of their predictive value with respect to the value of jobs;
- passenger forecasts for all train movements; this data is important to be able to compute the amount of time it takes for a V&S team to check everyone on a train trip;
- calendar of social events, like soccer games, concerts, etc; this data is important because jobs performed around the time and area of those events are usually characterized as being *high risk*, and tend to be very valuable.

Beside the value, a job also has a level of priority associated, which can be one of the following (by decreasing order of importance): mandatory, high risk or normal. Jobs with a certain level of priority should not be covered at the cost of leaving a job with a higher level of priority uncovered. While mandatory jobs must be covered, high risk and normal jobs are considered optional jobs.

Planners assign jobs to teams inside *duty templates* that are derived from the teams' rosters. Each duty template can be regarded as a time interval

inside which assigned work must fit. The assigned work is a sequence of jobs, constituting a duty. A duty template can be of type *backup* or *normal*. Backup duty templates correspond to teams that are on alert at a specific station, in the sense that they may be called at any time while they are on duty to perform activities that are more urgent than the activities they were originally scheduled for. All duties are subject to several constraints (some apply to each separate duty, while others affect sets of duties), for example:

- the assigned work must start (end) with a sign in (out) operation on the base to which the team belongs;
- the sequence of jobs in a duty must be continuous in space and time;
- a team cannot work more than a certain amount of hours without having a meal break;
- consecutive jobs in a duty must be separated by sufficient time to take the possibility of delays into account;
- teams can only perform jobs that are geographically inside the operational range of the base to which the teams belong;
- the average length of duties of teams belonging to the same base is upper bounded;
- the number of times a railway network segment should be covered by the whole set of duties is lower bounded;
- teams performing backup duties can only be assigned optional platform control jobs on platforms of the station specified in the duty template.

Figure 1 illustrates how train control and platform control jobs are combined in a duty. As shown, train control typically assumes the form of a sequence of small train trips (because of the maxing out of value mentioned above). Since a typical duty has many transfers and we want to cover as much value as possible, jobs should be combined in a "clever" way, otherwise too much idle time will be found on duties.

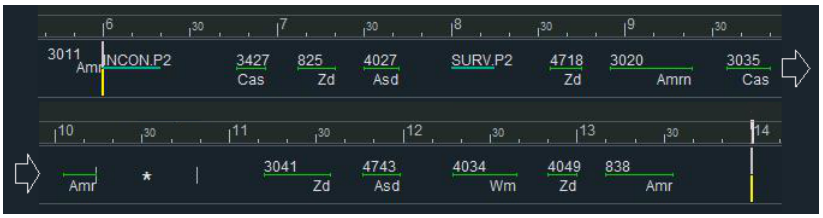

Fig. 1 A Gantt chart representation of a duty of a team based in Alkmaar; the arrows indicate that the duty is broken in two parts, where the second part (on the bottom) is the continuation of the first (on the top); the yellow vertical bars represent the time boundaries of the duty template; the dark green horizontal lines represent train control jobs, while the light green ones represent platform control; the names written below the jobs represent the ending stations of the jobs; between 10:15 and 10:45 the team takes a meal break in Alkmaar.

To conclude, we identify the most challenging aspects of the Problem:

– computing the security and fare evasion values of jobs in a sensible way;
– the all-or-nothing behaviour of multi-team platform control jobs;
– the one-or-nothing behaviour of train control jobs on the same train;
– the desire to optimize combinations of jobs in the form of duties, which are subject to many constraints;
– the need to handle large problem instances, where the number of jobs available to be assigned can reach 85000 in a single region.

4 Solution method

In order to solve the SCSP we adopt a two-step approach: first we generate jobs and then we assign them to teams. Dividing the approach in two steps is a way of handling the large computation times that are inherent to the complexity of the SCSP, as the first step (called Job generation) generates only the most promising jobs (a small subset of all possible jobs) as input for the second step (called Job assignment).

4.1 Job generation

Basically, the job generation problem is the part of the solution method where jobs are created and assigned two values (one for preventing aggression and one for preventing fare evasion). This is rather straightforward for some jobs, as they are specified by hand by NS management or other experts: these jobs consist of all platform control jobs and the train control jobs with a priority level of mandatory or high risk. Specifying platform control jobs typically requires knowledge about the precise locations that correspond to these jobs, while any jobs with a priority of mandatory or high risk have received this property because of some extraneous event or agreement (e.g. soccer matches that require crowd control, special holiday events that require V&S assistance, fire drills in cooperation with the fire brigade and so on). We assume that often the number (and/or duration) of jobs that is specified by NS managers will not be sufficient to make efficient use of the available duties. The train control jobs with a priority level of 'normal' can be used to aid in this matter.

