Erasmus University Rotterdam
Erasmus School of Economics
Master Thesis Analytics and Operations Research In Logistics

Railway crew planning with fairness considerations

Stef van der Vlis (541120)

Supervisor:	Twan Dollevoet
Second assessor:	Bart van Rossum
Date final version:	May 1, 2024

Abstract

The crew planning problem is a challenging problem for railway operators across the globe. In recent times, there has been an increasing emphasis on equitably distributing attractive and unattractive tasks among employees in the research on this problem. We contribute to the literature on this subject by formulating a new solution approach which sequentially solves the crew scheduling and crew rostering problem with tailored column generation algorithms. We integrate fairness conditions into the latter of these algorithms and define attractiveness based on the Sharing Sweet & Sour attributes of NS. We test our methods on an instance based on a part of the railway network in the Netherlands. We test a model that accounts for both fairness as well as efficiency against a benchmark model that only considers efficiency as the objective. The results of the bi-objective model show a substantial improvement in fairness with only a minor reduction in efficiency compared to the benchmark model.

Keywords: Crew scheduling, Crew rostering, Fairness, Column generation

Contents

1 Introduction

Railway operators face a number of challenging problems in their operations. One such problem is that of crew planning, which consist of allocating the work on the trains to the companies' drivers and guards. Especially for large operators, such as Netherlands Railways (NS), this is a complex puzzle to solve. Therefore, these companies rely on Operations Research techniques that help in creating the rosters for their crews. The crew planning process can be broken down into two smaller problems. The first of these is the crew scheduling problem, where single pieces of work, i.e. trips from one station to another are combined into duties, which represent a shift for a crew member. In the second step duties are combined into rosters. A roster consists of duties an employee needs to cover for a given period. This second problem is called the crew rostering problem.

One obvious way to measure the quality of the resulting rosters is by its efficiency, which can be measured by the number of employees needed to cover all rosters or, alternatively, by the costs of the rosters. However, in recent research on the crew planning problem, there has been more focus on taking fairness into account. A fair solution of the crew planning problem is one in which relatively favourable and relatively unfavourable work is divided evenly over the employees. Creating fair rosters means we ensure better working conditions for the employees. This has multiple benefits. First of all, better working conditions makes the railway operator a more attractive employer, which is crucial in today's competitive labor market. Secondly, improved working conditions can improve the overall happiness and productivity of the employees. Finally, it decreases the risks of strikes. Strikes are not uncommon in the railway industry. For instance, in 2001, a conflict about the rosters at NS led to nationwide strikes. To resolve the conflict, the 'Sharing Sweet & Sour' rules were introduced. These rules indicate what elements make work desirable and undesirable for the drivers and guards (Abbink, Fischetti, Kroon, Timmer, & Vromans, 2005). These rules were incorporated in the crew planning operations of NS by distributing the sweet and sour work evenly over the different crew bases.

Another element of the current crew planning process which helps in achieving some level of fairness is the idea of cyclic rosters. A roster consists of multiple weeks of work. A group of crew members, called a roster group, cycles through the weeks of such a roster. This ensures that every crew member in a roster group gets the exact same amount of desirable and undesirable work. Unfortunately, these two modifications cannot ensure that the rosters are fair at an individual level. The 'Sharing Sweet & Sour' rules try to achieve a fair allocation over the crew bases and the cyclicity of the rosters guarantee fairness within a roster group. Unfortunately this means that considerable differences in fairness can still exist between roster groups.

In an attempt to better incorporate fairness, new methods have been introduced recently, such as those presented by van Rossum, Dollevoet, and Huisman (2022) and Breugem, Dollevoet, and Huisman (2022). In this thesis, we aim to contribute to the existing literature with a new approach to crew planning that accounts for fairness at the individual level. To this end we create personal rosters, that differ for each individual crew member, rather than cyclic rosters. Furthermore, we do not determine a capacity planning beforehand. A capacity planning includes general rosters for each crew member specifying the days they are supposed to be working, without any exact duties specified yet. We construct the actual rosters containing the exact duties directly, combining the tactical and operational planning phase into one. Thus, our research presents a novel way to tackle the crew planning problem with fairness considerations. We use two column generation algorithms, one for the crew scheduling step and another for the rostering step. The pricing problems of these column generation algorithms can be modeled as (resource constrained) shortest path problems, for which we design tailored solution methods. In the rostering step, we incorporate the idea of fairness by ensuring a minimum attractiveness level for all rosters. We apply these methods on a practical instance based on the timetable of NS and compare our method to a similar algorithm that does not account for fairness. This algorithm is our benchmark model. As such, we can see and discuss the difference in both efficiency and fairness between them.

We test our models on an instance based on a part of the timetable of NS. Our results show that the efficiency of a solution decreases if we incorporate fairness into the algorithm. However, we also observe an increase in fairness compared to the benchmark model. The percentage increase in fairness exceeds the percentage decrease in efficiency in most of our experiments. We also perform a sensitivity analysis on our methods and find that the conclusion above is true for varying parameter values.

While our research is mainly focused on the application of crew planning at NS, our methods can be extended to solve other types of crew planning problems. These include railway crew planning in other countries, but also airline crew planning. While the exact definition and quantification of attractiveness and rules on feasibility of duties and rosters may differ from ours, the general framework of our methods allow for such changes to be incorporated quite easily.

In Section 2, we describe the crew planning problem with fairness considerations in more detail. Section 3 provides an overview of the literature on related topics. Section 4 describes the methods and solution approaches we use in our research. We test our methods with some computational experiments. Section 5 contains the data and results of these experiments. Finally, in Section 6, we conclude this thesis.

2 Problem description

In this section we explain the crew planning problem with fairness considerations for railway operators in more detail. We assume we have a number of tasks, which represent a trip from one station to another. These tasks result form the timetable of the railway operator. The tasks have a start time, at which the train leaves, and an end time, at which the train arrives. They also have an origin, the location from which the train departs and a destination, the location where the train arrives. Furthermore, we have a number of attractiveness attributes that apply to these tasks. These attributes determine how favourable a task is for the employees. We assume that we know the tasks and their attributes for the entirety of the planning period beforehand. In practice, the tasks on a particular day may differ from the predetermined schedule due to maintenance work or other disruptions. This problem, which is solved using re-scheduling techniques, is beyond the scope of this thesis.

The goal of our problem is to create rosters. A single roster contains all the tasks an employee needs to carry out during the planning period. We need to create a number of rosters, such that these rosters cover all tasks in the planning period. As a first objective, we want to minimize the number of employees we need to hire to carry out the tasks. As each roster corresponds to one crew member, the objective is to minimize the number of rosters in the solution. Additionally, we want a fair allocation of the attractive and unattractive work. We thus aim to quantify the attractiveness of the rosters and minimize the difference in attractiveness between them. Our problem is therefore a bi-objective optimization problem.

The first part of our problem is the scheduling of the crew, where we aim to combine tasks into duties. In this process every single task must be contained in at least one duty. Allocating a task to multiple duties is inherently inefficient, but might be necessary sometimes to reposition a crew member. In practice, this means that one of the crew members on the task is not actively working, i.e. ‘deadheading’. A duty must adhere to a number of rules that make it a feasible day of work for an employee. We consider the following requirements:

1. A duty starts at a crew base and ends at the same crew base.
2. For any two subsequent tasks in a duty, it holds that the end location of the first task is the same as the start location of the second task.
3. For any two subsequent tasks in a duty, it holds that the end time of the first is at least T minutes before the start time of the second. This allows drivers and guards enough time to transfer to their next train.
4. A duty has a maximum duration of M hours.
5. A duty should contain a lunch break of at least T' minutes. The lunch break should be roughly in the middle of the duty, meaning we also set a maximum duration for a ‘half-duty’. We denote this maximum by H.

The second part requires us to create individual rosters from our duties. Rosters are schedules of duties for each individual employee, for the full planning period. For a roster to be deemed feasible, there are multiple rules that it must satisfy:

1. Each duty in a roster should start and end at the same crew base. This should be the base where the corresponding crew member resides. Note that we do not determine how many crew members we have at each base beforehand. The resulting roster tells us how many crew members we need at each base.
2. The minimum time between each subsequent pair of duties in a roster is R hours.
3. The working time in one week should add up to no more than $\overline{W}$ hours.
4. The average working time per week over the entire period should be no more than $\overline{T}$ hours for every crew member.

In the requirements for the duties and rosters we introduce a number of parameters. The exact values we use for these parameters in our computational experiments are discussed in Section 5. The roster rules, except for the first one, follow from labor agreements made by the unions with NS (Hartog, Huisman, Abbink, & Kroon, 2009). The complete set of labor rules is more extensive and incorporating these is beyond the scope of this thesis.

Attractiveness depends on both task-dependent characteristics and duty-dependent characteristics. In the case of railway crew planning for NS, the Sharing Sweet & Sour attributes include a number of task-dependent characteristics that play a part in the attractiveness. Each task can have one or multiple of the following three Sharing Sweet & Sour attributes:

1. **Type-A work**, which corresponds to performing a task on an intercity train. These trains stop less often on their routes, which is a desirable trait.
2. **Aggression work**, i.e. working on a train that has a higher risk of aggression from passengers towards crew members. Obviously, this is an undesirable attribute.
3. **Double-decker work**, which is also an undesirable trait, due to the fact that guards have to climb many stairs on double-decker trains.

We have one duty-dependent characteristic, which is the duration of a duty. These four attributes fully determine a duty's attractiveness. The attractiveness of a roster is fully dependent on the duties contained in the roster. The rest times between duties, and whether they should be considered desirable or not, are thus not taken into account. In order to quantify the attractiveness, we use the same weights on each attribute as Breugem, Dollevoet, and Huisman (2022). They calculate the unattractiveness score of a duty as follows:

$$0.5 * \textit{Duration} + \%\ \textit{Double-decker work} + \%\ \textit{Aggression work} - \%\ \textit{Type-A work} \tag{1}$$

Here, the duration is in minutes. Each of the three percentages is calculated by summing the total time of the tasks in the duty that possess the considered attribute, and dividing by the total duration of the duty. These weights indicate that each of the three task-dependent attributes is considered equally important. These weights also mean that, for instance, a 20 minute decrease in the duty length has the same weight as a 10% increase in type-A work or a 10% decrease in aggression work.

3 Literature review

The specific topic of our research, namely fairness in railway crew scheduling and rostering, has been researched by several authors. Four of them have also tested their methods on instances of NS. Abbink et al. (2005) introduce the first crew scheduling model that incorporates the Sharing-Sweet-and-Sour rules that were created after the nationwide conflict in 2001. They propose a column generation algorithm and divide the attractive and unattractive duties equally over the crew bases. Breugem, Dollevoet, and Huisman (2022) propose a model for crew rostering where fairness is considered. They analyse the trade-off between the attractiveness of the complete set of rosters and the fairness of the distribution over the employees. Hartog et al. (2009) also aim to solve this problems and account for preferences of employees by minimizing undesirable patterns in the rosters. Furthermore, van Rossum et al. (2022) assume that a capacity planning is given in the form of a template-based roster for each employee. They propose a column generation method, that determines the duties and assigns them to employees, partly integrating the scheduling and rostering step. Their heuristic uses a penalty-based feedback mechanism and a rolling horizon to achieve fairness over time. Jütte, Müller, and Thonemann (2016) consider fairness in a crew scheduling problem for a large European freight carrier, using other attributes that contribute to fairness than the Sharing-Sweet-and-Sour rules. Borndörfer et al. (2015) propose a heuristic method, namely an improvement method on the Lin-Kernighan heuristic, for solving the crew rostering problem. They test their algorithm on applications in public transit, vehicle routing and airline rostering.

