Public Transp (2017) 9:193–215
DOI 10.1007/s12469-016-0142-y

ORIGINAL PAPER

Decision support for scheduling security crews at Netherlands Railways

Hilbert Snijders[1] · Ricardo L. Saldanha[2]

Accepted: 25 October 2016 / Published online: 3 November 2016

Abstract We address the problem of scheduling work of security guards operating on trains and stations, and explain how operations research is suitable for solving a problem that adds new challenges to classical crew scheduling. Planning the work of these security guards is challenging because it requires dealing not only with the complexity inherent to crew scheduling problems but also with an optimisation goal that itself is defined more in qualitative than quantitative terms. Our contribution to handling these challenges is summarised as follows:

- we describe the problem and model it as a shortest path problem and a set covering problem with additional constraints;
- we adapt an existing heuristic that is based on Lagrangian relaxation, subgradient optimisation, column generation and greedy heuristics to be applicable to the problem (for instance, we add an improvement step based on local search);
- we implement the resulting solution method, resulting in a software prototype named TUTIS;
- we test TUTIS with a real problem instance supplied by Netherlands Railways.

Experimental results lead security experts to believe 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.

✉ Ricardo L. Saldanha
rsaldanha@siscog.pt

Hilbert Snijders
hilbert.snijders@ns.nl

1 Netherlands Railways, Laan van Puntenburg 100, Postbus 2025, 3500 HA Utrecht, The Netherlands

2 SISCOG-Sistemas Cognitivos, SA, Lisbon, Portugal

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 5000 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 and reported. NS employs around 700 so-called Veiligheid and Service (VS or Security and 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 VS 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 VS 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 (2016). The problem at hand is concerned with assigning work (jobs) to a fixed set of duties. It is a crew scheduling problem with a planning horizon of 1 day (rostering is out of the scope). Currently, jobs are distributed among duties locally by team managers or roster schedulers. They use generalised (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 VS duties are scheduled centrally and in greater detail than in the current planning process, hypothesising that a central, detailed approach will ultimately help improve social safety on the Dutch railway network.

This paper describes a first attempt to solve this problem and shows how operations research techniques can provide decision support to improve the planning process of V&S teams. Our contribution to handling this problem is summarised as follows:

- we describe the problem and model it as a shortest path problem and a set covering problem with additional constraints;
- we adapt an existing heuristic that is based on Lagrangian relaxation, subgradient optimisation, column generation and greedy heuristics to be applicable to the problem (for instance, we add an improvement step based on local search);
- we implement the resulting solution method, resulting in a software prototype;
- we test the prototype with a real problem instance supplied by Netherlands Railways.

The software prototype was named TUTIS and was built as an extension of CREWS (Morgado and Martins 1998), a software package that is being used in production at NS to schedule the work of train drivers and regular train guards. CREWS incorporates the aforementioned Lagrangian-based heuristic, which is known in the literature (see Abbink et al. 2011). This heuristic is capable of producing highly efficient schedules for drivers and guards but is not prepared to handle new challenges brought by a security crew. This paper mainly describes how this heuristic was extended to handle security-related issues like evaluating risk and optimising risk mitigation.

The remainder of this paper is organised as follows. Section 2 provides a short literature review of security patrol scheduling problems, while Sect. 3 is concerned with the problem description. In Sect. 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 Sect. 6.

2 Literature review

In recent years, the most prevalent methods of modelling scheduling of security patrols have involved 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 optimise 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 2016) 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 optimising the income from penalty fares (see Thorlacius and Clausen 2010). A case study that shares some properties with the fare evasion part of the NS crew scheduling problem for V&S teams is the optimisation of mobile toll inspector tours on German motorways (see Borndörfer et al. 2012). In this case, a Multi-Commodity flow problem, that is solved to determine optimal routes, and another Multi-Commodity flow problem, that is solved to assign inspectors to these routes, are integrated using coupling constraints. The result is a monthly roster for the inspectors. In general, it can be said that currently 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, although it seems to be growing steadily.

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.

3.1 The context of the problem

The scheduling problem that is the focus of this paper is the second step of a two-step process.

