RAILWAY STAFF ROSTERING

Albena Strupchanska¹, Martin Pěnička², Dines Bjørner³

¹Linguistic Modelling Department, Bulgarian Academy of Sciences
Address: 25A Acad. G. Bonchev Str. Sofia 1113, Bulgaria
Phone: (+359 2) 9796607, Fax: (+359 2) 707273, e–mail: albena@lml.bas.bg

²Czech Technical University, Faculty of Transportation Sciences, Department of Applied Mathematics
Address: Na Florenci 25, 110 00 Praha 1, Czech Republic
Phone: (+420) 224 890 712, Fax: (+420) 224 890 702, e–mail: penicka@fd.cvut.cz

³Technical University of Denmark, Computer Science and Engineering, Informatics and Mathematical Modelling
Address: Building 322, Richard Petersens Plads, DK–2800 Kgs.Lyngby, Denmark
Phone: (+45) 4525 3720, Fax: (+45) 4593 0074, e–mail: db@imm.dtu.dk

Abstract: The problem to be tackled is as follows: There is a railway net. Trains travel from station to station according to a schedule. There are depots (some, not all, stations) in a railway net to which sets of staff members are associated. Staff members are responsible for performing sets of actions in order to fulfill schedule demands. Given a schedule, a staff type, a set of depots and rules (related, for example to the assignment of staff to trains), the problem is to construct work schedules for staff members such that they conform to the rules and to schedule demands. This problem is approached by dividing it into two subproblems. (i) Staff scheduling: from a given schedule, staff type, depots, and some rules, to produce duties (sequence of actions) for staff members; and (ii) staff rostering: generation of base rosters from the duties — constructed in the previous stage — and assignment of particular staff members to them. Base rosters are cyclic sequences of duties for some planning period such that they conform to rules and cover the duties. The assignment of staff members to base rosters is done such that each staff member receives a roster according to that staff member’s personal characteristics (abilities, previous duties etc.). We relate this model to the descriptions provided in (Caprara et al., 1997; Caprara et al., 1999; Caprara et al., 2001; Ernst et al.; Kroon et al., 2000; Lentink et al., 2002).

Keywords: staff rostering, duties, rosters, staff.

1. INTRODUCTION

Staff planning is a typical problem arising in the management of large transport companies, including railway companies. It is concerned with building the work schedules (duties and rosters) for staff members needed to cover a planned timetable. Each work schedule is constructed for each staff type (engine men, conductors, catering staff, etc.).

There are two types of staff planning: long–term planning and short–term planning. We focus here on long term planning. Normally the long term planning task is separated into two stages: (1) staff scheduling and (2) staff rostering. (1) Staff scheduling yields short–term working schedules, called duties, for staff members, such that they satisfy schedule demands. After this stage it is easy to determine the overall required number of staff members, such that the working schedules can be performed. (2) Staff rostering arranges duties into long-term working schedules, called base rosters, and assigns specific staff members to them, such that each staff member performs a roster. During the stage of staff rostering we assume that we have enough staff members such that we can assign rosters to them.

In this paper we will explain and analyze the problem, first informally and then formally. Using a formal specification approach and the RAISE Specification Language, RSL (George et al., 1992), we will present a formal model of the domain.
of staff rostering.

1.1 The Major Functions

Given a schedule, a staff type, a depot, and rules, the task is to produce a set of rosters. What we understand by schedule, depot, and rules will be defined further into the paper.

\[ \text{gen\textunderscore} \text{ross}: \text{SCH} \times \text{StfTp} \times \text{Dep} \times \text{eRS} \rightarrow \text{Ros} \]

\text{gen\textunderscore} \text{ross} expresses all rosters for a given staff type and depot. Usually rosters are generated per depot and we assume that after the staff scheduling stage, all duties generated per depot are moved to the depot. If this is not the case we propose a function that integrates the two stages in staff rostering into one. So given a schedule, a staff type, set of depots, and rules, we produce all rosters per each depot for this staff type.

\[ \text{obtain\textunderscore} \text{ross}: \text{SCH} \times \text{StfTp} \times \text{Dep\textunderscore} \text{set} \times \text{eRS} \rightarrow \text{Ros\textunderscore} \text{set} \]

1.2 Requirements and Software Design

We formally characterize schedules, duties, and rosters in such a way as to meet staff rostering demands. On the basis of such formal domain characterizations we can then express software requirements.