Using the NS timetable, a large number of viable train control jobs can be found. Let L be the set of train control jobs to be covered. The set of train control jobs with a priority level of normal, L_n, is a subset of L. For each job $l \in L_n$, we assign values that take into account that time spent on a train loses usefulness over time. Let z be the number of passengers a V&S team can check (out) each minute, and p_l the number of passengers that corresponds to job l with duration d_l. After a certain duration threshold the train control job values max out (i.e. the threshold represents the moment where a V&S team could have checked (out) each passenger by making a single sweep through the train). This duration threshold h_l for each job l is defined as $h_l = \frac{p_l}{z}$. Using aggression and fare evasion reports as a basis, train control jobs receive

certain values for each minute of their duration until the duration threshold h_l is reached.

This means that for each job l value can be contributed for a duration of $u_l = min(d_l, h_l)$.

In a search for dependencies the only significant correlations that were found were those between reports (aggression/fare evasion) on the one hand and, independently, location (railway section) and train series (recurring trains in the timetable with a specified train route) on the other hand. For TUTIS, each job has two types of value: one that represents increasing security ($\mathsf{Security}_l$), constructed using the reports of aggression, and one that represents preventing fare evasion ($\mathsf{FareEvasion}_l$), constructed using the reports of issued fines. The (security or fare evasion) value that is added to a specific job for each minute of its duration is the sum of the number of reports that correspond to the location of that job and the number of reports that correspond to the train series of that job, multiplied by a scalar.

For instance, consider a train running from station Utrecht (Ut) at time 13:20, which stops at station Amsterdam Amstel (Asa) at time 13:48, and then arrives at station Amsterdam Centraal (Asd) at time 13:57. We know that last year there have been 278 cases where a fare evader was caught on the railway section that runs from Ut to Asd, and we know that 222 fare evaders were caught on the train series to which our example train belongs. Adding these numbers and multiplying the result by $\frac{1}{700}$ because of scaling purposes gives us a fare evasion value per minute of $\frac{(278+222)}{700} = \frac{5}{7}$. We also know that this train has a duration threshold (h_l) of 35 minutes. This means that job 1, where a team leaves the train at station Asa, gets fare evasion value for the full 28 minute trip ($\mathsf{FareEvasion}_1 = \frac{5}{7} \times min(28, 35) = 20$). Job 2, where a team leaves the train at station Asd, only receives value for 35 minutes of its duration, while the trip actually lasts 37 minutes ($\mathsf{FareEvasion}_2 = \frac{5}{7} \times min(37, 35) = 25$)). For a graphical illustration, see figure 2.

The job representing working on our train of interest can be found in figure 3, among other viable train control jobs. This figure was taken (and slightly adjusted) from Martin van Meerkerk's thesis on the scheduling of V&S teams at NS (Meerkerk (2014)).

In this figure we see two blue arrows departing from the leftmost node (station Ut, at time 13:20). One of these arrives at station Asa at time 13:48 and has a total value of 20, while the other arrives at station Asd at time 13:57 and has a value of 25. Both arrows correspond to working on the same train, where the team can either leave the train in Asa at 13:48 or in Asd at 13:57. While in this graph only fare evasion values are represented, aggression values can be calculated and represented in a similar way.

After all viable train control jobs are assigned a value for preventing fare evasion and a value for preventing aggression, at most one train control job is selected for each train (the so-called one-or-nothing behaviour). This is done taking into account minimum transfer times and the entire timetable, in such a way that sequences of train control jobs with high values remain for each