The problem in our research most closely resembles that of van Rossum et al. (2022), since we also aim to solve both the scheduling and rostering steps. However, we solve one single problem for a given planning period and thus our problem is not dynamic in nature as theirs is. Also, we combine the tactical and operational phases of the planning process, since we do not assume we have a capacity planning as input. This fact, combined with the individuality and non-cyclicity of the rosters, makes the problem we solve unique amongst similar research.

Railway crew planning in general, without fairness considerations, is a broader topic and has been researched more intensively. Abbink, Huisman, and Kroon (2018) offer a practical introduction to railway crew management. Caprara, Vigo, Fischetti, and Toth (1998) describe a method to solve the crew rostering problem, where the main goal is to minimize the number of crew members needed. This method, unlike many others is not based on a column generation algorithm. Similarly, Bansal, Anoop, and Rangaraj (2024) propose a different solution method for the crew scheduling problem using a heuristic containing bin-packing features. The integration of both steps of the crew planning process has not been researched frequently. Ernst, Jiang, Krishnamoorthy, Nott, and Sier (2001) introduce an optimization model that integrates the scheduling and rostering step, instead of carrying out both steps sequentially. Other variants of railway crew planning includes the research of Gattermann-Itschert, Poreschack, and Thonemann (2022), who include the preference of planners in the creation of duties in the crew scheduling problem. Moreover, Breugem, van Rossum, Dollevoet, and Huisman (2022) investigate the crew re-planning problem, where disturbance cause the need for an alteration of the schedules. They integrate both steps of the planning process in their model, which is an extension on the model of Huisman (2007), who only considers the scheduling phase.

Fairness aspects also exist in other applications of Operations Research. For example, Bertsimas, Farias, and Trichakis (2013) consider fairness in organ allocation and aim to allocate deceased donor organs to patients that need a transplant by means of subjective fairness constraints. Furthermore, Matl, Hartl, and Vidal (2019) look at fairness in vehicle routing. They aim to achieve workload equity by incorporating a number of workload resources into their methods. Finally, Lodi, Olivier, Pesant, et al. (2024) research a resource allocation problem with an application in healthcare, namely the allocation of ambulances. Next to this variety of applications, there have also been attempts to generalize resource allocation problems with fairness aspects. For instance, Bampis, Escoffier, and Mladenovic (2018) introduce a general model for the dynamic fair resource allocation problem.

4 Methodology

We solve the crew planning problem with fairness considerations by sequentially solving the crew scheduling problem and the crew rostering problem. The first is solved with a column generation heuristic, where the master problem is the LP-relaxation of the set-covering problem with an extra constraint. The pricing problem in the crew scheduling algorithm consists of finding new duties with negative reduced costs to add as variables to the restricted master problem. To solve the pricing problem, we construct a graph and employ a dynamic programming approach. The solution of the crew scheduling step, namely the set of duties we find, is the input to our crew rostering algorithm. This algorithm also consists of a column generation heuristic with a set-covering problem. The pricing problem is split into two stages. In the first stage we combine duties into roster-weeks and in the second stage we combine these roster-weeks into full-period rosters. We define resource constrained shortest path problems (RCSPPs) on suitably defined graphs to tackle both stages of the problem. As solution methods we use a labelling algorithm and a constructive heuristic, respectively. Fairness considerations are taken into account in the pricing problem of the crew rostering step.

The details of our methodology can be split up into two parts. Section 4.1 first describes the column generation algorithm we use for crew scheduling in more detail. Section 4.2 describes the algorithm used for rostering purposes.

4.1 Crew scheduling algorithm

The crew scheduling problem can be formulated as a set-covering problem with an extra constraint. We have a set of tasks $i \in I$ and a number of feasible duties $d \in D$. Let the binary parameters a_{id} specify whether duty d contains task i and let the parameters l_d define the length of a duty, measured in minutes. We also have a maximum average duty length, denoted by L. Furthermore, we have that a binary decision variable x_d equals 1 if we include duty d in our solution and 0 otherwise. We get the following mathematical formulation of our problem:

$$\begin{aligned}
\min \quad & \sum_{d \in D} x_d && && (2)\\
\text{s.t.} \quad & \sum_{d \in D} a_{id} x_d \geq 1 && \forall i \in I, && (3)\\
& \sum_{d \in D} l_d x_d \leq L \sum_{d \in D} x_d, && && (4)\\
& x_d \in \mathbb{B} && \forall d \in D. && (5)
\end{aligned}$$

The objective (2) states that we aim to minimize the number of duties we include in the solution. Constraints (3) ensure that each task is covered by at least one selected duty. This shows that this formulation allows that tasks are covered by more than one crew member, as this provides more flexibility in finding a solution. The objective of minimizing the number of duties ensures this will not happen very often. In practice, this results in one of the crew members performing the actual task and the others deadheading. Constraint (4) limits the average duty length to a maximum of L. We also have a constraint on the maximum length of a single duty. This constraint, however, is considered in the pricing problem. While the constraint on the maximum length of a single duty is a requirement, the constraint on the average length is optional. Excluding the constraint may result in a solution in which the durations of the duties are all very close to the maximum. Including the constraint, on the other hand, may increase the variety of durations in the set of duties in the solution. This can make it easier to adhere to the maximum number of hours in a week when generating rosters and can thus improve the solution of the rostering problem. In Section 5 we experiment with including and excluding this constraint. Finally, constraints (5) indicate the variables x_d are binary.

Given that we have the set of all feasible duties in this formulation, the solution would provide us the optimal duties to select for our instance. However, even for a small number of tasks, the number of possible duties is extremely large, as it grows exponentially in the number of tasks. This would result in a computationally intractable problem. To reduce the number of variables, we use a column generation heuristic. Column generation is a technique to solve linear programs with many columns (variables). This technique consists of a Master Problem (MP), a Restricted Master Problem (RMP) and a Pricing Problem (PP). For a general introduction on column generation, see Desaulniers, Desrosiers, and Solomon (2005). The MP is the linear program with the large number of columns we aim to solve. In our case, this is the LP-relaxation of problem (2) - (5), where constraint (5) changes to

$$x_d \geq 0 \qquad\qquad \forall d \in D. \tag{6}$$

The RMP is the MP with only a subset of the duties $D' \subset D$ as variables. The PP is the problem of finding variables with negative reduced costs, which are then added to the RMP. If we denote λ_i for the duals of constraints (3) and μ for the dual of (4), we can compute the reduced cost of a duty with the following formula:

$$RC(x_d) = 1 - \mu(L - l_d) - \sum_{i \in I} a_{id}\lambda_i. \tag{7}$$

We can simplify this formula, by letting $i \in d$ denote that task i is contained in duty d:

$$RC(x_d) = 1 - \mu(L - l_d) - \sum_{i \in d} \lambda_i. \tag{8}$$

We can find duties with negative reduced cost using a shortest-path problem in a directed graph. Let us first define a graph $G(V, E)$. The nodes V in this graph correspond to tasks, except for an artificial source node s and a sink node t. Every task i has four attributes, namely a starting time s_i, a finish time f_i, a start location/origin o_i and an end location/destination d_i. Our set of nodes $V = \{s, 1, ..., n, t\}$ is naturally ordered by the starting times of the tasks, i.e. we have that $s_j \geq s_i$ if $j > i$. For every pair of tasks, there exists a directed arc from node i to node j if and only if two conditions are satisfied. Namely, the origin of task j is the same as the destination of node i ($o_j = d_i$), and the starting time of task j is at least T minutes after the finish time of task i, where T is a constant number of minutes needed to transfer from one task to another ($s_j \geq f_i + T$). This requirement ensures that no cycles exist in the graph, since obviously $f_i > s_i$ for every task. The source node represents the start of a duty and, similarly, the sink node represents the end of a duty. Thus it holds that an arc (s, i) exists for every $i \neq t$ and an arc (i, t) exists for every $i \neq s$. Figure 1 shows an example of a graph that fits our description, where the transfer time T is 5.

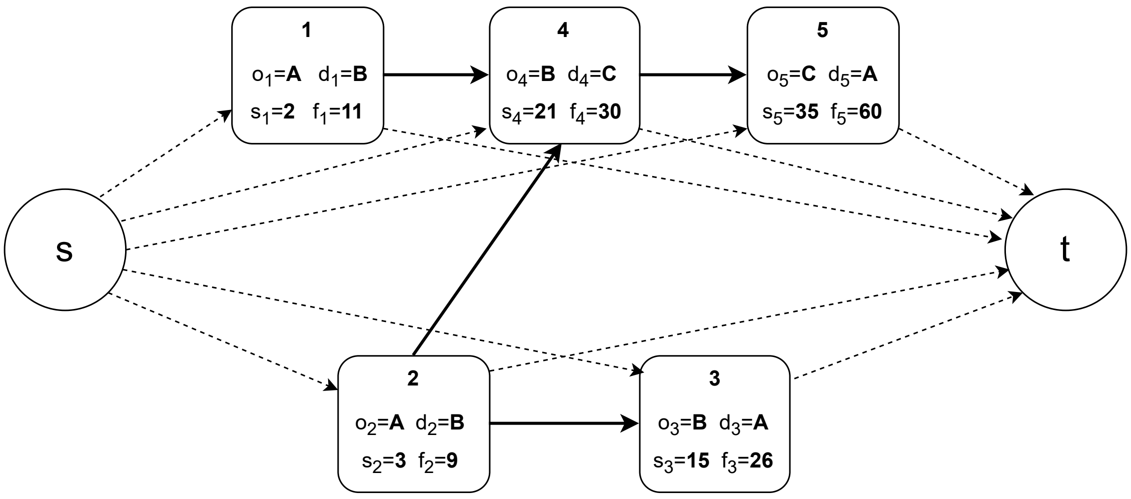

Figure 1: An example graph of tasks, with $T = 5$.

We can break down the reduced cost of a duty along the arcs of its associated path in the graph, such that the length of the path equals the reduced cost. To this purpose, we define the weights on the arcs w_{ij} as follows:

$$w_{ij} = \begin{cases} 0 & \text{if } j = t, \\ 1 - \mu L + \mu(f_j - s_j) - \lambda_j & \text{if } i = s, \\ \mu(f_j - f_i) - \lambda_j & \text{otherwise.} \end{cases} \tag{9}$$

This graph structure is the basis of the method we use to find feasible duties with negative reduced cost. A path from the source to the sink node represents a duty containing the tasks linked to the nodes on the path. Note that there is a possibility that this underlying duty is infeasible, since the duration of this duty might exceed the allowed maximum and there might be no time for a meal break between any pair of tasks.