The actual software design relies on the identification of suitable operations research techniques, such which can provide reasonably optimal solutions at reasonable computing times. It is not the aim of this paper to show such operations research algorithms. Instead we formalize the domain of railway staff rostering such that we can later apply it to operations research techniques discovered in further research work.

1.3 Paper Structure

The paper consists of five sections. Each section consists of an informal description of the problem (ie., the narrative) and its formalization.

Section 2 introduces the topology of railway nets from the perspective of staff management. Section 3 introduces the notions of a staff member and of related characteristics — such that are taken into account in the early stage of planning. The last three sections gradually show the creation of rosters from a schedule, a set of depots and rules. Section 4 is concerned with separating journeys observed from a schedule into trips. Thus the notions of journey and trip are introduced. Section 5 introduces the notion of a duty and specifies the concept of the set of duties per each depot. Finally Section 6 introduces more characteristics of staff members and the notion of rosters. The section specifies the concept of staff rosters.

2. NETS, STATIONS AND DEPOTS

We introduce the notions of nets, stations and depots. These are related to the topology of the railway net. They are specified from the staff manager point of view.

2.1 Narrative

We take as a base concept for the railway net, the topology of that net. From a railway net (Net) we can observe stations (Sta) and depots (Dep). Depots are staff bases, i.e., places where staff members “live”. The notion of staff member will be introduced in more details in subsequent sections. From a station we can observe the set of depots to which the station is said to belong. From a depot we can observe a set of stations from which it is easy (for staff) to reach the depot. Given a depot and a station we can observe the distance in time (TInt) between them. We will be interested in stations and depots which are ‘close’ to each other.

Some constraints are: There are at least two stations in a net (\(\alpha_1\)). There is at least one depot in a net (\(\alpha_2\)). The set of depots observed from a station consists of depots of the same railway net (\(\alpha_3\)). The set of stations observed from a depot consists of stations of the same railway net (\(\alpha_4\)).

2.2 Formal Model

We first state some abstract types, i.e. sorts, and some observer functions.

\[ \text{scheme NETWORK} = \]
\[ \text{class} \]
\[ \text{type} \text{Net, Sta, Dep, TInt, StaNm, DepNm} \]
\[ \text{value} \]
\[ \text{obs\textunderscore} \text{Stas}: \text{Net} \rightarrow \text{Sta\textunderscore} \text{set}, \]
\[ \text{obs\textunderscore} \text{StaNm}: \text{Sta} \rightarrow \text{StaNm}, \]
\[ \text{obs\textunderscore} \text{Dep}: \text{Net} \rightarrow \text{Dep\textunderscore} \text{set}, \]
\[ \text{obs\textunderscore} \text{DepNm}: \text{Dep} \rightarrow \text{DepNm}, \]
\[ \text{obs\textunderscore} \text{StaDep}: \text{Sta} \rightarrow \text{Dep\textunderscore} \text{set}, \]
\[ \text{obs\textunderscore} \text{DepSta}: \text{Dep} \rightarrow \text{Sta\textunderscore} \text{set}, \]
\[ \text{obs\textunderscore} \text{StDepDistance}: \text{Sta} \times \text{Dep} \rightarrow \text{TInt} \]
\[ \text{axiom} \]
\[ \forall n: \text{Net} \bullet \]
\[ \text{card obs\textunderscore} \text{Stas}(n) \geq 2 \land \]
\[ \text{card obs\textunderscore} \text{Dep}(n) \geq 1 \land \]
\[ \forall s: \text{Sta} \bullet s \in \text{obs\textunderscore} \text{Stas}(n) \Rightarrow \]
3. SCHEDULE, JOURNEYS AND TRIPS

We explain the notions of schedule, journeys and trips. They help us to introduce the notion of duties.

3.1 Narrative

Schedule and Exchange Stations: A schedule includes information about all train journeys such that each train journey is uniquely determined by a train number, a date, and a time. A train number is a unique identifier which remains the same from the first to the last station of its journey.

Some stations in the net are special from a staff management perspective because it is possible either to exchange staff members or for a staff member to start or to finish work there. We will call such stations exchange stations. From a station we can observe all the staff types for which this station is an exchange station (obs_ExchgStas). Given a station and a staff type we can check if the station is an exchange station or not for this staff type (is_exchgst). Exchange stations are assumed located “near” the depots of the railway network.