The first step is performed during a period of time that takes place from a few months until 1 week before the day of operations. It involves creating rosters, assigning these to V&S employees and updating the rosters when required. In this context a roster is a working schedule for several months specified as a sequence of *empty duties* interleaved with periods where employees are supposed to rest or take days off according to the established labour agreement. Empty duties are periods of time that start (end) with a sign in (sign out) operation that takes place at the employee's home base[1] and that starts (ends) inside the time interval $[s, s]$ ($[s, s + 8{:}30\ ]$), where s is specified for each empty duty separately. Between sign in and sign out the employee is supposed to do something that will be defined later.

[1] Also known in the literature as home depot, the home base is the location in the railway network where the employee starts (finishes) working after (before) a resting period or a day off.

Some of the empty duties are identified as *backup* empty duties. In these duties, between sign in and sign out, V&S employees should stay at a specific station within the network, where they should be ready to be called at any time to perform urgent activities that might pop up during the day of operations. If no urgent situation occurs employees perform low priority activities on that station.

The second step (our problem) is performed from 5 to 7 days before the day of operations. On day D (when the aforementioned risks can be evaluated with a reasonable accuracy) risk mitigation activities for day $D+n$ are identified and assigned (between sign in and sign out operations) to empty duties found in the employees' rosters for day $D+n$ (where $5 \leq n \leq 7$). In the assignment of activities to empty duties, roster feasibility is assured by checking the temporal and spatial constraints defined for each individual empty duty. In the remainder of this document we use the term *Problem* to address the problem corresponding to this second step.

3.2 The problem itself

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.[2] If we want to make an efficient schedule, we should ensure that the limited number of teams available

[2] A similar concept is described in Borndörfer et al. (2016), where a profit value varying over time is associated with each motorway section.

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.

On day D planners pick the empty duties found in the teams' rosters for day $D + n$ and assign jobs to these. So, henceforth, when we say that a job is being assigned to a particular empty duty, we are also implicitly saying that it is being assigned to the team that has the corresponding empty duty planned in its roster for day $D + n$. Furthermore, when we say that a job is being assigned to a particular team, we are also implicitly saying that it is being assigned to the empty duty planned in its roster for day $D + n$.

Empty duties with assigned jobs are called *duties*. Different duties may represent the same empty duty with different work assignments, different empty duties with the same work assignment or different empty duties with different work assignments. All duties are subject to several constraints. Some of them apply to each separate duty (horizontal), while others affect sets of duties (vertical). For example:

- the assigned work is a sequence of jobs that must occur between a sign in and a sign out operation that comply with the spatial and temporal constraints defined for the corresponding empty duty (horizontal);
- the sequence of jobs in a duty must be continuous in space and time (horizontal);
- a team cannot work more than a certain amount of hours without having a meal break (horizontal);
- consecutive jobs in a duty must be separated by sufficient time to take the possibility of delays into account (horizontal);
- teams can only perform jobs that are geographically inside the operational range of the home base to which the teams belong (horizontal);
- teams performing backup duties can only be assigned optional platform control jobs on platforms of the station specified in the corresponding empty duty (horizontal);

- the average length of duties of teams belonging to the same home base is upper bounded (vertical);
- the number of times a railway network segment should be covered by the whole set of duties is lower bounded (vertical).

Figure 1 illustrates how train control and platform control jobs are assigned to an empty duty. It shows a Gantt chart representation of a duty of a team based in Alkmaar (Amr). 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 large vertical bar at 6:00 represents the sign in start time specified for the empty duty. The horizontal lines with abbreviated names written below the horizontal lines represent train control jobs, while the horizontal lines without names written below the horizontal lines represent platform control jobs. The abbreviated names corresponding to the train control jobs represent the ending stations of the jobs. Between 10:15 and 10:45 the team takes a meal break in Amr. As shown, train control typically assumes the form of a sequence of small train trips (because of the maxing out of value mentioned above). By looking at the picture one can easily entail that the number of possible (partial) assignments of thousands of jobs to dozens of empty duties (i.e. what happens in real life problem instances) is huge. This means that a brute force algorithm that enumerates all possible assignments first and chooses the best one second will not be usable in practice because it will take too much time to provide a solution.

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 need to handle large optimisation problem instances, where the number of jobs available to be assigned to empty duties can reach 85,000 in a single region.