To account for the restrictions of meal breaks, we first find 'half-duties' using our graph structure. We use a Dynamic Programming (DP) approach in the form of the Floyd-Warshall algorithm to find the shortest path between all pairs of vertices in our graph, except for the source and sink nodes. Let us define the DP variables $d^k(i, j)$ as the shortest path from i to j, using only the first k nodes. We can initialize our DP variables with the weights on the graph.

$$d^0(i,j) = \begin{cases} 0 & \text{if } i = j, \\ w_{ij} & \text{if } (i,j) \in E, \\ \infty & \text{otherwise.} \end{cases} \tag{10}$$

In the recurrence relation we check whether we can decrease the length of the path by adding node k to the path from i to j:

$$d^k(i,j) = min(d^{k-1}(i,j), d^{k-1}(i,k) + d^{k-1}(k,j)). \tag{11}$$

The shortest distance from i to j is then given by $d^n(i,j)$. The paths corresponding to these distances represent half-duties of which the first task is i and the last is j. A half-duty has a maximum duration H. Let us define the final half-duty distances $HD(i,j)$ that account for this constraint.

$$HD(i,j) = \begin{cases} d^n(i,j) & \text{if } f_j - s_i \leq H, \\ \infty & \text{otherwise.} \end{cases} \tag{12}$$

To include the restriction of a meal break, we observe that a meal break can be seen as an extended transfer time. We can use this observation to create a similar graph with different arcs $G'(V, E')$, where tasks are connected if they can be done sequentially with a break in between the tasks. This graph is constructed in the same manner as before, only with a new, longer transfer time T'. Using this graph, let us define the 'break distances' b_{ij}.

$$b_{ij} = \begin{cases} 0 & \text{if } i = j, \\ w_{ij} & \text{if } (i,j) \in E', \\ \infty & \text{otherwise.} \end{cases} \tag{13}$$

Using the break distances b_{ij} and the half-duty distances $HD(i,j)$, we can compute the shortest paths for duties that start with task i and end with task j that have a break immediately after the first task. These values are computed with

$$B(i,j) = min_k(b_{ik} + HD(k,j)). \tag{14}$$

The equation shows that the values for $B(i,j)$ are found by finding the task k directly after the break that results in the minimum distance for this path.

The next and final step in obtaining the shortest paths for every combination of two nodes is finding a full duty containing a break using the variables we have computed so far. We define

$$T(i,j) = min_k(HD(i,k) + B(k,j)) \tag{15}$$

as the total minimal distance for the path from node i to node j. Note that in this equation, we look for the optimal task k to be the last task before the break. To obtain the final reduced cost of the underlying duty, we must still add the weight on the arc from the source to the first task. Thus, $w_{si} + T(i,j)$ is the reduced cost we obtain.

We can reconstruct the path corresponding to $T(i,j)$ by keeping track of the nodes that formed the half-duties in equation (11) and by the nodes that were picked around the break in (14) and (15). Not all the minimum-cost paths are feasible duties with negative reduced cost. We only add the duties corresponding to $T(i,j)$ values to the RMP for which the following conditions hold:

1. The origin of task i is equal to the destination of task j ($o_i = d_j$) and this location is a depot.
2. The duration of the duty is less than or equal to the maximum allowed duration M ($f_j - s_i \leq M$).
3. The negative reduced cost of the duty is negative ($w_{si} + T(i,j) < 0$).

Using the RMP and the PP above, we iteratively solve the LP-relaxation of our main problem with more and more columns. We initialize the RMP with artificial single-task duties that have an artificial, high penalty in the objective, such that they are never optimal to include in the solution. We terminate our column generation procedure once we do not find any negative reduced cost duties in the PP. Then, we use all the columns that we found to solve our original problem (2) - (5) with the binary decision variables. The solution to this problem is our final set of duties.

4.2 Crew rostering algorithm

From the solution of the crew scheduling algorithm, we obtain a number of duties for a relatively small period, such as one single day. These duties are linked to a depot, where the duty begins and ends. We solve the crew rostering algorithm for a longer period of time, and for a single depot. Therefore, we select all the duties from the result of the crew scheduling algorithm that start at one particular depot for multiple consecutive days. This set of duties is the input to our problem and we denote this set by D. With these duties we can make a set of feasible rosters R. We can use a set-covering formulation to model the crew rostering problem, similar to our formulation of the crew scheduling problem in Section 4.1. To this end, we let the binary parameters b_{dr} specify whether roster $r \in R$ contains duty $d \in D$. The binary decision variables y_r equal 1 if we include the roster in our solution and 0 otherwise. We have the following formulation of our problem:

$$\begin{aligned}
\min \quad & \sum_{r \in R} y_r && && (16)\\
\text{s.t.} \quad & \sum_{r \in R} b_{dr} y_r \geq 1 && \forall d \in D, && (17)\\
& y_r \in \mathbb{B} && \forall r \in R. && (18)
\end{aligned}$$

The objective (16) minimizes the number of rosters, and thus the number of crew members we need during the planning period. Constraints (17) ensure that each duty is included in at least one of the selected rosters. In the same way as in the crew scheduling formulation for tasks, we allow that duties are covered by more than one crew member, to provide more flexibility in finding a solution. Again, the objective of minimizing the number of rosters ensures this will not happen very often. In practice, there is no need for deadheading a complete duty. Whilst in the case of tasks this might be necessary to reposition a crew member to the location where they begin their next task, this is not necessary for duties. Duties always start and end at the same location. Therefore, when a duty is contained in multiple rosters, we can delete this duty from all but one roster and still obtain a feasible solution. Finally, constraints (18) show the variables y_r are binary variables. Note that this formulation does not include any fairness considerations. We solely include fairness in the PP of the column generation heuristic.

Given a complete set of all rosters R, solving this problem to optimality would provide us a solution containing the optimal rosters we can make with our previously obtained duties D. However, the number of feasible rosters grows exponentially in the number of duties, resulting in an immense number of rosters even for a small number of duties. Finding all feasible rosters is a difficult task in itself, not to mention that the computation time of the problem would be extremely high. Thus, we rely on a column generation heuristic once more.

We can define a column generation heuristic for this problem in a similar manner as in Section 4.1. We thus consider the LP-relaxation of the problem as our Master Problem, where constraints (18) changes to

$$y_r \geq 0 \qquad \forall r \in R. \tag{19}$$

We initialize this RMP with artificial rosters that have a large penalty in the objective. We create such an artificial rosters for each duty in D, such that the first iteration of the RMP has a feasible solution.

Whilst the RMP closely resembles that of the crew scheduling problem, the PP is considerably different. With the PP we aim to find new rosters to add to the RMP that have negative reduced cost. Denoting ν_d for the duals of constraints (17), we compute the reduced cost of a roster by

$$
\begin{aligned}
RC(y_r) &= 1 - \sum_{d \in D} b_{dr} \nu_d && (20) \\
&= 1 - \sum_{d \in r} \nu_d. && (21)
\end{aligned}
$$

Here, in equation (21), we simplify our notation by letting $d \in r$ denote that duty d is contained in roster r.

We solve the PP in two stages. In the first stage we combine duties into roster-weeks. In the second we combine roster-weeks into full-period rosters. These two stages are necessary, since we have two constraints that are more easily incorporated by splitting the problem. One of them considers the maximum number of working hours in a single week, while the other considers the maximum average number of working hours.

In each iteration we find a number of rosters with negative reduced cost using the PP and re-solve the RMP with the new rosters as added columns. After a number of iterations, we do not find any negative reduced cost rosters with our PP. We then terminate the column generation algorithm. As in the crew scheduling step, we use the rosters that were generated by the column generation algorithm in the set-covering formulation to find an integer solution.

In the following two subsections, we provide a detailed explanation of our solution approach for both stages of the PP.

4.2.1 Combining duties into roster-weeks

To solve the first stage of the PP, we define another directed, acyclic graph $G(V, E)$. The nodes V in this graph correspond to the duties in D that are all in the same week of the planning period, in addition to an artificial source node s and a sink node t. Every node $v \in V$, except for the artificial nodes, has two relevant attributes, namely a start time s_v and a finish time f_v. Note that locations are not relevant in this problem, since we solve the crew rostering algorithm for each depot seperately. The source and sink nodes represent the start and end of a roster-week, respectively. Since every duty can theoretically be the first duty in a roster-week, the source node connects to every other node v with a directed arc (s, v). Similarly, each duty can be the last in a roster-week and thus there exists an arc (v, t) for every v. Two non-artificial nodes u and v are connected if the duties underlying the nodes are compatible, which means the start time of the second duty is at least R time units after the finish time of the first duty, where R is the minimum resting time between two duties. Mathematically, an arc (u, v) exists if and only if $s_v \geq f_u + R$. The starting times of the nodes in V are non-decreasing and $f_v > s_v$ for every node v, and thus no cycles exist in our graph. Figure 2 presents an example of a graph that fits the description above, containing 4 tasks. The minimum resting time R in this example is 12.

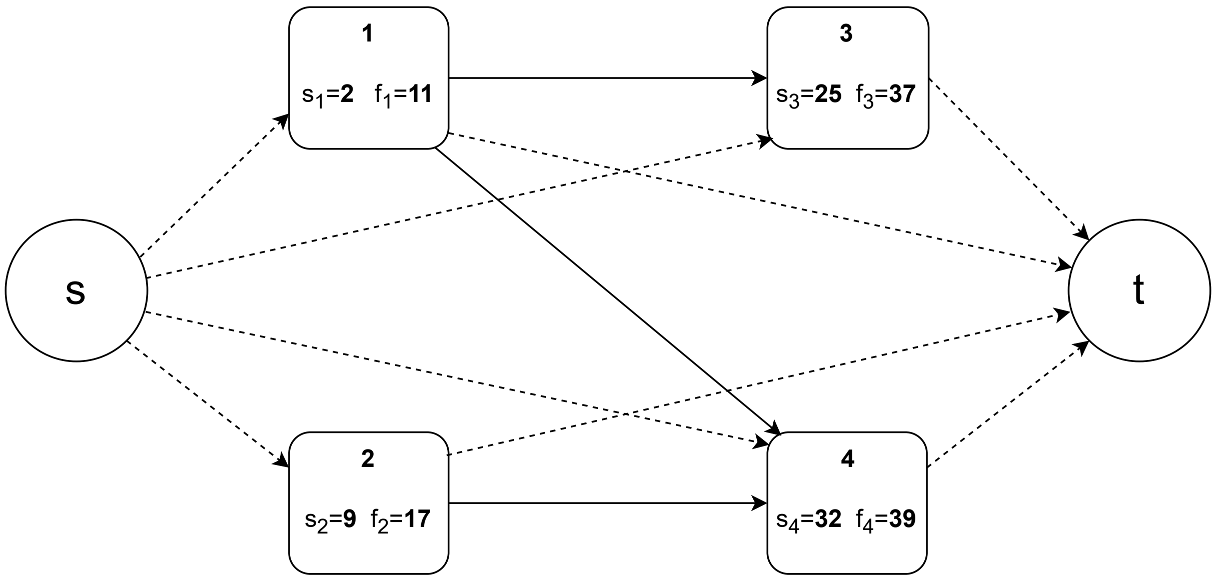

Figure 2: An example graph of duties, with $R = 12$.

Ultimately, we use this graph to find roster-weeks, which we then use to find negative reduced cost rosters. The reduced cost of a full-period roster can be split up into the sum of the reduced costs of the roster-weeks it contains. The reduced cost of these roster-weeks can be further broken down into the contribution of each duty. We can put these contributions on the arcs of our graph as weights/distances d_{uv}. We define these values as follows:

$$d_{uv} = \begin{cases} 0 & \text{if } v = t, \\ -\nu_v & \text{otherwise.} \end{cases} \tag{22}$$

Note that only the summation in (21) is taken into account with these weights. The constant cost of 1 is accounted for in the second stage of the PP. The reduced cost contribution of a roster-week is the length of a path from the source to the sink, given the weights d_{uv} on the arcs.

We model the constraint on the maximum number of working hours in a roster-week as a resource constraint on paths in our graph. This maximum is equal to the available resource $\overline{W}$. The resource consumption on each arc t_{uv} is the duration of the duty underlying the node at the end of the arc. Thus,

$$t_{uv} = \begin{cases} 0 & \text{if } v = t, \\ f_v - s_v & \text{otherwise.} \end{cases} \tag{23}$$

In the graph we have defined, solving a resource constrained shortest path problem (RCSPP) aligns with finding relevant roster-weeks to use in the second stage of the PP. The RCSPP is an extension of the standard shortest path problem where an additional constraint on a resource must be satisfied. In this problem, each arc in the network has an associated weight and a resource consumption. For a more detailed introduction to the RCSPP, see Beasley and Christofides (1989).