Journeys and Trips: Staff members are responsible for performing some actions in order to fulfill schedule demands. Some actions are related to train journeys. Train journeys can be both actual journeys with passengers or freights, or can be “empty” train journeys. A train journey is a sequence of rides with the same train number. A ride is characterized by a departure station, a departure time, an arrival station,
an arrival time and a, i.e., the train between these two stations. Given a schedule we can extract a set of train journeys (\text{journ} set).

There are some restrictions about maximal working time for a staff member between rests. Taking into account these restrictions, it is natural to divide a journey into indivisible pieces of work for staff members. To this end we introduce the notion of a trip. A trip is a sequence of rides of a train journey such that the first and the last station of a trip are exchange stations and the duration of a trip is less or equal to the maximal allowed un-interrupted working time (maxUnIntWrkHr). Each trip is characterized by a train, a departure time, a departure station, an arrival time, an arrival station, and possibly additional attributes. From a trip we can observe train characteristics, for instance ‘kind of engine’, ‘staff types’ and the number needed (for each staff type) to perform a trip etc.

### 3.2 Formal Model

First we will state some abstract and concrete types and some observer functions.

\[
\text{scheme} \quad \text{SCHEDULE} =
\]

\[
{\text{extend STAFF with}}
\]

\[
\text{class}
\]

\[
\text{type}
\]

\[
\begin{align*}
\text{Date}, \text{Hour}, &\, \text{Trn}, \text{TrnId}, \text{LongDistance}, \\
\text{Urban}, &\, \text{ICE}, \text{TGV}, \text{StfAttr}, \text{NoStf} \\
\text{TrnChar} = &\, \text{LongDistance}|\text{Urban}|\text{ICE}|\text{TGV} \\
\text{DateTime} = &\, \text{Date} \times \text{Hour}, \\
\text{Ride} = &\, \{\text{rd} : \text{Ride} \mid \text{wFr(d)(rl)}\}, \\
\text{Journey} = &\, \text{Ride}^*, \\
\text{Trip} = &\, \text{Ride}^*, \\
\text{TrpAttr} = &\, \text{Overnight} \cup \text{Other}, \\
\text{SCH} = &\, \text{Trnld}_{\text{m}} (\text{DateTime}_{\text{m}} \text{Journey})
\end{align*}
\]

\[
\text{value}
\]

\[
\begin{align*}
\leq &\, \text{DateTime} \times \text{DateTime} \rightarrow \text{Bool}, \\
\leq &\, \text{TInt} \times \text{TInt} \rightarrow \text{Bool}, \\
\leq &\, \text{DateTime} \times \text{DateTime} \rightarrow \text{TInt}, \\
\leq &\, \text{TInt} \times \text{TInt} \rightarrow \text{TInt}, \\
\leq &\, \text{DateTime} \times \text{DateTime} \rightarrow \text{Bool}, \\
\geq &\, \text{TInt} \times \text{TInt} \rightarrow \text{Bool}, \\
\end{align*}
\]

\[
\begin{align*}
\text{consInti} &\, \text{DateTime} \times \text{DateTime} \rightarrow \text{Bool}, \\
\text{obsTrnld} &\, \text{Trnld} \rightarrow \text{TInt}, \\
\text{tmchr} &\, \text{Ride} \rightarrow \text{TrnChar}, \\
\text{stfchr} &\, \text{TrnChar} \rightarrow \text{StfTp}_{m} \text{Nat}, \\
\text{obsExchgStas} &\, \text{Sta} \rightarrow \text{StfTp}_{\text{set}}, \\
\text{techTime} &\, \text{Sta} \times \text{Trn} \times \text{StfTp} \rightarrow \text{TInt}, \\
\maxUnIntWrkHr &\, \text{StfTp} \rightarrow \text{TInt},
\end{align*}
\]

\[
\begin{align*}
\text{wfFr} &\, \text{Ride} \rightarrow \text{Bool}, \\
\text{wfFr}(\text{rd}) &\, \equiv \text{dt}(\text{rd}) < \text{at}(\text{rd}).
\end{align*}
\]