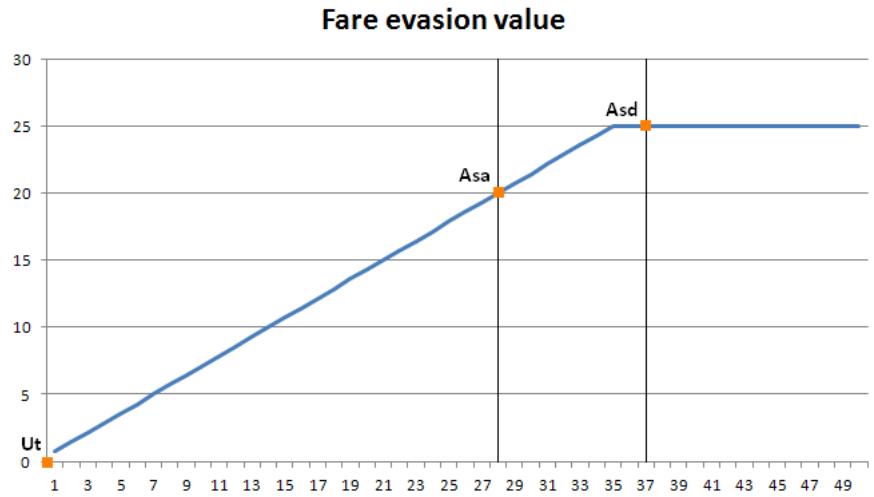

Fig. 2 An example of the way fare evasion value is assigned to jobs corresponding to a single train as a result of the duration of the job; the x-axis represents the duration of the job in minutes, the y-axis represents fare evasion value, the orange dots represent points where a V&S team could leave the train.

railway section. These are the only train control jobs with a priority level of 'normal' that are used in the job assignment problem. Selecting these jobs can be done using a standard shortest path heuristic. For any further information and deliberations on the merits of choosing specific shortest path heuristics for the job generation problem, we refer to Meerkerk (2014).

4.2 Job assignment

In rough terms, the job assignment problem (JAP) is the problem of combining jobs in the form of duties (see Figure 1) and of assigning them to a given set of teams, so that the maximum amount of value is assigned and all constraints are satisfied. With respect to classical crew scheduling problems (Abbink et al (2011)), it contains the following additional challenging features:

- most of the work will be left uncovered; this feature is already known but only in the context of rescheduling where only a little amount of work is left uncovered (Potthoff et al (2010));
- it involves not only covering work with duties but also assigning work to employees (inside their corresponding duty templates); this problem is already known but only in the context of rescheduling (Huisman (2007); Potthoff et al (2010));
- the all-or-nothing behaviour, which is new in the crew scheduling literature, to our knowledge.

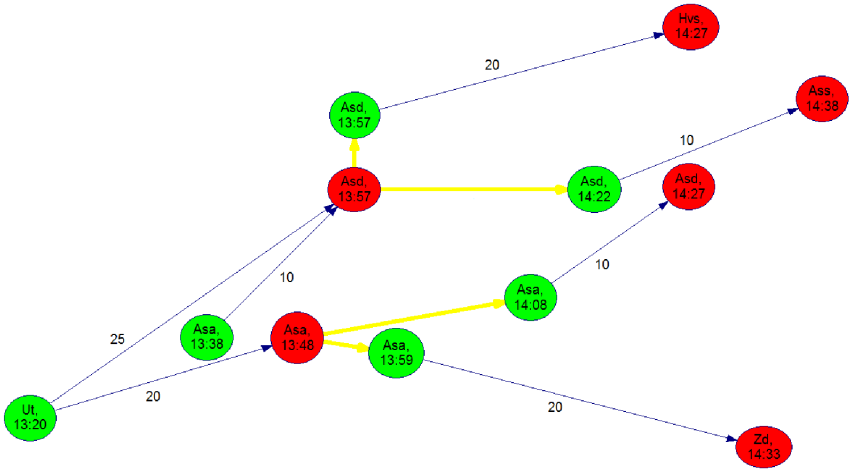

Fig. 3 A graphical representation of train control jobs (blue arrows) and waiting times at stations (yellow arrows), where the numbers near the arrows represent the fare evasion values of the jobs, the green nodes represent departures at stations, while the red nodes represent arrivals.

Before presenting the algorithm used to solve the JAP, we formulate it as a set covering problem with additional constraints.

4.2.1 Problem formulation

Since there are jobs (e.g. platform control operations) that have to be covered by two or more teams, we introduce the concept of *task* in order to formulate the JAP as a set covering problem with additional constraints. More precisely, we associate n tasks to each job that has to be performed by n teams, and say that each task has to be covered (at least) by one team.