Our objective is not to find the single shortest path from the source node to the destination node that respects the resource constraint, as is normally the case for the RCSPP. We, however, want to find multiple short paths. As in Section 4.1, we consider paths between every pair of vertices. We aim to find all the 'dominant' paths between two vertices. The concept of 'dominance' in the RCSPP tackles the challenging combination of finding the shortest possible path, while also accounting for its resource consumption. Let us denote a path, and with that its underlying roster-week, by w, which consists of arcs $e = (u, v)$. A path has a certain resource consumption $t(w) = \sum_{(u,v)\in w} t_{uv}$ and it also has an associated distance $d(w) = \sum_{(u,v)\in w} d_{uv}$. One path w_1 dominates another path w_2 if its distance is shorter, while also having a lower resource consumption. It also dominates the other if either the distance or the resource consumption is lower and the other metric is equal for both paths. We thus have that w_1 dominates w_2 if

$$(t(w_1) < t(w_2) \wedge d(w_1) \leq d(w_2)) \vee (t(w_1) \leq t(w_2) \wedge d(w_1) < d(w_2)). \tag{24}$$

If we have a set of paths W, we refer to a path as dominant, if it is not dominated by any other path in this set. Furthermore, a path is considered feasible, if it does not violate the maximum amount of the available resource T. With this definition, we can formulate the goal of this stage of the PP. We aim to find the dominant, feasible paths in the set of paths that start at a certain node and end at a certain node, for every combination of nodes.

To this end, we use a labelling algorithm. With this algorithm we find all dominant paths from a given non-artificial start node to all other non-artificial nodes in the graph. The weights and resources on the arcs connected to the artificial nodes are included implicitly. For every week in the planning period, we construct the graph as in the description above. In this graph, we consider a non-artificial node v_0 as the start node for our paths. The following algorithm finds the set of feasible, dominant paths to every other node.

Algorithm 1: Labelling algorithm for generating roster-weeks

```
Data: The graph $G(V,E)$, where $V = \{s, 1, ..., n, t\}$ is a set of nodes, ordered by their
      start time $s_d$, and $E$ is a set of directed arcs of which $E'$ are the arcs not
      connected to either of the artificial nodes $s$ and $t$. The distances $d_{uv}$ and
      resource consumptions $t_{uv}$, where $u \in V$, $v \in V$ and $(u,v) \in E$. Also, we have a
      given start node $v_0 \in V$. Finally, an upper bound on the resource $\overline{W}$.
Result: A set of paths $W(v) = \{w(v)_1, ..., w(v)_k\}$ from the start node $v_0$ to every other
        node $v \in V$, the corresponding distances of these paths $D(v) = \{d(v)_1, ...d(v)_k\}$
        and the associated resource consumptions $T(v) = \{t(v)_1, ..., t(v)_k\}$.
1  We initialize $D(v_0) = \{d_{s,v_0}\}$ and $D(v) = \{\infty\}$ for $v \neq v_0$ and $W(v) = \{(s, v_0)\}$,
   $T(v) = \{t_{s,v_0}\}$, $\forall v \in V$.
2  for $u \in \{v_0, ..., n\}$ do
3      for $v$ s.t. $(u,v) \in E$ do
4          for $w(u)_i \in W(u)$ do
5              Create a new path to $v$ denoted by $w^*(v)$ by adding arc $(u,v)$ to $w(u)_i$.
6              The distance of this path is $d^*(v) = d(u)_i + d_{uv}$.
7              The resource consumption of this path is $t^*(v) = t(u)_i + t_{uv}$.
8              if $w^*(v)$ is feasible ($t^*(v) < \overline{W}$) and is dominant within the set $W(v)$ then
9                  Add $w^*(v)$ to $W(v)$.
10                 Add $d^*(v)$ to $D(v)$.
11                 Add $t^*(v)$ to $T(v)$.
12                 Delete all $w(v)_i \in W(v)$ that are dominated by $w^*(v)$.
13                 Delete all corresponding $d(v)_i \in D(v)$.
14                 Delete all corresponding $t(v)_i \in T(v)$.
15 return $W(v)$, $D(v)$ and $T(v)$, $\forall v \in \{v_0, ..., n\}$.
```

Note that the natural ordering of the nodes allows us to only consider potential paths from the starting node to nodes of which the starting time is higher. We use this observation in line 2 of the algorithm. Furthermore, note that the resource consumption and the distance on the arc from the source node to the starting node v_0 is accounted for in the initialization of the algorithm.

We solve this algorithm for every week of the planning period and consider every non-artificial node as the starting node v_0. This results in a number of roster-weeks we use as the input to the second stage of the PP. Note that we have not put any restriction on the reduced cost/length of the path in the algorithm above, nor have we taken fairness into account in any way. This all happens in the second stage.

4.2.2 Combining roster-weeks into full-period rosters

From the first stage described in Section 4.2.1, we obtain a set of roster-weeks V. Every roster-week $v \in V$ has a starting time s_v, which is the starting time of the first duty contained in the roster-week. Likewise, it has a finish time f_v, which is the finish time of the last duty it contains. It also has an amount of working time t_v, which is the sum of the durations of the duties in the roster-week and a distance d_v that follows from the path of the first stage. This distance can be seen as the contribution to the reduced cost of this week. Moreover, it has a week-number n_v that shows to what week of the planning period this roster-week belongs.

Furthermore, it has an attractiveness score. This score is relevant for the fairness considerations in our algorithm. The attractiveness of a roster-week depends on the attractiveness of the duties in it. As mentioned in Section 2, attractiveness of a duty depends on task-dependent characteristics and duty-dependent characteristics. An example of a task-dependent characteristic is whether the task is carried out on an intercity train or on a sprinter. An example of a duty-dependent characteristic is the duration of the duty. In Section 2, we specify the exact characteristics that determine the attractiveness in our computational experiments and how we quantify these characteristics. Our formulation, however, allows for other characteristics to be included as well and we aim to describe a model that is as general as possible.

The scores we define are unattractiveness scores, rather than attractiveness scores, meaning that higher values indicate less attractive duties. We have L task-dependent attributes. Let us denote the attribute score l for task i by $\hat{u}_i^l$. Similarly, we have K duty-dependent attribute scores. Attribute score k for duty d is denoted by $\overline{u}_d^k$. We can calculate the total unattractiveness score of a duty ϕ_d by

$$\phi_d = \sum_{k=1}^{K} \overline{u}_d^k + \sum_{i \in d} \sum_{l=1}^{L} \hat{u}_i^l. \tag{25}$$

The score for a roster-week ϕ_v is the sum of the scores of the duties contained in the roster-week $\sum_{d \in v} \phi_d$.

To solve the second stage of the PP, we define yet another directed, acyclic graph. In this graph $G(V, E)$, the nodes corresponds to the roster-weeks we obtain in the first stage. Paths through this graph represent full-period rosters and thus the source and sink node represent the start and end of the roster, respectively. In our construction of full-period rosters, we require that a roster-week is assigned to each week in the period. In other words, there are no complete weeks off in our final rosters. This is in line with the reality of rostering crew members, since employees expect planned work during the weeks in which they have not requested vacation time. Thus, the source node connects to every roster-week with week-number 1 and every roster-week in the final week of the schedule is connected to the sink node. For an arc to exist between two other nodes u and v, we have two conditions. First, the roster-weeks must be in consecutive weeks. In other words, the week-number of the node v at the end of the arc must be one more than the week-number of the node at the start u ($n_v = n_u + 1$). Secondly, we must remember that we have a minimum rest time between duties. We also need to account for this rest period for subsequent roster-weeks. Thus, our second condition is $s_v \geq f_u + R$. Figure 3 provides an example of a roster-week graph.

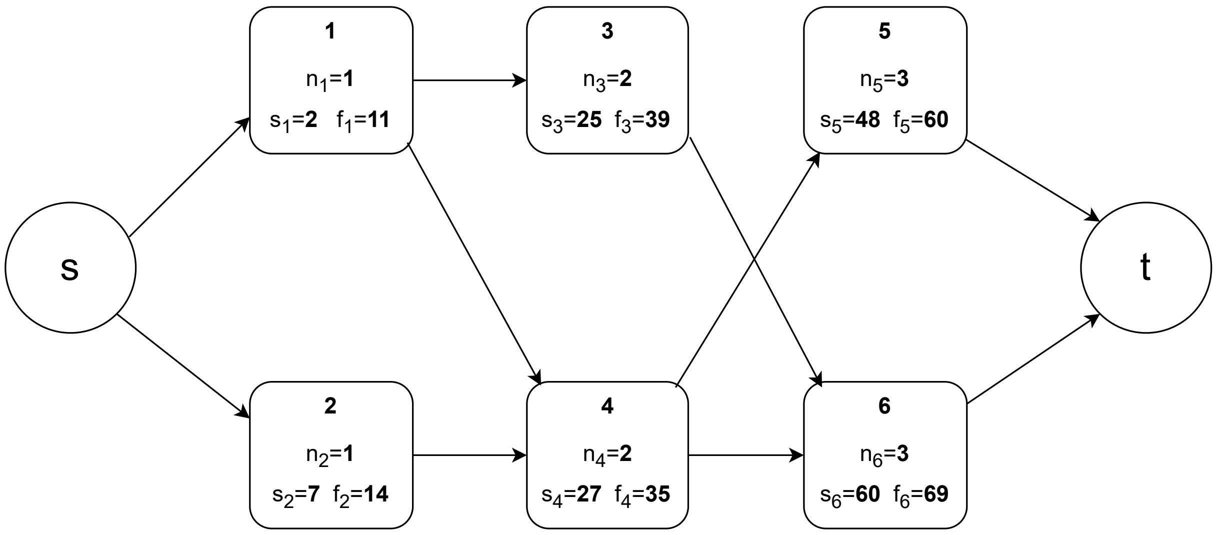

Figure 3: An example graph of roster-weeks, with $R = 12$.

Since we want to model the problem as another RCSPP, we need to define the weights/distances δ_{uv} on the arcs, such that they add up to the reduced cost of the underlying roster in equation (21). We define

$$\delta_{uv} = \begin{cases} 0 & \text{if } v = t, \\ 1 + d_v & \text{if } u = s, \\ d_v & \text{otherwise.} \end{cases} \tag{26}$$

Recall that d_v is the sum of all the negatives of the dual variables corresponding to the duties in the roster-week. This ensures that the length of a path from the source to the sink is precisely the reduced cost of the underlying roster.

Additionally, we define two resources and associated resource consumptions on the arcs. The first of these is similar to the constraint in the first stage. For the full planning period, we have a restriction on the maximum average number of working hours in a week. This number is lower than the maximum working hours in a single week we addressed in the first stage. Multiplying this maximum average $\overline{T}$ by the number of weeks in the planning period, gives us the maximum number of total working hours $\hat{T}$. This number is the maximum available amount of our first resource. The resource consumptions on the arcs of the graph τ_{uv} are determined by the working hours in the roster-weeks:

$$\tau_{uv} = \begin{cases} 0 & \text{if } v = t, \\ t_v & \text{otherwise.} \end{cases} \tag{27}$$

The second resource constraint models the fairness considerations in our problem. Less variety in the attractiveness amongst the final rosters is regarded as a fairer solution. Ideally, the attractiveness of all rosters is exactly equal. However, no employee complains about a roster that is more attractive than the average. Rosters that are significantly less attractive than the average are the main issue. Thus, we want to limit how unattractive a roster can be. The (un)attractiveness of a roster fully depends on the (un)attractiveness of the duties contained in the roster and can be quantified using the task-dependent attribute scores and the duty-dependent attribute scores. We denote our second maximum resource, the maximum unattractiveness score, by $\overline{U}$. We define the related resource consumptions ϕ_{uv} as follows:

$$\phi_{uv} = \begin{cases} 0 & \text{if } v = t, \\ \phi_v & \text{otherwise.} \end{cases} \tag{28}$$

For our benchmark model, in which fairness is not taken into account, we only use the first resource constraint. For the model where we do include fairness, we include both the first and second resource constraints. We can vary $\overline{U}$ to be either more or less strict in the minimum attractiveness of a roster.