\[
\begin{align*}
\text{wFr} &\, \text{Journey} \rightarrow \text{Bool}, \\
\text{wFr}(\text{j}) &\, \equiv \\
\forall &\, \text{i} : \text{Nat} \left( \text{i} : {\text{i} + 1} \subseteq \text{inds j} \right) \Rightarrow \\
\text{obs}_{\text{TrnId}}(\text{tmj}(\text{j}(\text{i}))) &\, \equiv \\
\text{obs}_{\text{TrnId}}(\text{tmj}(\text{i}+1)) &\, \land \\
\text{at}(\text{j}(\text{i}+1)) &\, \land \\
\text{maxUnIntWrkHr} &\, \text{at}(\text{j}(\text{i})).
\end{align*}
\]

\[
\begin{align*}
\text{journ} &\, \text{set} : \text{SCH} \rightarrow \text{Journey-set} \\
\text{journ} &\, \text{set}(\text{sc}) \equiv \\
\{ &\, \text{j} : \text{Journey} \\
\forall &\, \text{tmj} : \text{TrnId}. \text{tmj}:\text{DateTime} \bullet \\
\text{tmj} &\, \in \text{dom} \text{ sc} \land \\
\text{tmj} &\, \in \text{dom} \text{ sc}(\text{tmj}) \Rightarrow \\
\text{j} &\, = \text{sc}(\text{tmj})[\text{tmj}] \}
\end{align*}
\]

\[
\begin{align*}
\text{journ} &\, \text{set1} : \text{SCH} \rightarrow \text{Journey-set} \\
\text{journ} &\, \text{set1}(\text{sc}) \equiv \\
\cup &\, \{ \text{rng} (\text{tm} : \text{TInt} \text{Journey}) \} \\
\wedge &\, \text{trn} \in \text{rng sc}\}
\end{align*}
\]

Each train journey is divided into trips with subject to a staff type. The following is a function that divides a journey into trips.

\[
\begin{align*}
\text{tripList} &\, \text{List} \times \text{StfTp} \rightarrow \text{List}^* \\
\text{tripList}(\text{j}, \text{stft}) &\, \equiv \text{trpl} \\
\forall &\, \text{i} : \text{Nat} \mid \text{i} \in \text{inds trpl} \Rightarrow \\
\text{wFr}(\text{stft}\text{trip}(\text{trpl}(\text{i}), \text{stft})) &\, \land \\
\text{checkSeparation}(\text{trpl}(\text{i}), \text{stft}),
\end{align*}
\]

A trip is well formed if it consists of consecutive rides, if the first and the last stations of a trip are exchangeable stations, and if the train during the trip has the same characteristics as seen from a staff member perspective.

\[
\begin{align*}
\text{wFr} &\, \text{trip} : \text{Trip} \times \text{StfTp} \rightarrow \text{Bool} \\
\text{wFr} &\, \text{trip}(\text{trp}, \text{stft}) \equiv \\
\text{isExchg} &\, \text{trip}(\text{trip}(\text{trp}), \text{stft}) \land \\
\text{isExchg} &\, \text{trip}(\text{trip}(\text{trp}), \text{stft}) \land \\
\sim &\, \{ \text{possibleExchgInside}(\text{trip}(\text{stft})) \} \land \\
\text{tripInt}(\text{trp}) &\, \rightarrow \text{tripInt}(\text{trp}) \land \\
\leq &\, \text{maxUnIntWrkHr}(\text{stft}) \land \\
\text{sameTrn}(\text{trp}, \text{stft}),
\end{align*}
\]

\[
\begin{align*}
\text{isExchg} &\, \text{Sta} \times \text{StfTp} \rightarrow \text{Bool} \\
\text{isExchg}(\text{s}, \text{stft}) &\, \equiv \\
\text{stft} &\, \in \text{obsExchgStas}(\text{s})
\end{align*}
\]

\[
\begin{align*}
\text{possibleExchgInside} &\, \text{Trip} \times \text{StfTp} \rightarrow \text{Bool} \\
\text{possibleExchgInside}(\text{trip}, \text{stft}) &\, \equiv \\
\forall &\, \text{i} : \text{Nat} \mid \text{i} \in \{1..\text{len trp}−1\} \Rightarrow \\
\text{if} &\, \text{isExchg}(\text{nta}(\text{trip}(\text{i})), \text{stft}) \text{ then} \\
\text{dt}(\text{trp}(\text{i} + 1)) &\, \rightarrow \text{dt}(\text{trp}(\text{i})) \land \\
\geq &\, \text{techTime}(\text{trp}(\text{i}), \text{stft})
\end{align*}
\]

else false end
On the stations, where we separate train journeys, there should be enough time for exchanging staff members or for a staff member to change a train. The time interval between departure and arrival time of a train at this station should be greater or equal to this technical time. Thus technical time is the smallest interval of time for which it is possible to exchange staff members or a staff member to change a train.