Let M be the set of tasks to be covered. Among those, $\mathcal{M}_m$ denotes the set of mandatory tasks (related with mandatory jobs), and $\mathcal{M}_o$ the set of optional tasks (related with optional jobs). To every task $i \in \mathcal{M}_o$ we associate a positive number C_i representing the cost of leaving i uncovered. We denote T as the set of teams available and N as the set of duties representing different possibilities of covering tasks in M and assigning them to teams in T. Each duty $j \in N$ assigns tasks to team $k_j \in T$ and has a duration of d_j. For every team $k \in T$, we consider the set N_k containing the possible duties corresponding to work assignments to team k, i.e. $N_k = \{j \in N | k_j = k\}$. To every duty j we associate a positive number c_j representing the cost of using j in the solution. We define a binary parameter a_{ij} that indicates whether duty j covers task i. We denote B as the set of existing operational bases. For every base b we denote T_b as the teams belonging to b and N_b as the duties belonging to b, which is given by $N_b = \{j \in N | k_j \in T_b\}$. We denote S as the set of existing network segments, s_n as the number of times network segment $n \in S$ must be covered,

and w_{nj} as the number of times duty j covers network segment n. To each duty j we associate a binary variable x_j that takes the value 1 or 0 depending on whether j is part of the solution or not. To each task i we associate a binary slack variable ϑ_i that takes the value 1 or 0 depending on weather i is left uncovered in the solution or not. The problem can now be formulated as follows:

$$\min \sum_{j \in N} c_j x_j + \sum_{i \in \mathcal{M}_o} C_i \vartheta_i \tag{1}$$

$$\sum_{j \in N} a_{ij} x_j \geq 1 \quad \forall i \in \mathcal{M}_m, \tag{2}$$

$$\sum_{j \in N} a_{ij} x_j + \vartheta_i \geq 1 \quad \forall i \in \mathcal{M}_o, \tag{3}$$

$$\sum_{j \in N_k} x_j = 1 \quad \forall k \in T, \tag{4}$$

$$\sum_{j \in N} w_{nj} x_j \geq s_n \quad \forall n \in S, \tag{5}$$

$$\sum_{j \in N_b} (d_j - \overline{d}) x_j \leq 0 \quad \forall b \in B, \tag{6}$$

$$x_j \in \{0,1\} \; \forall j \in N, \tag{7}$$

$$\vartheta_i \in \{0,1\} \; \forall i \in \mathcal{M}_o. \tag{8}$$

The objective is to minimize the total cost of the duties in the solution plus the total cost of the tasks left uncovered in the solution. Constraints (2) ensure that every mandatory task is covered by a duty in the solution. Constraints (3) ensure that every optional task is either covered by a duty in the solution or left uncovered (with $\vartheta_i = 1$). Constraints (4) ensure that each team gets exactly one duty in the solution. Additional constraints (5) make sure that each network segment $n \in S$ is covered at least s_n times in the solution. Additional constraints (6) make sure that, for each operational base b, the average duty length of the duties belonging to that base in the solution is not larger than $\overline{d}$. Constraints (7) and (8) ensure that decision variables x_j and ϑ_i are binary.

Cost function

We suggest the following values for the coefficients of objective function (1):

$$c_j = \mathsf{FixedCharge} - \textstyle\sum_{i \in M_j} \mathsf{value}_i, \tag{9}$$

$$C_i = \mathsf{value}'_i. \tag{10}$$

The parcel $\mathsf{FixedCharge}$ is a positive fixed cost large enough to keep c_j positive. In each parcel $-\mathsf{value}_i$ (one for each task i belonging to the set of tasks M_j covered

by j) we consider the benefit of covering i, which is somehow proportional to the value associated with it. In the parcel value'_i we consider the inconvenience of leaving task i uncovered, which is somehow proportional to the value associated with it.

Jobs with all-or-nothing behaviour

Platform control jobs performed on platforms with n exits, normally require n teams, and therefore have a set of n tasks associated. This set should either be completely covered or not covered at all. Partially covered sets are not interesting because fare evaders can always choose an exit with no teams controlling tickets. In mathematical terms, we can define A as the set of jobs with all-or-nothing behaviour. Each job $l \in A$ has a set of all-or-nothing tasks $M_l \subset \mathcal{M}_o$ associated. Let i_l be one of the tasks in M_l, chosen arbitrarily. The all-or-nothing behaviour can be expressed in the following constraints to be added to (2)-(8):

$$\sum_{j \in N} a_{ij} x_j + \vartheta_{i_l} = 1, \forall i \in M_l, \forall l \in A. \tag{11}$$

These constraints state that either $\vartheta_{i_l} = 0$ and all all-or-nothing tasks M_l related with job l are covered or $\vartheta_{i_l} = 1$ and all those tasks are left uncovered.

4.2.2 Algorithm

In order to solve the problem, we decided to use a modified version of the heuristic described in Abbink et al (2011), which is based on Lagrangian relaxation and column generation. The reason why we adopted an heuristic instead of an exact approach is because real-world problem instances are too large to be solved exactly. In fact, our problem instances have more than 5000 jobs, which means that the number of variables x_j in the problem is in the order of many millions.