4 Solution method

In order to solve the problem 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

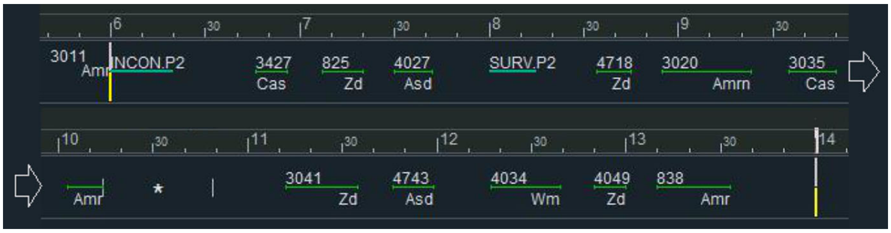

Fig. 1 A Gantt chart representation of a duty with assigned train control and platform control jobs

Problem, 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. The variables that are not significantly correlated include the time of the report (hour, part of the day and day of the week considered separately), the travelling direction on the railway network, passenger flows, geographical region and rolling stock type. 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. Let job 1 correspond to a team entering the train at Ut and leaving the train at Asa, and let job 2 correspond to a team entering the train at Ut and leaving the train at Asd. 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 gets fare evasion value for the full 28 min trip ($\text{FareEvasion}_1 = \frac{5}{7} \times min(28\ ,35) = 20$). Job 2 only receives value for 35 min of its duration, while the trip actually lasts 37 min ($\text{FareEvasion}_2 = \frac{5}{7} \times min(37\ ,35) = 25$)). For a graphical illustration, see Fig. 2.

The job representing working on our train of interest can be found in Fig. 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

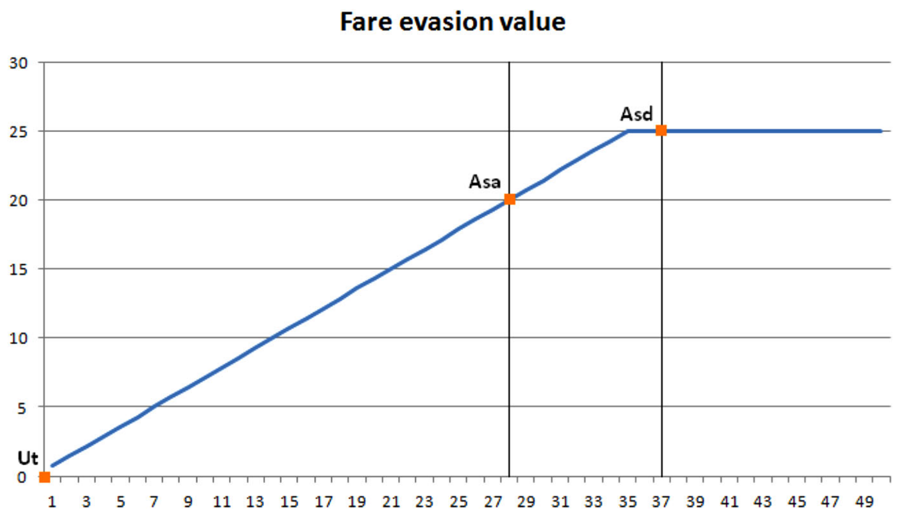

Fig. 2 An example of the way fare evasion value is assigned to jobs corresponding to a single train as a result of the duration of the job; the *x-axis* represents the duration of the job in minutes, the *y-axis* represents fare evasion value, the *orange dots* represent points where a V&S team could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