Finally given a schedule and a staff type we produce the trip set such that each journey that can be extracted from a schedule is divided into trips.

4. ACTIONS AND DUTIES

4.1 Narrative

Actions: Each staff member performs some actions. Actions could be (serving on a) sequence of trips, undergoing rests, and some human resource activities (training, etc.). Rests could be rests between trips, meal rests, rests away from home depot including sleeping in dormitories (external rest) etc. By human resource activities we mean activities performed by a staff member in order to increase qualifications (seminars, courses, etc.).

Sequences of trips are characterized by start times, end times, and lists of rides. A rest is characterized by a start and an end time, a station name, and also some attributes. We will assume that a rest starts and ends at the same station. Human resource activities have the same characteristics as rests.

Duties: Each staff member is related to a given depot, the home depot, in a railway net. A depot represents starting and ending point of staff work segments. A natural constraint imposes that each staff member must return to the home depot within some period of time. This leads to the introduction of the concept of duty as a list of actions spanning \( L \) consecutive days such that its start and end actions are related to the same depot (see Fig. 1). A duty conforms to some rules and satisfies some requirements like continuity, working hours per duty etc. Each duty is concerned with members...
of the same staff type. From a duty we can observe duty attributes such as: 'duty with external rest', 'overnight duty', 'heavy overnight duty', 'long duty', etc. Also each duty has some characteristics, such as:

- **Start time**: Specified explicitly when the first action of a duty is either a rest or a human resource activity; in case of a trip it is defined as the departure time of its first ride minus the sum of technical departure time and briefing time.

- **End time**: Specified explicitly when the last action of a duty is either rest or human resource activity; in case of a trip it is defined as the arrival time of its last ride plus the sum of technical arrival time and debriefing time.

- **Paid time**: Defined as the elapsed time from start time to end time of a duty,

- **Working time**: Defined as the time interval between the start time and the end time of a duty minus the external rest, if any.

The above–mentioned characteristics are common for every duty. There are other possible characteristics of a duty but they strictly depend on a staff type. For instance taking into account the engine men staff type we could observe:

- **Driving time**: it is defined as the sum of the trip durations plus all rest periods between consecutive trips which are shorter than, say \( M \), minutes eg. 30 minutes,

Duty attributes and characteristics are taken into account in the scheduling process while selecting feasible, efficient and acceptable duties per each depot and in sequencing duties into rosters. This will be dealt with in the next sections.

Given the schedule, staff type, set of depots, and rules we can specify duty sets per each depot.

![Fig. 1 Trips and Duties](image_url)

**4.2 Formal Model**

```
scheme DUTY =
  extend SCHEDULE with
class type
  RestAttr, HRAtrr, DtChar,
  Ac ==
    mk_trip(st:DateTime, tripl:Trip*, et:DateTime)
  | mk_rest(sr:DateTime, rsta:Sta, ratt:RestAttr, er:DateTime)
  | mk_hra(sh:DateTime, hsta:Sta, hatt:HRAtrr, eh:DateTime)
  Duty = Ac*,
```
Each duty shall take into account some depot and some staff type. The following is a function which defines a duty set for a depot. It expresses all possible duties for the depot.

```
dutys(trps, stft, dep, rs) ≡
```

Each duty has to start and to end at the same depot and has to conform to some rules. Rules are related to the sequence of actions in a duty, maximal number of actions with a given characteristics, rest time between actions, overall rest time, overall working time, etc. These rules we will call 'rules at a duty level'. Given a trip set, a staff type, a depot, and rules we can characterise a duty for the depot. The function below expresses a duty such that its fist and its last action starts and respectively finishes at the depot, the depot is a home depot for staff members of the given staff type, and the duty satisfy the rules.

```
duty: Trip-set × StfTp × Dep × RS → Duty
duty(trps, stft, dep, rs) as d
```