The heuristic solves a minimum cost set covering problem with additional constraints, which can be written as follows:

$$\begin{aligned} \text{Minimize} \quad & c^T x && (12)\\ \text{Subject to} \quad & \\ & Ax \geq 1, && (13)\\ & Bx \geq b, && (14)\\ & B'x \leq b', && (15)\\ & x \in \{0,1\}, && (16) \end{aligned}$$

where x is the decision variable vector, c, is the cost function vector, A is the covering matrix, B and B' are the additional constraint matrices, and b and b' are the constant vectors of the additional constraints.

We now describe the modifications performed over the heuristic described in Abbink et al (2011) in order to make it capable of handling the JAP.

Handling optional tasks

The challenge here was to make constraints (3) become regular covering constraints like (13). To achieve that, we need to make slack variables ϑ_i become decision variables like x_j. This is achieved by associating to each variable ϑ_i a slack duty j_i that covers only task i and assigns it a fictitious team that is not in T. The cost C_i becomes the cost of including it in the solution. Therefore, $\vartheta_i = 1$ means j_i is in the solution, and $\vartheta_i = 0$ means the opposite. We have to keep in mind that tasks covered by slack duties in the solution are not really covered. With this modification the total set of duties that can be used to cover tasks becomes $N \cup \{j_i | i \in \mathcal{M}_o\}$.

Handling teams

The first thing we did on this behalf was to relax the duty assignment constraints (4) into regular covering constraints like (13), namely:

$$\sum_{j \in N_k} x_j \geq 1 \; \forall k \in T. \tag{17}$$

By making this relaxation we allow having teams with two or more duties assigned in the solution. We avoid that by adding a fixed component in the cost of a duty (see below) and add a cleaning step at the end of the greedy heuristic that removes all the duties assigned to the same team, except the one with the smallest cost (c_j).

Removing duties in excess may result in uncovered mandatory tasks, partially covered all-or-nothing jobs and other infeasibilities. In order to solve those, we add a solution improvement step based on local search (see below).

Another important change in the algorithm related with teams is the fact that the pricing algorithm is run individually for each team (instead of each home base and each day of the week). By doing so, we are able to run the pricing algorithm over a connection network that enforces all the constraints related with a particular team (e.g. the ones that make sure the duty fits inside the duty template), and we are also able to associate the generated duties with the corresponding team. This also opens a door for running the pricing algorithm in parallel for several teams at the same time.

Cost function

In order to convert the cost function (1) into a cost function that fulfils all requirements, we need to decompose each parcel of coefficient c_j given in (9) and of coefficient C_j given in (10) into several parcels. We need to consider parcels that guide the heuristic towards the desired direction, and parcels that allow us to compare solutions that vary over certain dimensions.

The parcel $\mathsf{FixedCharge}$ is not decomposed, but set to a value that is large enough to help the heuristic minimise the number of duties in the solution.

The result of decomposing value_i into more parcels is:

$$\begin{aligned}\mathsf{value}_i = \quad & w_{\mathsf{s}} \times \mathsf{Security}_i + w_{\mathsf{fe}} \times \mathsf{FareEvasion}_i + w_{\mathsf{m}} \times \mathsf{Mandatory}_i + \\ & w_{\mathsf{hr}} \times \mathsf{HighRisk}_i + w_{\mathsf{pc}} \times \mathsf{PlatformControl}_i + w_{\mathsf{tc}} \times \mathsf{TrainControl}_i,\end{aligned} \tag{18}$$

where the parcels are weights multiplied by the corresponding amounts, which are respectively: the security value of i , the fare evasion value of i, and four booleans equal to 1 or 0 depending on whether: i is a mandatory covering task or not, i is a high risk task or not, i is an platform control task or not, i is a train control task or not.

The result of decomposing value'_i into more parcels is:

$$\begin{aligned}\mathsf{value}'_i = \quad & w'_{\mathsf{s}} \times \mathsf{Security}_i + w'_{\mathsf{fe}} \times \mathsf{FareEvasion}_i + w'_{\mathsf{m}} \times \mathsf{Mandatory}_i + \\ & w'_{\mathsf{hr}} \times \mathsf{HighRisk}_i + w'_{\mathsf{pc}} \times \mathsf{PlatformControl}_i + w'_{\mathsf{tc}} \times \mathsf{TrainControl}_i,\end{aligned} \tag{19}$$

where the parcels are weights multiplied by the corresponding amounts, which are the same as in (18).