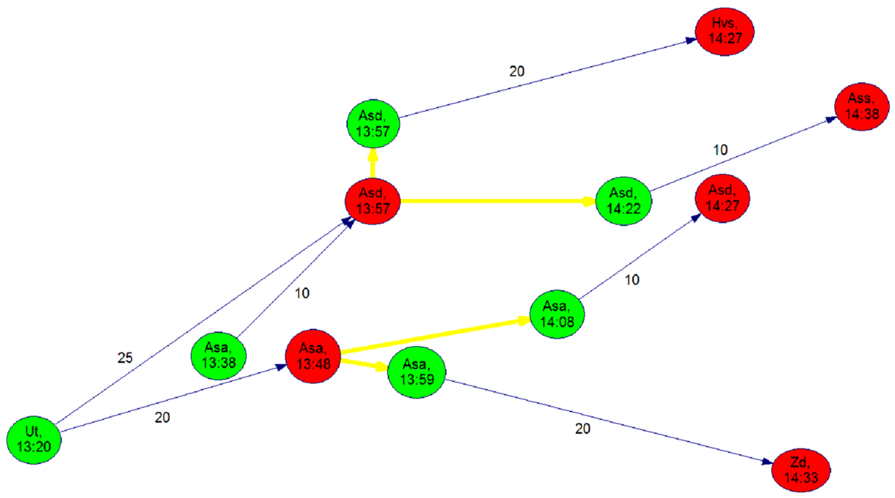

Fig. 3 A graphical representation of train control jobs (*blue arrows*) and waiting times at stations (*yellow arrows*), where the *numbers near the arrows* represent the fare evasion values of the jobs, the *green nodes* represent departures at stations, while the *red nodes* represent arrivals

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 in order to avoid checking the same passengers twice (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 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 Dijkstra's algorithm. 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 Fig. 1) and of assigning them to a given set of teams, so that the maximum amount of value is assigned and any relevant constraints are satisfied (see Sect. 3.2 for some examples of these constraints). 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 relatively small 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 empty duties); 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.

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 a *task*. 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 candidate 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 home bases. For every base $b \in B$ we denote N_b as the duties belonging to b, which is given by $N_b = \{j \in N | k_j \in T_b\}$, where T_b is the set of teams belonging to 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:

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

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

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

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

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

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

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

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

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 home 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.

4.2.2 Cost function

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

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

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

The parcel FixedCharge is a positive fixed cost large enough to keep c_j positive. In each parcel—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 proportional to the value associated with it. By replacing (9) and (10) in (1) we get:

$$\sum_{j \in N} \left(\mathsf{FixedCharge} - \sum_{i \in M_j} \mathsf{value}_i \right) x_j + \sum_{i \in \mathcal{M}_o} \mathsf{value}'_i \vartheta_i. \tag{11}$$

4.2.3 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 $l \in A$ is associated with a set of all-or-nothing tasks $M_l \subset \mathcal{M}_o$. 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):

$$\sum_{j \in N} a_{ij} x_j + \vartheta_{i_l} = 1, \forall i \in M_l, \forall l \in A. \tag{12}$$

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.

4.2.4 Algorithm

Instead of solving the JAP with an exact approach, we adopt a heuristic because 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.

Instead of building a heuristic from scratch, we decide to use the heuristic described in Abbink et al. (2011) as a baseline, which henceforth will be referred to as the *baseline heuristic*. This heuristic is based on Lagrangian relaxation, subgradient optimisation, column generation and greedy heuristics, and obtained efficient solutions for crew scheduling problems, related with drivers and regular guards of NS, which are characterised for having labour rules that are very similar to the ones defined in the problem. This means that one of our major contributions is to modify the baseline heuristic so that it becomes capable of solving the JAP. Before describing those modifications in detail, we briefly present the baseline heuristic.

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

$$\text{Minimize } c^T x \tag{13}$$

$$\text{Subject to} \quad Ax \geq 1, \tag{14}$$

$$b' \geq B'x \geq b, \tag{15}$$

$$x \in \{0, 1\}, \tag{16}$$

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

In the heuristic several iterations are run, in each of which new variables with negative reduced cost are added to the restricted master problem. The variables (representing candidate duties) are generated by a pricing procedure that uses as a starting point the solution of a shortest path problem over the network of connections. The algorithm looks for variations over the shortest path (including the shortest path itself) and it collects the ones that have negative reduced cost and that are feasible. A path is feasible if it satisfies some additional constraints such as the meal break constraint. The network of connections is a graph where nodes represent tasks and arcs represent the possibility of two tasks be performed consecutively by the same crew member in the same duty. For more details, see Abbink et al. (2011).

The reduced cost is computed based on Lagrangian multipliers [associated with constraints (14–15)], which in turn are computed with the subgradient optimisation method. For more details, see Abbink et al. (2011).