On the graph we defined, we aim to solve another RCSPP. Our graph consists of 'layers', one for each week-number in the planning period. A path from the source to the sink must contain exactly one node of every layer. Consequently, the number of non-artificial nodes contained in the path is equal to the number of weeks in the planning period. With this notion in mind, we can formulate a constructive heuristic to solve the RCSPP. A constructive heuristic builds a final solution step-by-step, starting with an empty solution and iteratively adding components to it, creating intermediate, partial solutions. In our case, it starts at the source node and iteratively finds a partial path to the next layer.

The selection procedure of (partial) paths depends on the concept of 'dominance' again. Let us denote a (partial) path, consisting of arcs $e = (u, v)$ by p. For the case where we do not consider the resource consumption on attractiveness, a (partial) path has a certain resource consumption $\tau(p) = \sum_{e \in p} \tau_{uv}$ and it also has an associated distance $\delta(p) = \sum_{e \in w} \delta_{uv}$. Dominance of a path is then defined exactly as in Section 4.2.1. In the other case, we define dominance in a very similar manner. The unattractiveness resource consumption of a path is $\phi(p) = \sum_{e \in p} \phi_{uv}$. A path p_1 dominates a path p_2, if it has a lower score for at least one of $\delta(p)$, $\tau(p)$ or $\phi(p)$ and the rest of the scores are equal. Again, in a set of (partial) paths P a path is dominant if it is not dominated by any other path. A partial path is considered feasible, if it does not violate the relevant maximum allowed resources $\hat{T}$ and, in the case we consider attractiveness, $\overline{U}$.

The idea of our constructive heuristic is as follows. For the first week, we select all partial paths from the source node to a node in the first layer that are dominant among these single-arc partial paths. Then, we consider two-arc partial paths by adding all existing arcs to all single-arc dominant partial paths. Among these two-arc partial paths, we again select only the dominant paths. Note that these partial paths are not necessarily optimal at this stage. In the choice of selecting the first roster-week, we did not consider the consequence that adding a second week to it has. We might have selected a roster-week that is incompatible with some other roster-weeks, due to the resting time constraint. Thus, there is a possibility that we would get a better two-arc partial path if we selected a different roster-week in week one. This illustrates the greedy nature of this heuristic: we select the optimal choice given the information in the current week, which does not necessarily lead to a globally optimal solution. We repeat this process of adding an arc to dominant partial paths for every week in the planning period, until we have complete and feasible paths. Note that (partial) paths that are dominant, but not feasible are not considered in the next iteration of the restricted master problem. For the final paths, we check whether the distance is negative, which indicates a negative reduced cost. The rosters underlying these paths are added to the RMP. The algorithm below provides a more detailed explanation.

Algorithm 2: Constructive heuristic for generating rosters

```
Data: The graph $G(V, E)$, where $V = \{s, 1, ..., n, t\}$ is a set of nodes and $E$ is a set of
      directed arcs. A number of $n$ subsets $V_i \subset V$, with $v \in V_i$ if $n_v = i$. The
      distances $\delta_{uv}$, time resource consumptions $\tau_{uv}$ and unattractiveness resource
      consumptions $\phi_{uv}$, with $u \in V$, $v \in V$ and $(u, v) \in E$. Finally, upper bounds on
      the resources $\hat{T}$ and $\overline{U}$ that determine feasibility.
Result: A set of complete paths $P$ from the source node to the sink node. In this
        algorithm, for ease of notation, a path is denoted by the nodes visited
        sequentially on the path. These paths correspond to negative reduced cost
        rosters.
1  Let $P^i = \{p^i_1, ..., p^i_k\}$ be the set of paths containing $i$ non-artificial nodes. We denote the
     last node of a path $p^i_j$ by $l^i_j$. Furthermore, their corresponding distances, time resource
     consumptions and unattractiveness resource consumptions are $\delta^i_j \in \Delta^i$, $\tau^i_j \in \mathrm{T}^i$ and
     $\phi^i_j \in \Phi^i$.
2  We initialize $P^0 = \{s\}$ and $\Delta^0 = \mathrm{T}^0 = \Phi^0 = \{0\}$.
3  for $i \in \{1, ..., n\}$ do
4      Initialize $P^i = \Delta^i = \mathrm{T}^i = \Phi^i = \emptyset$.
5      for $p^{i-1}_j \in P^{i-1}$ do
6          for $v \in V_i$ s.t. $(l^{i-1}_j, v) \in E$ do
7              Create a new (partial) path denoted by $p^i_*$ by adding node $v$ to $p^{i-1}_j$.
8              The distance of this path is $\delta^i_* = \delta^{i-1}_j + \delta_{uv}$.
9              The time resource consumption of this path is $\tau^i_* = \tau^{i-1}_j + \tau_{uv}$.
10             The unattractiveness resource consumption of this path is $\phi^i_* = \phi^{i-1}_j + \phi_{uv}$.
11             if $p^i_*$ is feasible and dominant in the set $P^i$ then
12                 Add $p^i_*$ to $P^i$.
13                 Add $\delta^i_*$, $\tau^i_*$ and $\phi^i_*$ to respectively $\Delta^i$, $\mathrm{T}^i$ and $\Phi^i$.
14                 Delete all $p^i_j \in P^i$ that are dominated by $p^i_*$.
15                 Delete all corresponding $\delta^i_j$, $\tau^i_j$ and $\phi^i_j$ from $\Delta^i$, $\mathrm{T}^i$ and $\Phi^i$ respectively.
16 Let the final set of paths be $P = \emptyset$.
17 for $p^n_j \in P^n$ do
18     if $\delta^n_j < 0$ then
19         Add $p^n_j$ to $P$.
20 return $P$.
```

Note that this algorithm can be used for the case of one as well as two resource constraints. The only difference is the definition of dominance and feasibility. The rosters corresponding to the paths P are added to the RMP in the next iteration.

5 Computational experiments

In this section, we test the approach presented in Section 4 by means of some computational experiments. In Section 5.1, we present the process of creating our synthetic dataset we use as an instance for our problem and show some of its characteristics. We also set the parameters introduced in Section 2. The intermediate results of the crew scheduling problem are presented in Section 5.2, whereas Section 5.3 presents our final results of the crew rostering algorithm. Section 5.4 contains a sensitivity analysis on some of the parameters we use in our experiments.

5.1 Data

To obtain the set of tasks we need as input to our problem, we create a synthetic dataset based on a part of the actual timetable of NS. We consider a train network containing tracks between six large cities in the Netherlands, namely Amsterdam (Ams), Rotterdam (Rtd), Utrecht (Ut), The Hague (Gvc), Leiden (Lei) and Almere (Alm). Of these cities, the first three are considered crew bases. The figures below illustrate this network.

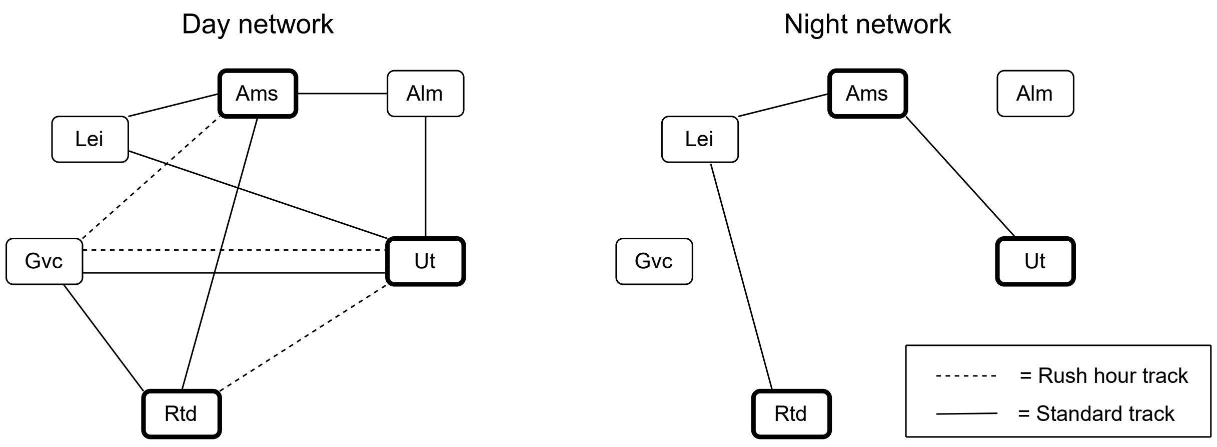

Figure 4: The railway networks in our experiments.

In Figure 4, the network on the left shows the one we consider during the day, i.e. between 5 AM and 1 AM. During the night, from 1 AM to 5 AM, we use the tracks in the sparser network on the right. Dotted lines indicate tracks that are only operated on during rush hours. We construct a set of tasks that represent trips on the tracks of these networks, for both a typical 24-hour working day and a weekend day. On the weekend days, the rush hour tracks are unused. The number of tasks in the working day schedule is 626, whilst a weekend day schedule consists of 422 tasks.

Table 1 and Table 2 present some of the relevant statistics of the tasks for both a working day and a weekend day. For each location, it shows the number of tasks that either start or end there. It also shows what percentage of tasks possess each of the attributes for each location, and in total.

Table 1: Summary statistics of tasks in working day.

	Tasks	Duration	Double-decker	Type-A	Aggression
Total	626	37.2	29.9%	54.6%	42.2%
Ams	311	36.7	38.9%	76.8%	50.2%
Rtd	231	36.5	65.4%	68.8%	49.8%
Ut	243	40	27.2%	42.4%	29.2%
Lei	160	40.7	0.0%	10.0%	39.4%
Gvc	169	36.4	21.3%	57.4%	42.0%
Alm	138	31.3	0.0%	50.7%	37.7%

Table 2: Summary statistics of tasks in weekend day.

	Tasks	Duration	Double-decker	Type-A	Aggression
Total	422	35.5	23.7%	44.5%	66.8%
Ams	219	34	31.5%	68.0%	78.1%
Rtd	141	35.2	48.9%	54.6%	75.9%
Ut	139	40.4	22.3%	28.1%	43.9%
Lei	154	40.6	0.0%	10.4%	66.2%
Gvc	95	31.8	32.6%	32.6%	63.2%
Alm	96	27.7	0.0%	66.7%	65.6%

In these tables, we see that the three depots have the most tasks, making them practical locations to start and end duties. Note that there are no double-decker tasks to and from Leiden and Almere. Double-deckers are operated on a few standard lines in our network, and most of them are connected to Rotterdam. Furthermore, note that Utrecht is the city with the least aggression on the tracks connected to it. When comparing the tasks in a working day with those in a weekend day, we notice a considerable difference in the percentage of tasks that have a high likelihood of passenger aggression. The reason aggression occurs more on weekends is the difference in customers between the workweek and the weekend. Whilst workweek passengers are mostly commuters, weekend passengers include people who may be more aggressive due to increased social activities involving alcohol consumption.