Handling all-or-nothing jobs

We could not include (11) because we would end up with a model different from (12)-(16), i.e. the model solved by the heuristic. Instead, we made two adjustments to avoid partial covering of all-or-nothing jobs: we modified the column fixing step to force some all-or-nothing jobs to become uncovered and introduced in the solution improvement step a goal related with all-or-nothing jobs (as we will see later).

In the column fixing step of the algorithm described in Abbink et al (2011) we fix all slack duties corresponding to spare all-or-nothing tasks, which in practice means forcing those tasks to be left uncovered in the solution. Spare all-or-nothing tasks are tasks from all-or-nothing jobs that are ruled out by other tasks competing for the same teams.

In order to identify spare all-or-nothing tasks we iterate over all all-or-nothing jobs to check if their tasks are spare. If so, then their corresponding slack duties are fixed. We check if the tasks M_l of an all-or-nothing job l are spare in the following way:

1. let $\tilde{M}_l$ be the set of all mandatory, high-risk and all-or-nothing tasks that overlap in time with any of the tasks in M_l;
2. let $\tilde{T}$ be the set of teams with no work assigned so far;
3. build a bipartite graph where nodes are tasks in $\tilde{M}_l$ and teams in $\tilde{T}$ and where arcs link tasks with teams and represent the possibility of assigning tasks to teams with a certain cost (given below in the description of the cost function); add fictitious nodes and arcs with infinite cost to assure there is always a one to one assignment;

```
1  algorithm IMPROVE (initial, pool, T) returns final
2    begin
3      final ← initial, current ← initial, visited ← ∅, unvisited ← initial
4      while unvisited ≠ ∅ do
5        begin
6          j ← arg max_{j'∈current} VIOLATION(current \ {j'})
7          current ← current \ {j}
8          visited ← visited ∪ {j}
9          unvisited ← unvisited \ {j}
10         while |current| ≤ 1.1 × |T| do
11           begin
12             j ← arg min_{j'∈pool \ visited} VIOLATION(current ∪ {j'})
13             current ← current ∪ {j}
14           end
15         current ← Remove worse overcovering duties from current
16         if VIOLATION(current) < VIOLATION(final) then
17           final ← current, visited ← ∅, unvisited ← current
18       end
19     return final
20   end
```

Table 1 Local search for improving a given solution

4. run the Hungarian algorithm and check in the solution if at least a task of M_l was assigned to a fictitious duty; if so, then all tasks in M_l are considered spare.

We use the following value for the cost c_{ik} of each arc between a task $i \in \tilde{M}_l$ and a team k in $\tilde{T}$:

$$c_{ik} = 10000 + 1 \times \mathsf{Mandatory}_i + 100 \times \mathsf{HighRisk}_i + 10000 \times \mathsf{PlatformControl}_i - \\ - \frac{10000 w_{\mathsf{S}}}{w_{\mathsf{S}} + w_{\mathsf{fe}}} \times \mathsf{Security}_i - \frac{10000 w_{\mathsf{fe}}}{w_{\mathsf{S}} + w_{\mathsf{fe}}} \times \mathsf{FareEvasion}_i .$$

Solution improvement step

The algorithm that improves the solution is shown in Table (1).

The algorithm uses the function VIOLATION, which receives a solution s and returns the total amount of violation to the constraints 2, 5, 6 and 11 found in s plus the cost of s computed with cost function (1).

The algorithm basically improves the solution by swapping duties between the solution itself and a given pool of duties. More precisely, it starts by removing from the solution the duty that produces the largest increase in the violation amount (line 6). Then it adds to the solution the duties found in the pool that produce the smallest increase in the violation amount (line 12) until the solution has 10% of overcovering duties (line 10). Then, it removes the worse overcovering duties (line 14). It repeats the same process (from line 6) until it is not possible to improve the solution. Overcovering duties are duties

that are assigned to a team that already has a duty assigned. The worse duties are the duties that, once removed from the solution, introduce the smallest amount of violation in the solution.

5 Experimental results

The TUTIS prototype was developed not only to support a new planning process where the plans are produced automatically in a much more detailed form but also to serve as a decision support tool for designing new ways of deploying the teams.

In order to test the prototype we ran several what-if scenarios were we solve the same problem instance under different configurations of preferences and constraints. In each scenario we see the effect of changing a particular weight of the cost function or of relaxing a particular constraint in the model.