Primal solutions are obtained with a greedy algorithm that uses Lagrangian multipliers to obtain solutions. At early stages the algorithm typically obtains solutions that are infeasible with respect to the additional constraints, but after some iterations, when the multipliers get closer to their optimal values, the algorithm returns completely feasible solutions (unless there are none in the master problem). For more details, see Abbink et al. (2011).

An important technique used by the heuristic is column fixing. This means that some duties with the most negative reduced cost are temporarily forced to be part of the solution. The purpose of this action is to force the heuristic to focus on a particular region of the solution space. After some iterations the fixes are released and new columns are chosen to be fixed, and the process is repeated until it is terminated. For more details see Abbink et al. (2011).

We now describe the modifications performed over the baseline heuristic so that it becomes capable of solving the JAP.

4.2.5 Handling optional tasks

The challenge here is to make the cost function (11) look like (13) and covering constraints (3) look like (14). To achieve that, we rewrite (11) and (3) in the following way:

$$\sum_{j \in N} \left(\mathsf{FixedCharge} - \sum_{i \in M_j} \mathsf{value}_i \right) x_j + \sum_{i \in \mathcal{M}_o} \mathsf{value}'_i x_{j_i}, \tag{17}$$

$$\sum_{j \in N} a_{ij} x_j + x_{j_i} \geq 1 \tag{18}$$

where j_i is a special duty, called a slack duty, that (i) covers only task i; (ii) assigns i to a fictitious team that is not in T; and (iii) belongs to a fictitious home base that is not in B. When $x_{j_i} = 1$, it means that slack duty j_i is selected to be part of the solution, which in turn means that task i is left uncovered. With this modification the total set of duties that can be used to cover tasks becomes $N \cup \{j_i | i \in \mathcal{M}_o\}$, which means that the size of vector x in (13–16) is $|N| + |\mathcal{M}_o|$.

4.2.6 Handling teams

We first relax the duty assignment constraints (4) into regular covering constraints like (14), namely:

$$\sum_{j \in N_k} x_j \geq 1 \forall k \in T \tag{19}$$

By making this relaxation we allow having teams with two or more duties assigned in the solution. We avoid this from occurring in the final solution 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 procedure is run individually for each team (and not for each home base and each day of the week, as with the baseline heuristic). By doing so, we are able to run the pricing procedure over a connection network that enforces all the constraints related with a particular team (e.g. the ones that make sure the duty complies with the spatial and temporal constraints specified by the corresponding empty duty), and we are also able to associate the generated duties with the corresponding team.[3] This also generates the opportunity to run the pricing algorithm in parallel for several teams at the same time.

4.2.7 Cost function

Because we need cost function (17) to guide the heuristic towards the desired direction and also to allow us to obtain solutions that vary over certain dimensions, we made the adjustments described below.

The parcel $\mathsf{FixedCharge}$ is set to a value large enough to help the heuristic minimise the number of duties in the solution.

The coefficient value_i is decomposed in the following way:

$$\begin{aligned} \mathsf{value}_i = {} & w_{\mathsf{s}} \times \mathsf{Security}_i + w_{\mathsf{fe}} \times \mathsf{FareEvasion}_i + w_{\mathsf{m}} \times \mathsf{Mandatory}_i \\ & + w_{\mathsf{hr}} \times \mathsf{HighRisk}_i + w_{\mathsf{pc}} \times \mathsf{PlatformControl}_i + w_{\mathsf{tc}} \times \mathsf{TrainControl}_i, \end{aligned} \tag{20}$$

where the parcels are weights multiplied by corresponding factors, which are respectively: the security value of i, the fare evasion value of i, and four booleans that signify whether: i is a mandatory covering task, i is a high risk task, i is a platform control task, i is a train control task.

The coefficient value'_i is decomposed in the following way:

$$\begin{aligned} \mathsf{value}'_i = {} & w'_{\mathsf{s}} \times \mathsf{Security}_i + w'_{\mathsf{fe}} \times \mathsf{FareEvasion}_i + w'_{\mathsf{m}} \times \mathsf{Mandatory}_i \\ & + w'_{\mathsf{hr}} \times \mathsf{HighRisk}_i + w'_{\mathsf{pc}} \times \mathsf{PlatformControl}_i + w'_{\mathsf{tc}} \times \mathsf{TrainControl}_i, \end{aligned} \tag{21}$$

where the parcels are weights multiplied by corresponding factors, which are the same as in (20).