In Section 2, we defined a number of parameters. In our main experiments we use the following parameter values. For the crew scheduling algorithm we set the transfer time in minutes $T = 5$, the minimum break time in minutes $T' = 30$, the maximum duration of a half-duty in hours $H = 5.5$ and the maximum duration of a full duty in hours $M = 10$. For the crew rostering algorithm, we set the minimum rest time between duties $R = 12$, the maximum number of working hours in a single week $\overline{W} = 45$ and the maximum average number of working hours in a week $\overline{T} = 40$. These last three parameters are all measured in hours.

5.2 Crew scheduling

In this section, we look at the intermediate results of the crew scheduling step. We focus mainly on the determination of the optimal value of L, the maximum average duty duration. Secondly, we focus on how the duties are divided over the crew bases. In Table 3 and 4 we present the results for the instance of a working day and a weekend day, respectively. The results in the top rows of these tables correspond to the crew scheduling model without constraint (4), where we do not restrict the average duration of a duty. The other rows show the results for when this constraint is included, for three different values for L. These values correspond to a maximum average duration of 8, 7.5 and 7 hours. In the column next to the values of L, we show the actual value of the average duration of the duties in the solution. The tables also include the number of duties in the best solution we find. The integer programming problem (2) - (5) cannot always be solved to optimality in reasonable time, even with the limited number of duties we get from the column generation algorithm. Therefore, we limit the computation time of the final branch-and-bound algorithm to 10 minutes. In case we do not reach optimality in that time, we use the best feasible solution that was found. In that case, we also report the lower bound (LB) on the number of duties in the optimal solution and the optimality gap as a percentage of the best feasible solution. We also show the average unattractiveness scores of the duties. Finally, we present the number of duties per location and their average unattractiveness scores.

Table 3: Crew scheduling results for a working day.

Total					Amsterdam		Rotterdam		Utrecht	
L	Duration	Duties	LB	Score	Duties	Score	Duties	Score	Duties	Score
∞	510	82	76 (7.3%)	268	37	270	16	277	29	268
480	479	86	78 (9.3%)	252	33	247	22	265	31	252
450	448	79	79 (0.0%)	237	34	234	16	265	29	237
420	420	84	83 (1.2%)	223	38	210	19	239	27	223

Table 4: Crew scheduling results for a weekend day.

Total					Amsterdam		Rotterdam		Utrecht	
L	Duration	Duties	LB	Score	Duties	Score	Duties	Score	Duties	Score
∞	528	51	50 (2.0%)	296	19	300	13	312	19	296
480	479	51	51 (0.0%)	273	17	293	14	288	20	273
450	448	53	53 (0.0%)	255	22	267	13	250	18	255
420	419	58	56 (3.4%)	241	23	247	13	234	20	241

In both tables we can see a similar pattern. As we decrease the value of our parameter L, the lower bounds on the number of duties and the average unattractiveness scores go down as well. A lower value for L means that, on average, we can put less tasks in a duty. This explains the monotonic increase in the lower bound. The decrease in scores is due to the fact that the duration of a duty plays a big part in its attractiveness. We see no clear pattern in the gaps between the lower bounds and the number of duties in the best feasible solution, other than the fact that the gaps are smaller for the weekend day instance. This instance is smaller, which may be a reason that the solver gets closer to the optimum in less time. When we look at the duties per depot, we see that we have significantly less duties that start and end in Rotterdam.

Rotterdam connects to less tracks in the network in Figure 4, which explains this difference. Furthermore, we notice that the average score in weekend days is higher than that in working days and that the average score is the highest for Rotterdam. These observations are a direct cause of the characteristics of the tasks, shown in Table 1 and 2. In these tables, we see that weekends contain more aggression work and tasks connected to Rotterdam are more often on double-decker trains.

The question remains which set of duties we use in the next stage of our solution method, namely the crew rostering phase. We make this decision based on the total unattractiveness of the duties. The total unattractiveness score can be computed by multiplying the number of duties by the average score for a duty. For instance, for the working day instance and $L = 480$, this score is $86 * 252 = 21672$. In both the case of a working day and a weekend day, we find that these scores are minimal for $L = 450$. Thus, these are the sets of duties we use as input to the crew rostering algorithm.

We zoom in on the results for $L = 450$. We look at the unattractiveness scores and the shares of these scores belonging to each attribute in particular. We consider an average week of duties, consisting of five times the average duty of a working day and twice the average duty of a weekend day. In Table 5, we show the average score of such a week of duties and the contribution of each attribute to this score. The table contains these values for the duties of each of the three depots, as well as for the set of all duties.

Table 5: Crew scheduling results for a week of duties.

Depot	Score	Duration	Double-decker	Type-A	Aggression
All	1695	1568	149.4	-243	224.2
Ams	1704	1576	169.4	-262.3	225.4
Rtd	1825	1669	176.9	-298.9	276.3
Ut	1613	1511	99.5	-172.5	182.1

First, we see that the duties starting and ending in Utrecht are, on average, the most attractive, whilst those linked to Rotterdam are the least attractive. One of the reasons for this is that the duties we generate for Rotterdam are significantly longer than those for the other depots. The duration of a duty is an important attractiveness characteristic, as it makes up for most of the score. As for the task-dependent characteristics, we see that Rotterdam has the most aggression work and double-decker work, further increasing its average unattractiveness score. On the other hand, type-A work is also most common in duties for which Rotterdam is the depot. This slightly decreases its average score. In contrast to Rotterdam, the set of duties for Utrecht contains significantly less type-A, aggression and double-decker work. Finally, for Amsterdam, these numbers are all in between those of the other two depots.

5.3 Crew rostering

We now evaluate the performance of our solution method for the crew rostering problem. We test our approach with a number of experiments. The input data for our experiments consists of the working day and weekend day duties that we obtain in the crew scheduling step, namely the duties that we obtain with parameter $L = 450$. These duties contain all tasks we aim to cover, including those scheduled both during the day and at night. For our main experiments, the planning period is five weeks. We can create a set of duties for one single week by concatenating five working day duty sets and adding two weekend days of duties to that set. Five of these weeks stringed together makes up our entire planning period. We solve the crew rostering problem for each depot separately, and thus we divide this set into three distinct sets, one for each depot.

Table 6 shows the results for the model without fairness considerations. This model is the benchmark against which we compare. We show results for each depot separately and for all depots combined. The results include the number of duties in the input dataset, the number of rosters in the solution, the average number of working hours of these rosters and the average, maximum and standard deviation of the unattractiveness scores of these rosters.

Table 6: Crew rostering results without fairness considerations.

Depot	Duties	Rosters	Hours	Avg. Score	Max. Score	Sd. Score
Ams	1070	60	173	5690	6989	970
Rtd	530	29	174	5762	6893	1047
Ut	905	61	158	5236	6651	1372
All	2505	150	168	5562	6989	1145

The results show that we create 150 rosters, which means we would need to hire 150 employees to carry out the 2505 duties in our planning period. Of these 150 employees, 60 start their duties in Amsterdam, 29 in Rotterdam and 61 in Utrecht. Even though we have more duties to cover in Amsterdam than in Utrecht, the number of rosters in the solution is lower. This indicates that our algorithm was more efficient for the Amsterdam instance. This is also reflected in the average number of working hours, where we see that this number is considerably lower for Utrecht. The overall average of 168 hours equates to roughly a 34-hour workweek, which is reasonable. Furthermore, we see that the average unattractiveness score is highest in Rotterdam and lowest in Utrecht, which aligns with the results we see in Section 5.2. The most unattractive roster starts and ends in Amsterdam and has a score of 6989.

This maximum gives us an indication of what values we should use for the parameter that determines the maximum unattractiveness score $\overline{U}$ in the crew rostering method where we consider fairness. In Table 7, we present results for the crew rostering algorithm for different values of $\overline{U}$. We decrease the value of $\overline{U}$ by steps of 500, starting at 6500, which is roughly a 500 decrease in maximum score compared to the benchmark model. We present the aggregated results for the rosters of all depots combined. The detailed results per depot can be found in Appendix A. The table includes the same metrics as Table 6. It also displays the percentage change from the benchmark model for all metrics.

Table 7: Crew rostering results for different values of $\overline{U}$.

$\overline{U}$	Rosters	Hours	Avg. Score	Max. Score	Sd. Score
∞	150	168	5562	6989	1145
6500	152 (+1.3%)	166 (-1.3%)	5489 (-1.3%)	6499 (-7.0%)	1100 (-3.1%)
6000	158 (+5.3%)	159 (-5.1%)	5280 (-5.1%)	5999 (-14.2%)	978 (-14.6%)
5500	169 (+12.7%)	149 (-11.2%)	4937 (-11.2%)	5500 (-21.3%)	875 (-23.6%)
5000	178 (+18.7%)	142 (-15.7%)	4687 (-15.7%)	4999 (-28.5%)	651 (-43.1%)

We see a few patterns emerge in this table. First of all, the numbers of rosters increase as we decrease the value of $\overline{U}$. A lower value of this parameter means a more restrictive problem, resulting in higher objective values (number of rosters). Note that the constraint that $\overline{U}$ sets is restrictive, as the difference between the actual maximum unattractiveness score and the limit we set by $\overline{U}$ is negligible in every instance. The number of rosters and the average score have an inversely proportional relationship, as the total amount of unattractiveness is determined by the duties. This illustrates the trade-off between efficiency and attractiveness in our problem. More attractive rosters, with lower scores, mean that we need to hire more personnel. In the same manner, we also have an inversely proportional relationship between the number of rosters and the number of working hours. For $\overline{U} = 5000$, we see that the average number of working hours is 142, which equates to 28 working hours per week. Lower values of $\overline{U}$ would result in even less working hours, resulting in rosters that are unrealistic. Therefore, we do not consider values below 5000.

The question remains whether the incorporation of $\overline{U}$ results in fairer rosters, and at what cost in efficiency this increase in fairness comes. We can measure the fairness of a solution with the maximum score, which shows how unattractive the least attractive roster is. We can also measure it with the standard deviation of the scores, which indicates how much variability there is in the scores. Table 7 show that both these metrics indicate that our incorporation of fairness does indeed result in fairer rosters, as both the maximum score and the standard deviation drop when we incrementally decrease the value of $\overline{U}$. For every value of $\overline{U}$, the percentage drop in the maximum and the standard deviation of the scores is significantly higher that the percentage increase in the number of rosters. In the cases of $\overline{U} = 6500$ and $\overline{U} = 6000$, the percentage decreases are even more than twice as high as the percentage increase. This indicates that the increase in fairness is greater than the decrease in efficiency, which is a very positive result.

We show two more figures that display the trade-off between efficiency and fairness. In Figure 5, we present a bar chart that shows the number of rosters for every maximum fairness score $\overline{U}$ we consider. The bars also show how many of the rosters belong to each depot. Note that the first bar shows the results for the model without fairness taking into consideration, where our maximum unattractiveness score is 6989.

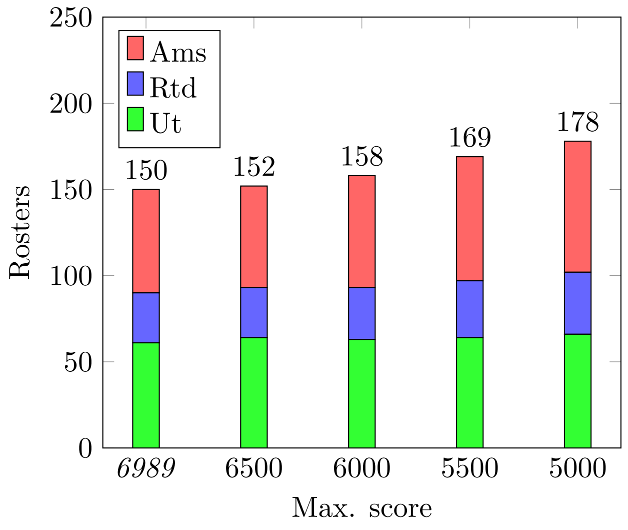

Figure 5: Number of rosters (per depot) for each value of $\overline{U}$.