We used a problem instance supplied by NS that corresponds to a region called *Randstad Noord*, located in the Northwest of Holland, and having the following features:

- 4 operational bases, namely Almere, Amersfoort, Alkmaar and Amsterdam;
- 87608 potential train control jobs of at most one hour;
- 5018 jobs with 128 all-or-nothing jobs among them;
- 5164 tasks with 274 all-or-nothing tasks among them;
- 29 teams with 12 backup teams among them.

The 5018 jobs were the result of job generation and include part of the 87608 potential train control jobs and all the required platform control activities broken into pieces of 0:30 length.

The default weights for the cost function of the job assignment algorithm are $\mathsf{FixedCharge} = 200000$, $w_{\mathsf{s}} = 5$, $w_{\mathsf{fe}} = 5$, $w_{\mathsf{m}} = 5000$, $w_{\mathsf{hr}} = 3500$, $w_{\mathsf{pc}} = 1750$, $w_{\mathsf{tc}} = 100$, $w'_{\mathsf{s}} = 50$, $w'_{\mathsf{fe}} = 50$, $w'_{\mathsf{m}} = 200000$, $w'_{\mathsf{hr}} = 100000$, $w'_{\mathsf{pc}} = 50000$, $w'_{\mathsf{tc}} = 10000$.

All tests were made with a maximum average duty duration $\overline{d}$ of 8:00 for all operational bases.

Table (2) shows the results obtained for the several what-if scenarios, where the execution of each scenario took from 10 to 25 hours of running time in a computer equiped with a Intel Core2Quad Q9550 2,83GHz, with 4096 MB of RAM.

In the first what-if scenario, shown in Table (2), we see the effect of widening the geographical range of the teams to the whole region. The first line (short range) corresponds to the current situation, where teams are confined to work around their own personnel base. The second line (long range) corresponds to the situation where the teams are allowed to work across the whole region. It is clearly shown that widening the geographical range of the teams increases both security (5th column) and fare evasion value (6th column), resulting in a more efficient use of the teams. Although widening the geographical range of the teams seems to be a good idea, attention should be paid to the fact that

widening too much may make it difficult for the teams to get familiar with the local layout of stations or local behaviour patterns of travellers, which may make them less effective in dissuading or stopping violence.

What-if scenario	Average duration	Train control	Platform control	Security value	Fare evasion value
Short range	7:56	51:49	25:30	20075	19831
Long range	7:57	53:58	23:30	23363	21019
Any routes	7:56	51:49	25:30	20075	19831
All routes	7:57	48:56	24:30	15992	15618
Train control	7:56	51:49	25:30	20075	19831
Platform control	7:51	27:11	66:00	14134	13571
Security	7:55	47:28	31:30	20271	18832
Fare evasion	7:56	47:48	37:30	18109	19299

Table 2 What-if scenarios. Each row represents a solution s obtained for a specific scenario. The columns display, respectively: the name of the scenario corresponding to s, the average duration of the duties in s (given in hh:mm), the total amount of time spent performing train control in s (given in hh:mm), the total amount of time spent performing platform control in s (given in hh:mm), the total amount of security value covered in s, and the total amount of fare evasion value covered in s.

In the second what-if scenario shown in Table (2) we compare a scenario (any routes) where constraints (5) are dropped with a scenario (all routes) where all network segments in S are forced to be covered at least twice. On one hand, such enforcement is interesting because it makes sure that all network segments are visited by a team at least twice a day, but at the cost of decreasing both the security and the fare evasion value.

In the third what-if scenario shown in Table (2) we compare a situation where we focus on train control with a situation where we focus on platform control. In the former (train control) we use the default parameter setting, while in the latter (platform control) we set $w_{\mathsf{tc}} = w'_{\mathsf{tc}} = 0$. The numbers in the table show that the total covered train and platform control working hours change accordingly. Overall, focusing on train control jobs seems more valuable than focusing on platform control jobs. If security experts and/or results from real life testing prove this to be a false conclusion, however, this scenario might indicate that platform control jobs have not been valued correctly (remember that these are given a value manually).

In the fourth what-if scenario shown in Table (2) we compare a situation where we focus on security with a situation where we focus on fare evasion. In the former (security) we set $w_{\mathsf{s}} = 10, w'_{\mathsf{s}} = 100, w_{\mathsf{fe}} = w'_{\mathsf{fe}} = 0$, while in the latter (fare evasion) we set $w_{\mathsf{s}} = w'_{\mathsf{s}} = 0, w_{\mathsf{fe}} = 10, w'_{\mathsf{fe}} = 100$. The numbers in the table show that the security and fare evasion values change accordingly. The scenario shows how the TUTIS prototype helps change work assignments in a goal-oriented fashion: if it is preferable to focus on preventing aggression, a user could make schedules that have a large security value, if it is preferable to focus on catching fare evaders, a user could make schedules that have a large fare evasion value.