[3] Borndörfer et al. (2013) use a similar approach applied to duty templates.

4.2.8 Handling all-or-nothing jobs

We could not include (12) because we would end up with a model different from (13–16), i.e. the model solved by the baseline 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;
4. run the Hungarian algorithm and check in the solution if at least a task of M_l was assigned to a slack 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 $\tilde{T}$:

$$\begin{aligned} c_{ik} = {} & 10{,}000 + 1 \times \mathsf{Mandatory}_i + 100 \times \mathsf{HighRisk}_i + 10{,}000 \times \mathsf{PlatformControl}_i \\ & - \frac{10{,}000 w_{\mathsf{s}}}{w_{\mathsf{s}} + w_{\mathsf{fe}}} \times \mathsf{Security}_i - \frac{10{,}000 w_{\mathsf{fe}}}{w_{\mathsf{s}} + w_{\mathsf{fe}}} \times \mathsf{FareEvasion}_i. \end{aligned}$$

4.2.9 Solution improvement step

The algorithm that improves the solution is shown in Algorithm 1.

Algorithm 1 Local search for improving a given solution

```
1: procedure IMPROVE($initial, pool, T$)
2:     $final \leftarrow initial, current \leftarrow initial, visited \leftarrow \emptyset, unvisited \leftarrow initial$
3:     while $unvisited \neq \emptyset$ do
4:         $j \leftarrow \arg\max_{j' \in current} \text{VIOLATION}(current \setminus \{j'\})$
5:         $current \leftarrow current \setminus \{j\}$
6:         $visited \leftarrow visited \cup \{j\}$
7:         $unvisited \leftarrow unvisited \setminus \{j\}$
8:         while $|current| \leq 1.1 \times |T|$ do
9:             $j \leftarrow \arg\min_{j' \in pool \setminus visited} \text{VIOLATION}(current \cup \{j'\})$
10:            $current \leftarrow current \cup \{j\}$
11:        end while
12:        $current \leftarrow$ Remove worse overcovering duties from $current$
13:        if VIOLATION($current$) < VIOLATION($final$) then
14:            $final \leftarrow current, visited \leftarrow \emptyset, unvisited \leftarrow current$
15:        end if
16:    end while
17:    return $final$
18: end procedure
```

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 (12) 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 4). Then it adds to the solution the duties found in the pool that produce the smallest increase in the violation amount (line 9) until the solution has 10% of overcovering duties (line 8). Then, it removes the 'worse' overcovering duties (line 12). It repeats the same process (from line 4) 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. 'Worse' duties are defined as duties that, once removed from the solution, introduce the smallest amount of violation in the solution.

The algorithm does not guarantee the removal of all constraint violations. However, practice has shown that all violations are removed when the amount of constraint violation in s is small enough, which is something that normally happens in the later iterations of the heuristic. So, with this improvement step the heuristic returns feasible solutions in practice unless the problem is over-constrained from the beginning (e.g. there are more mandatory tasks than the available teams can handle).

5 Experimental results

In order to be able to use in practice the job generation and job assignment heuristics described in Sects. 4.1 and 4.2, a software prototype called TUTIS was developed as an extension of CREWS, a software product (see Morgado and Martins 1998) that has been used in production for several years at NS to produce and update work schedules from long-term planning to real-time dispatching for train drivers and regular train guards. With TUTIS it is not only possible to run the job assignment heuristic for several scenarios but also to perform useful operations with the obtained solutions, such as: view, edit, validate, make reports, etc. The heuristic is implemented in C++ as an extension to an existing heuristic that is part of CREWS (see Abbink et al. 2011).

The TUTIS prototype was used as a decision support tool for studying the impact of changing certain conditions and assumptions under which job assignments are performed. Below we present the results obtained when running several what-if scenarios where 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 that has the following features:

– 4 home bases, namely Almere, Amersfoort, Alkmaar and Amsterdam;
– 87,608 potential train control jobs of at most 1 h;
– 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 87,608 potential train control jobs and all the required platform control activities broken into segments of 0:30 length.