The figure shows that the proportion of rosters for each depot are roughly equal for different values of $\overline{U}$, although the number of rosters for Amsterdam seems to increase more rapidly than the number of rosters for Utrecht and Rotterdam. Moreover, this figure shows that the increase in rosters is not monotonic in the decrease in $\overline{U}$. While this is the case for the number of rosters for all depots combined, it is not the case for every single depot. For instance, we see that the number of rosters for Utrecht is lower in the third bar than in the second bar. This result may seem illogical, as the model with $\overline{U} = 6000$ is more restrictive than the model with $\overline{U} = 6500$ and the solution for the former model is thus also feasible for the latter. However, our method is a heuristic and thus optimality is not guaranteed. That fact is illustrated in this example.

Figure 6 zooms in on the relationship between the standard deviation and the maximum of the unattractiveness scores. The black line shows this relation for all rosters, while the red, blue and green lines show the relation for the rosters of Amsterdam, Rotterdam and Utrecht respectively.

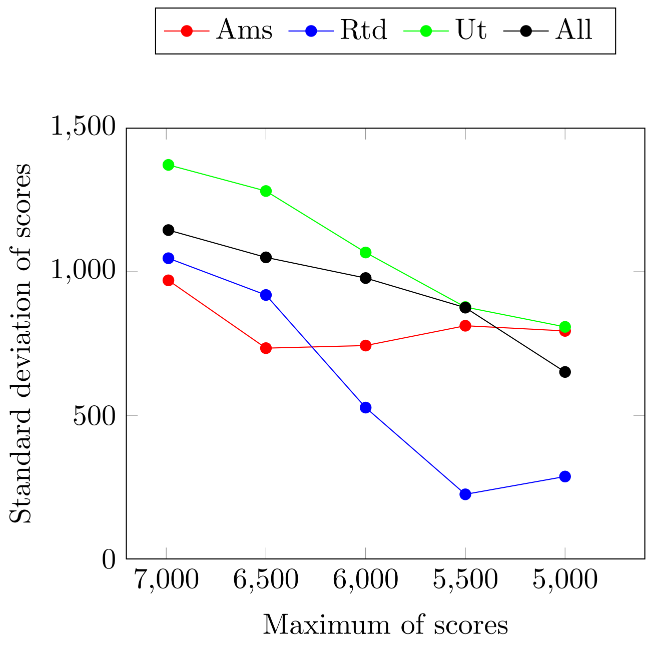

Figure 6: The standard deviation and maximum of the scores for each depot and in total.

The black line shows a monotonic decrease in the standard deviation as the maximum of the unattractiveness score drops. For the rosters per location, however, we see that a decrease in the maximum does not always come with a decrease in the standard deviation, especially for Amsterdam. The positive correlation between these two metrics is most significant for Rotterdam and the least significant for Amsterdam. This might be caused by the fact that the instance for Rotterdam is the smallest and the instance for Amsterdam is the largest, although these results are not enough evidence to prove this hypothesis.

To conclude this section, we dissect the unattractiveness scores in our results. Recall that there are four factors that play into the attractiveness, namely the number of working hours, the amount of double-decker work, the amount of type-A work and the amount of aggression work. In Table 8, we look at these attributes in detail, for our considered values of $\overline{U}$. The table shows the average and the standard deviation of the unattractiveness score and the number of working hours. For each attribute, it shows the average percentage of time spent on tasks with that specific attribute amongst the rosters, as well as the standard deviation of that percentage.

Table 8: Attractiveness attribute scores for different values of $\overline{U}$.

	Score		Hours		Double-decker		Type-A		Aggression	
$\overline{U}$	Avg.	Sd.	Avg.	Sd.	Avg.	Sd.	Avg.	Sd.	Avg.	Sd.
∞	5562	1145	168	35.2	27.1%	7.4%	50.3%	8.9%	51.4%	7.9%
6500	5489	1100	166	33.2	27.1%	7.6%	50.3%	10.2%	51.4%	8.7%
6000	5280	978	159	30.2	27.1%	8.2%	50.3%	9.1%	51.4%	7.8%
5500	4937	875	149	26.8	27.1%	8.1%	50.3%	9.0%	51.4%	10.2%
5000	4687	651	142	19.9	27.1%	7.3%	50.3%	8.4%	51.4%	7.8%

The average score and the average number of working hours follow an inversely proportional relationship with the number of rosters. The average percentage of each task-dependent attribute, however, is equal no matter the value of $\overline{U}$. This is true because the percentage of tasks that have a certain attribute is constant for all solutions and how these tasks are divided over the rosters does not affect the average. The standard deviations of the scores show a decreasing pattern, as we established in Table 7 already. In Table 8, we see that this same pattern holds for the working hours, but not for the task-based attributes. The decrease in the standard deviation of the working hours can be partly explained by the fact that the average number of working hours drops as well. It can also be explained by the fact that lower values of $\overline{U}$ means we select more rosters with a score closer to the limit. Thus, the standard deviation of the overall score drops. And, since the number of working hours contributes a lot to this score, the standard deviation of the number of working hours drops too. The fact that the task-based attribute standard deviations do not show a decreasing pattern do not prove that they do not contribute to the drop in the standard deviation of the score. Tasks with certain attributes might be better combined for lower values of $\overline{U}$ to create more balanced rosters. For example, a duty containing a lot of type-A work might be put in the same roster as a duty that contains a lot of aggression work to balance out the attractiveness. This is not reflected in the standard deviation values in the table.

5.4 Sensitivity analysis

In this section we look at the effect of changing some key parameters in our algorithm. We perform a sensitivity analysis on the number of weeks in the planning period and on the maximum allowed number of working hours in a single week, as well as the maximum allowed average number of working hours.

Table 9 shows results for different lengths of the planning period we consider. In Section 5.3, we only consider a planning period of five weeks. We want to investigate the performance of our model for both a shorter period of three weeks, as well as a longer period of seven weeks. For these three lengths, we consider the model without fairness considerations and the model with fairness considerations with two different values for $\overline{U}$. For the planning period of five weeks, we present the results of $\overline{U} = 6000$ and $\overline{U} = 5000$ again. The decrease in our parameter is thus roughly 1000 per step, which equates to 200 per week. For the lengths of three and seven weeks, we want to have comparable steps. For that reason, we decrease the value of $\overline{U}$ by 600 for the three week planning period and by 1400 for the seven week planning period. The results include the number of rosters in the solution, the average number of working hours and the maximum and standard deviation of the unattractiveness scores. It also shows percentage changes. Note that we do not consider the average attractiveness score, as we do in Table 7, since these results are incomparable for different period lengths and we have established that the relation between this average and the number of rosters is deterministic. Finally, the table displays the average computation time of the algorithm. This is the average over the three depots for which the algorithm is solved.

Table 9: Crew rostering results for different lengths of the planning period.

Weeks	$\overline{U}$	Rosters	Hours	Max. Score	Sd. Score	Comp. time
3	∞	140	108	4100	348	244
	3500	150 (+7.1%)	100 (-7.4%)	3500 (-14.6%)	301 (-13.5%)	276
	2900	172 (+22.9%)	88 (-18.5%)	2900 (-29.3%)	292 (-16.1%)	254
5	∞	150	168	6989	1145	768
	6000	158 (+5.3%)	159 (-5.1%)	5999 (-14.2%)	978 (-14.6%)	788
	5000	178 (+18.7%)	142 (-15.7%)	4999 (-28.5%)	651 (-43.1%)	764
7	∞	166	213	9384	2011	1554
	8000	178 (+7.2%)	198 (-7.0%)	7998 (-14.8%)	1409 (-29.1%)	1588
	6600	190 (+14.5%)	186 (-12.7%)	6595 (-19.1%)	1230 (-38.8%)	1555

The table above shows that the performance of our algorithm that does not consider fairness is worse for longer planning periods. For shorter periods, we have less rosters and thus a more efficient solution. This result may be caused by the heuristic nature of the second part of our solution method for the pricing problem of the crew rostering algorithm. In this constructive heuristic we select the best roster-week for each week of the planning period, not considering the consequences this has for further weeks of the planning period. The probability that this results in sub-optimal rosters is larger for longer planning periods. We also see that the same trade-off between efficiency and fairness is visible for all lengths of the planning period. The number of rosters increases as we decrease the value of $\overline{U}$, but the maximum score and the standard deviation of the score decrease more significantly. Our incorporation of fairness in the algorithm thus works for all planning period lengths. Finally, the results for the computation times show that they do not depend on the value of $\overline{U}$. They do, however, depend on the period length and the increase in the computation time seems to be more than linear in the number of weeks.

Table 10 shows results for different values of the parameter $\overline{W}$, the maximum allowed number of working hours in a single week of a roster. The standard value we consider for this parameter in our main result section is 45 hours. We now consider both a lower value of 40 as well as a higher value of 50. Note that the parameter value for the maximum average number of hours in a roster-week, $\overline{T}$, remains the same as in our main experiments, namely 40. Thus, in the case where $\overline{W} = 40$, the limit on the hours in an average week in a roster is the same as that for a single week. This makes the constraint set by $\overline{T}$ redundant. For each value of $\overline{W}$, we consider the model without fairness considerations, and the model with $\overline{U} = 6000$ and $\overline{U} = 5000$. We thus decide not to change this parameter value based on the maximum unattractiveness score in the model without fairness, as we do in Table 9. In this way, we isolate the effects of changing $\overline{W}$ and ensure we get a fair comparison for different values of this particular parameter.

Table 10: Results for different values of $\overline{W}$.

$\overline{W}$	$\overline{U}$	Rosters	Hours	Max. Score	Sd. Score
40	∞	176	143	6520	1401
	6000	184 (+4.5%)	137 (-4.2%)	5998 (-8.0%)	1232 (-12.1%)
	5000	186 (+5.7%)	135 (-5.6%)	5000 (-23.3%)	1004 (-28.3%)
45	∞	150	168	6989	1145
	6000	158 (+5.3%)	159 (-5.1%)	5999 (-14.2%)	978 (-14.6%)
	5000	178 (+18.7%)	142 (-15.7%)	4999 (-28.5%)	651 (-43.1%)
50	∞	145	173	6755	1077
	6000	155 (+6.9%)	162 (-6.4%)	6000 (-11.2%)	977(-9.3%)
	5000	176 (+21.4%)	143 (-17.3%)	4996 (-26.0%)	743 (-31.0%)

For all values of $\overline{W}$, we see similar patterns. Namely, a decrease in the value of $\overline{U}$ comes with an increase in the number of rosters. At the same time, the maximum and the standard deviation of the scores decrease. In percentages, these decreases exceed the increase in number of rosters. The conclusions we draw are thus the same for each value of $\overline{W}$: the increase in fairness outweighs the decrease in efficiency. When we compare the results for different values of $\overline{W}$ to each other, we note that higher values result in less rosters. Higher values of $\overline{W}$ means we have more freedom in generating roster-weeks, because they can contain more working hours. The results show that we can create better full-period rosters with these roster-weeks and this improves our final solution. We can also see that the variation of scores in the rosters drops as the value of $\overline{W}$ increases. This result may seem counter-intuitive at first. Higher values of $\overline{W}$ provide more freedom to our algorithm, which could result in more variation in the number of working hours. However, the maximum average number of working hours in a week remains 45 for all three instances, which restricts this variation. The main effect of increasing $\overline{W}$ is that we get better solutions due to the increased number of possible roster-weeks. In general, better solutions have a lower standard deviation in the scores, since the scores of the rosters are closer to the maximum score.