6 Conclusions and future developments

We described security crew scheduling as a problem that adds new challenges to classical crew scheduling and showed how OR techniques can help solve it under many different conditions.

Based on our model we built a prototype to plan the work of part of NS' V&S staff. Preliminary results show that it can bring significant improvements to V&S planning and provide decision support on developing the V&S planning process into a centralized, nation-wide form. It could also be used to link performance indicators to V&S duties and study the effects of changes in labour rules and personnel availability on these indicators. Experimental results show that the prototype not only presents meaningful results, but also can help decide the way V&S staff will be deployed in the future.

In February 2015, NS has started a pilot with 6 operational bases in which a fully implemented version of the prototype has been tested in a production environment. In the duration of 3 months, about 4000 duties were constructed by personnel scheduling specialists who were (re)trained for the use of TUTIS. These duties contained approximately 25000 different train control tasks and events with an elevated security risk, and they have impacted the work of more than 250 V&S security guards. For the purposes of the pilot, the prototype was tested and improved extensively. Changes to the model include the use of different kinds of duty templates, implementing the backup concept as a constraint rather than as a duty template, extensive changes to labour rules and more focus in both job generation and job assignment on the importance of having V&S staff available for surveillance on stations. The evaluation of this pilot is forthcoming.

Acknowledgements We thank all who contributed to this work. Hans Munk initialized the research, provided the data, and gave invaluable insights on the work processes of V&S teams. Luís Albino, Jorge Roussado, Rudi Araújo and Filipa Morgado developed the prototype. Luís and Jorge gave essential contributions on the solution method for job assignment. Finally, we would like to thank Martin van Meerkerk for the dedication he put into working with us on job generation.

References

Abbink EW, Albino L, Dollevoet T, Huisman D, Roussado J, Saldanha RL (2011) Solving large scale crew scheduling problems in practice. Public Transport 3(2):149–164

Huisman D (2007) A column generation approach for the rail crew re-scheduling problem. European Journal of Operational Research 180(1):163–173

Lau HC, Gunawan A (2012) The patrol scheduling problem, practice and theory of automated timetabling. In: PATAT, pp 175–192

Meerkerk M (2014) Scheduling security & service personnel at netherlands railways. Master's thesis, Utrecht University

Morgado E, Martins JP (1998) Crews-ns: Scheduling train crew in the netherlands. AI Magazine 19(1):25–38

Ordóñez F, Tambe M, Jara JF, Jain M, Kiekintveld C, Tsai J (2013) In: Handbook of Operations Research for Homeland Security, Springer, New York, pp 45–72

Pita J, Jain M, Marecki J, Ordóñez F, Portway C, Tambe M, Western C, Paruchuri P, Kraus S (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 Conference on Autonomous Agents and Multiagent Systems: industrial track, pp 125–132

Pita J, Tambe M, Kiekintveld C, Cullen S, Steigerwald E (2011) Guards: Game theoretic security allocation on a national scale. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, pp 37–44

Potthoff D, Huisman D, Desaulniers G (2010) Column generation with dynamic duty selection for railway crew rescheduling. Transportation Science 44(4):493–505

Shieh E, An B, Yang R, Tambe M, Baldwin C, DiRenzo J, Maule B, Meyer G (2012) Protect: A deployed game theoretic system to protect the ports of the united states. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, pp 13–20

SISCOG (2015) www.siscog.eu

Teamcore (2015) Teamcore research group, university of southern california. http://teamcore.usc.edu/projects/security

Thorlacius P, Clausen J, Brygge K (2010) Scheduling of inspectors for ticket spot checking in urban rail transportation. In: Trafikdage ved Aalborg Universitet 2008, http://research.create.usc.edu/nonpublished_reports/71

Tsai J, Kiekintveld C, Ordóñez F, Tambe M, Rathi S (2009) Iris - a tool for strategic security allocation in transportation networks. Tech. rep., http://research.create.usc.edu/nonpublished_reports/71

Yin Z, Jiang AX, Johnson MP, Kiekintveld C, Leyton-Brown K, Sandholm T, Tambe M, Sullivan JP (2012) Trusts: Scheduling randomized patrols for fare inspection in transit systems. In: Proceedings of the 24th Innovative Applications of Artificial Intelligence Conference (IAAI-12), pp 2348–2355