The default weights for the cost function of the job assignment algorithm are $\mathsf{FixedCharge} = 200,000$, $w_{\mathsf{s}} = 5$, $w_{\mathsf{fe}} = 5$, $w_{\mathsf{m}} = 5000$, $w_{\mathsf{hr}} = 3500$, $w_{\mathsf{pc}} = 1750$, $w_{\mathsf{tc}} = 100$, $w'_{\mathsf{s}} = 50$, $w'_{\mathsf{fe}} = 50$, $w'_{\mathsf{m}} = 200,000$, $w'_{\mathsf{hr}} = 100,000$, $w'_{\mathsf{pc}} = 50,000$, $w'_{\mathsf{tc}} = 10,000$.

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

Table 1 Results obtained from the execution of several what-if scenarios

What-if scenario	Average duration	Train control	Platform control	Security value	Fare evasion value
Short range	7:56	51:49	25:30	20,075	19,831
Long range	7:57	53:58	23:30	23,363	21,019
Any routes	7:56	51:49	25:30	20,075	19,831
All routes	7:57	48:56	24:30	15,992	15,618
Train control	7:56	51:49	25:30	20,075	19,831
Platform control	7:51	27:11	66:00	14,134	13,571
Security	7:55	47:28	31:30	20,271	18,832
Fare evasion	7:56	47:48	37:30	18,109	19,299
30 teams	7:57	61:05	23:30	23,168	22,784
45 teams	7:59	115:25	37:00	39,519	41,172

Table 1 shows the results obtained for the several 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. The execution of each scenario took from 10 to 25 h of running time in a computer equipped with a Intel Core2Quad Q9550 2,83GHz, with 4096 MB of RAM.

In the first what-if scenario, shown in Table 1, 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.

In the second what-if scenario shown in Table 1 we compare a scenario (any routes) where constraints (5) are dropped [i.e. $s_n = 0$ in (5)] with a scenario (all routes) where all network segments in S are forced to be covered at least twice [i.e. $s_n = 2$ in (5)]. 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 1 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 1 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.

In the fifth what-if scenario shown in Table 1 we study the impact of increasing workforce capacity while keeping the same workload. We compare a situation where job assignment is performed with 5164 tasks and 30 teams with a situation where it is performed with the same 5164 tasks and 45 teams (50% increased capacity). Results show that we get an increase of circa 71% of security (81% of fare evasion) value when we increase the number of teams in 50%. This result suggests that hiring more staff brings additional costs to the company but also the opportunity to mitigate more risk. The fact that value increases with a higher rate than capacity might be due to the fact that the 15 additional teams were not performing backup duties, which means that the scenario with 45 teams has, in comparison with the scenario with 30, relatively less backup teams, i.e. less teams that can only perform platform control tasks with low value.

Another experiment was performed to study how sensitive job assignment is to changes of fare evasion and security values of jobs. We made several runs with the same set of jobs and teams but with different amounts of perturbation introduced in the security and fare evasion values of the tasks. For each task i we changed the corresponding security (fare evasion) value from $\mathsf{Security}_i$ to $(1+p)\mathsf{Security}_i$ (from $\mathsf{FareEvasion}_i$ to $(1+p)\mathsf{FareEvasion}_i$), where $p \in [-\pi, \pi]$ is randomly generated for each task (with an uniform distribution), and where $\pi \in [0, 1[$ is fixed for each run. Each line of Table 2 shows (in column 2) the percentage of tasks shared among a reference solution obtained with a zero perturbation ($\pi = 0$) and a solution obtained with a specific perturbation π. The results show that if more perturbation is introduced, the set of tasks covered with respect to the reference solution becomes more diverse. The fact that the rate, at which the number of shared tasks decreases, is smaller for larger perturbations is due to the fact that the mandatory tasks force all the solutions to share at least a minimum amount of tasks. The reason why the amount of shared tasks in the first scenario is not 100% is that the heuristic is non deterministic and there are many solutions with the same covered value because tasks often share the same security and fare evasion values.

From a practical point of view the behaviour shown in the first scenario is interesting because the heuristic can produce different solutions with identical quality, allowing end users to compare and choose the one that best satisfies preferences that cannot be easily expressed in a cost function. End users can overcome the inconvenience of getting different solutions when making several runs over the same problem instance by using the same random number generator seed (used by the heuristic) on every run.