To conclude the sensitivity analysis, we investigate the effects of varying the parameter $\overline{T}$, the maximum average number of weekly working hours in a roster. For the experiments in Section 5.3, we use $\overline{T} = 40$, we now additionally consider $\overline{T} = 35$ and $\overline{T} = 45$. The value of $\overline{W}$ is our standard value of 45 in these experiments. This means that for $\overline{T} = 45$, the maximum on the average hours is equal to the maximum in a single week again. We present the same metrics as in Table 10, for the same values of $\overline{U}$.

Table 11: Results for different values of $\overline{T}$.

$\overline{T}$	$\overline{U}$	Rosters	Hours	Max. Score	Sd. Score
35	∞	172	146	6304	1277
	6000	183 (+6.4%)	137 (-6.2%)	5999 (-4.8%)	1222 (-4.3%)
	5000	191 (+11.0%)	132 (-9.6%)	5000 (-20.7%)	1083 (-15.2%)
40	∞	150	168	6989	1145
	6000	158 (+5.3%)	159 (-5.1%)	5999 (-14.2%)	978 (-14.6%)
	5000	178 (+18.7%)	142 (-15.7%)	4999 (-28.5%)	651 (-43.1%)
45	∞	148	170	7608	1432
	6000	158 (+6.8%)	159 (-6.5%)	5999 (-21.1%)	854(-40.4%)
	5000	176 (+18.9%)	143 (-15.9%)	4998 (-34.3%)	779 (-45.6%)

Table 11 shows us that the same patterns we see in the other tables generally hold true for different values of $\overline{T}$. Again, we see an increase in the number of rosters, and a more significant decrease in the maximum and standard deviations of the scores as we use lower values of $\overline{U}$. Only in the second row in the table we see that the percentage drop in the maximum and the standard deviation is smaller than the percentage increase in the number of rosters. This is an exception on the rule we see in all the other results. Furthermore, we see that a lower value of $\overline{T}$ results in more rosters. This result is unsurprising, since the value of $\overline{T}$ determines the maximum number of working hours in a roster and lowering this value thus means we need more roster to cover all the duties. Generally speaking, the standard deviation of the scores is lower for higher values of $\overline{T}$. The case of $\overline{T} = 45$ and $\overline{U} = \infty$ is an exception. In this case, the only restriction we have is that a single roster-week can be no longer than 45 hours. The absence of other restrictions on the duration or attractiveness results in a high maximum score and a high standard deviation.

6 Conclusion

In this thesis, we consider the railway crew planning problem with fairness considerations. This problem consists of two sub-problems. The first of these being the crew scheduling problem, where we generate a number of duties such that these duties cover all tasks. Secondly, the crew rostering problem, where we combine the duties into full-period rosters. Our aim is to generate a set of rosters that is both efficient, meaning we aim to minimize the number of rosters, and fair, meaning we also incorporate the attractiveness of the rosters in our problem. The attractiveness of a roster is determined by the Sharing Sweet & Sour attributes of its duties and tasks. Thus, our problem is a bi-objective optimization problem. We create individual, non-cyclic rosters and do not assume any capacity planning beforehand, combining the tactical and operational stages of the planning problem.

We solve the sub-problems sequentially, with two column generation algorithms. The pricing problem of the crew scheduling algorithm is modelled as as shortest path problem and solved with a dynamic programming approach. This algorithm incorporates the specific requirements for a duty that follow from regulations, such as a maximum duration and a minimum time for a meal break. For the crew rostering problem, we split the pricing problem into two steps. Both problems in these steps are modelled as resource constraint shortest path problems. We first construct roster-weeks using a labelling algorithm, before combining these week into full-period rosters with a constructive heuristic. In this constructive heuristic, there is an option to incorporate fairness by setting a maximum unattractiveness score for the generated rosters. The model that does not include this option is our benchmark model, against which we compare models with varying maximum unattractiveness scores.

We test our model on an instance based on a considerable part of the timetable of NS, spanning a five-week planning period. We measure the fairness of a final solution by means of the maximum unattractiveness score as well as the standard deviation of the scores. In general, we see that the efficiency of a solution decreases if we incorporate fairness into the algorithm. At the same time, we observe an increase in fairness, and this increase exceeds the decrease in efficiency in relative terms for almost all our experiments. We test the robustness of our solution method with a sensitivity analysis on a number of parameters and find that, generally, the same conclusions hold true for a variety of different parameter values. The sensitivity analysis also shows our model has better performance for shorter planning periods.

We conclude that our method successfully integrates the scheduling and rostering step in the railway crew planning problem, whilst accounting for fairness in the solution. The algorithm is able to solve instances of considerable size and thus has potential for railway operators. To be used in practice, our algorithm still needs some refinements. Not all rules and regulations of duties and rosters are incorporated into the pricing problems of the algorithms. The algorithms, however, provide enough flexibility to add new restrictions. The full railway network of the Netherlands is considerably larger than our network. Our algorithm might not be able to handle instances for such a network. One modification that might improve the computational performance is to not consider all roster-weeks in the constructive heuristic in the pricing problem of the crew rostering stage. Instead, one may only consider roster-weeks with a sufficiently low reduced cost contribution.

Other suggestions for further research we give are the incorporation of personal preferences and that of re-scheduling due to disruptions. Since our rosters are individual, all employees can carry out any of the rosters in the solution, assuming that all employees have the same qualifications. We have assumed attractiveness is perceived equally by all employees, which might not be the case. Some employees might find certain attributes less important than others. With individualized attractiveness scores we might be able to assign rosters to employees in an even fairer manner, without hurting the efficiency of the solution. Finally, we have assumed that we know all tasks for the entire planning period beforehand. Of course, this is not the case in real-life. Disruptions cause the need for modifications to our rosters. We encourage other researchers to find a way to alter our solution method to account for this fact.

References

Abbink, E., Fischetti, M., Kroon, L., Timmer, G., & Vromans, M. (2005). Reinventing crew scheduling at Netherlands Railways. *Interfaces*, *35*(5), 393–401. doi: 10.1287/inte.1050.0158

Abbink, E., Huisman, D., & Kroon, L. (2018). Railway crew management. In R. Borndörfer, T. Klug, L. Lamorgese, C. Mannino, M. Reuther, & T. Schlechte (Eds.), *Handbook of optimization in the railway industry* (pp. 243–264). Cham: Springer International Publishing. doi: 10.1007/978-3-319-72153-8_11

Bampis, E., Escoffier, B., & Mladenovic, S. (2018). Fair resource allocation over time. In *Proceedings of the 17th international conference on autonomous agents and multiagent systems* (p. 766–773). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Bansal, A., Anoop, K., & Rangaraj, N. (2024, 01). Heuristic for railway crew scheduling with connectivity of schedules. *Transportation Research Record: Journal of the Transportation Research Board*. doi: 10.1177/03611981231223190

Beasley, J. E., & Christofides, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks*, *19*(4), 379-394. doi: https://doi.org/10.1002/net.3230190402

Bertsimas, D., Farias, V. F., & Trichakis, N. (2013). Fairness, efficiency, and flexibility in organ allocation for kidney transplantation. *Operations Research*, *61*(1), 73-87. doi: 10.1287/opre.1120.1138

Borndörfer, R., Reuther, M., Schlechte, T., Schulz, C., Swarat, E., & Weider", S. (2015, 07). Duty rostering in public transport - facing preferences, fairness, and fatigue [Technical report].

Breugem, T., Dollevoet, T., & Huisman, D. (2022). Is equality always desirable? Analyzing the trade-off between fairness and attractiveness in crew rostering. *Management Science*, *68*(4), 2619–2641. doi: 10.1287/mnsc.2021.4005

Breugem, T., van Rossum, B., Dollevoet, T., & Huisman, D. (2022). A column generation approach for the integrated crew re-planning problem. *Omega*, *107*, 102555. doi: https://doi.org/10.1016/j.omega.2021.102555

Caprara, A., Vigo, D., Fischetti, M., & Toth, P. (1998, 11). Modeling and solving the crew rostering problem. *Operations Research*, *46*. doi: 10.1287/opre.46.6.820

Desaulniers, G., Desrosiers, J., & Solomon, M. M. (2005). *Column generation*. Springer.

Ernst, A., Jiang, H., Krishnamoorthy, M., Nott, H., & Sier, D. (2001, 11). An integrated optimization model for train crew management. *Annals OR*, *108*, 211-224. doi: 10.1023/A:1016019314196

Gattermann-Itschert, T., Poreschack, L., & Thonemann, U. (2022, 03). Using machine learning to include planners' preferences in crew scheduling optimization. *Transportation Science*, *57*(3), 796-812.

Hartog, A., Huisman, D., Abbink, E., & Kroon, L. (2009, 06). Decision support for crew rostering at NS. *Public Transport*, *1*, 121-133. doi: 10.1007/s12469-009-0009-6

Huisman, D. (2007). A column generation approach for the rail crew re-scheduling problem. *European Journal of Operational Research*, *180*(1), 163-173. doi: https://doi.org/10.1016/

j.ejor.2006.04.026

Jütte, S., Müller, D., & Thonemann, U. W. (2016, Sep). Optimizing railway crew schedules with fairness preferences. *Journal of Scheduling*, *20*(1), 43–55. doi: 10.1007/s10951-016-0499-4

Lodi, A., Olivier, P., Pesant, G., et al. (2024). Fairness over time in dynamic resource allocation with an application in healthcare. *Math. Program*, *203*.

Matl, P., Hartl, R., & Vidal, T. (2019). Workload equity in vehicle routing: The impact of alternative workload resources. *Computers Operations Research*, *110*, 116-129. doi: https://doi.org/10.1016/j.cor.2019.05.016

van Rossum, B., Dollevoet, T., & Huisman, D. (2022). *Dynamic railway crew planning with fairness over time* (WorkingPaper). (Railway optimisation, Integrated crew planning, Fairness over time, Column generation, Rolling horizon)

A Detailed results crew rostering

Table 12: Crew rostering results for $\overline{U} = 6500$.

Depot	Duties	Rosters	Hours	Avg. Score	Max. Score	Sd. Score
Ams	1070	59	180	5858	6495	734
Rot	530	29	174	5724	6499	919
Utr	905	64	145	4804	6404	1281
All	2505	152	166	5462	6499	1050

Table 13: Crew rostering results for $\overline{U} = 6000$.

Depot	Duties	Rosters	Hours	Avg. Score	Max. Score	Sd. Score
Ams	1070	65	165	5288	5999	743
Rot	530	30	171	5608	5995	527
Utr	905	63	150	5057	5988	1067
All	2505	158	162	5318	5999	978

Table 14: Crew rostering results for $\overline{U} = 5500$.

Depot	Duties	Rosters	Hours	Avg. Score	Max. Score	Sd. Score
Ams	1070	72	152	4959	5499	812
Rot	530	33	162	5210	5499	225
Utr	905	64	145	4744	5500	877
All	2505	169	153	4971	5500	875

Table 15: Crew rostering results for $\overline{U} = 5000$.

Depot	Duties	Rosters	Hours	Avg. Score	Max. Score	Sd. Score
Ams	1070	76	139	4674	4999	794
Rot	530	36	145	4764	4997	287
Utr	905	66	142	4565	4996	808
All	2505	178	142	4668	4999	651