Although the main purpose of the TUTIS prototype is to help improve the planning process and not its final output (the operational schedules), one can argue about the quality of the solutions presented above. We can say that most of the

Table 2 Sensitivity of solutions to value computation

Max perturbation introduced (value of π in %)	Shared tasks (%)
0	73
10	45
20	43

solutions presented above were evaluated by NS' security experts by visual inspection and received positive feedback. Unfortunately, there was no other way to assess the quality of the solutions. First, it was not possible to compare them with existing solutions because the latter are produced with much less level of detail. Second, job generation cannot be evaluated by computing the value of an objective function. It requires further evaluation to see whether risk mitigation values are being computed in a correct way. Only practice over a long period of time can show if a certain solution method decreases the occurrence of fare evasion and security faults. Third, it is not possible to compare the obtained job assignment solutions with an exact approach because the number of decision variables of real life problem instances is in the order of many millions, which make the solvers run out of memory. In favour of the quality of the produced solutions we can also say that the heuristic that was used as a baseline for our job assignment approach (see Abbink et al. 2011) is currently being used in production at NS and obtains very efficient solutions for train drivers and regular train guards. As described in Abbink et al. 2011, it was able to produce solutions that were more efficient than existing solutions that already represented a great improvement in comparison with solutions produced manually by human planners. Moreover, the baseline heuristic obtains solutions with an overestimated gap to optimality ranging from 0.80% to 1.70% in crew scheduling problems with a number of tasks ranging from 380 to 1280.[4]

6 Conclusions and future developments

We described security crew scheduling as a problem that adds new challenges to classical crew scheduling and showed how Operations Research techniques can help solve it. More precisely, we modelled the problem as a set covering problem with additional constraints, and we adapted an existing heuristic that is based on Lagrangian relaxation, subgradient optimisation, column generation and greedy heuristics to be applicable to the problem. Based on our model we built a software prototype named TUTIS (as an extension of the CREWS software package) to plan the work of part of NS' V&S staff. Preliminary results indicate that it can provide decision support on developing the V&S planning process into a centralised, 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. NS' security experts claim 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 home 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 25,000 different train control tasks and events with

[4] These gaps were obtained in the following way: we ran a problem with the baseline heuristic and with a similar heuristic that uses full enumeration of variables; we then computed the gap by subtracting the upper bound obtained by the baseline heuristic from the lower bound obtained by the full enumeration heuristic (which corresponds to the master problem of the column generation approach).

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 empty duties, implementing the backup concept as a constraint rather than as an empty duty, 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 Transp 3(2):149–164

Borndörfer R, Sagnol G, Swarat E (2012) A case study onoptimizing toll enforcements onmotorways. In: Ravizza S, Holborn P (eds) Proceedings of the 3rd student conference on operational research. Open acess series in informatics, vol 22, pp 1–10

Borndörfer R, Langenhan A, Löbel A, Schulz C, Weider S (2013) Duty scheduling templates. Public Transp 5(1–2):41–51

Borndörfer R, Sagnol G, Schlechte T, Swarat E (2016) Optimal duty rostering for toll enforcement inspectors. Ann Oper Res pp 1–24

Huisman D (2007) A column generation approach for the rail crew re-scheduling problem. Eur J Oper Res 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 and service personnel at Netherlands Railways. Master's Thesis, Utrecht University

Morgado E, Martins JP (1998) Crews-ns: scheduling train crew in The Netherlands. AI Mag 19(1):25–38

Ordóñez F, Tambe M, Jara JF, Jain M, Kiekintveld C, Tsai J (2013) Handbook of operations research for homeland security. Springer, New York

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. Transp Sci 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 (2016) http://www.siscog.eu

Teamcore (2016) Teamcore research group, University of Southern California. http://teamcore.usc.edu/projects/security

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

Tsai J, Kiekintveld C, Ordóñez F, Tambe M, Rathi S (2009) Iris—a tool for strategic security allocation in transportation networks.In: Proceedings of the 8th international conference on autonomous agents and multiagent systems (AAMAS 2009)

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