/* checks if all the trips in */
/* a duty has the same characteristics from staff point of view */

```
duty(trps, stft, dep, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf: Duty × StfTp × RS → Bool
satf(dt, stft, rs) ≡
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs) ≡
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs) ≡
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```

/* checks if the rules for sequencing */
/* actions in a duty are satisfied */

```
satf(dt, stft, rs)
```
The set of all duties for a depot has to obey to some rules too. The rules, ie., restrictions, can be related to a maximal number of duties with specific characteristics per depot, to maximal number of duties per depot, etc. We will call these for ‘rules on a depot level’.

The function below expresses a subset of a duty set, generated in a previous stage, such that it satisfies the rules on the depot level and there is enough staff at the depot to perform the duty set.

\[
\text{dutys} : \text{Trip} \times \text{StfTp} \times \text{Dep} \times \text{Dep} \times \text{RS} \\
\text{post} \\
\forall \text{ds} \in \text{Duty-set} \cdot \text{ds} \in \text{ds} \land \\
\text{cover} : \text{Duty-set} \times \text{Trip} \rightarrow \text{Boolean},
\]

The union of generated sets of duties per each depot has to conform to some overall rules: The number of duties as a whole with a given characteristics, must, for example, not exceed some defined number etc. Also the generated duties as a whole have to cover all the trips that can be observed from a schedule. Finally given a schedule, a staff type, a set of depots, and rules we can express the set of duties per each depot such that the above-mentioned constraints are satisfied.

The following are some auxiliary functions concerning duties and their characteristics.

\[
\text{dutydep} : \text{Duty} \times \text{StfTp} \rightarrow \text{Dep},
\]

Finally given a trip set, a staff type, a depot set and rules we can generate a set of duties per each depot.
5. ROSTERS AND STAFF MEMBERS

We explain the notion of a roster and how it is related to staff members.

5.1 Narrative

Rosters: During the second stage of staff rostering, the duties generated at a previous stage are ordered in rosters. These are long term working schedules assigned to specific staff members. For each depot, in a depot set, a separate staff rostering problem is solved considering only the corresponding duties. We will introduce two auxiliary notions in order to explain the concept of roster and its stages of “construction”.

A ‘plan roster’ is a sequence of duties generated for anonymous staff members of the same staff type. A ‘base roster’ is a cyclic sequence of a plan roster such that it spans through a planning period determined by a schedule. In other words, a plan roster is that part of the base roster which is repeated several times and a base roster is just a cyclic sequence of duties (see Fig. 2). Each base roster has to satisfy some rules. The rules are about the order of duties in consecutive days, and their attributes. There are, additionally, some constraints concerning number of duties in a base roster with specific attributes. These rules we will call ‘conventional rules at the roster level’.

So given a schedule, a staff type, a depot, and rules we can express base rosters for the given depot. These base rosters have to cover all the duties corresponding to this depot and have to conform to some rules. The rules at this level we will call ‘conventional rules at the overall roster level’.

All the duties in a base roster has to be performed by a specific staff member. A roster is a cyclic sequence of (base roster) duties for a specific staff member such that that staff member can perform them. From a base roster, and a staff type we can express rosters. The number of staff members assigned to the base roster is equal to the length of the plan roster. All staff members perform the base roster but starting on different days.

Staff Members: While allocating duties in a base roster to staff members, specific staff members are now considered. At this stage the specificity of staff members comes into play — as one is interested in their personal information. From a staff member personal information we can observe his/her private information (obs_PriInf), such as date of birth, place of living, address, etc. We can further observe staff qualifications (obs_Qual), special work requirements (obs_SpWrkReq), and the list of previous duties (obs_PrevDuty). Given a base roster and a staff member we can observe the staff roster.

**Fig. 2 Rosters**
5.2 Formal Model

scheme ROSTER =
  extend DUTY with
  class
type
  Info, WrkReq, Qualification
  PIROS = Duty*
  BRos = PIROS × Nat
  RoR = PIROS × StfTp → Bool
  RoRS = RoR-set
  OvR = BRos × SttF → Bool
  OvRS = OvR-set
  eRS = RS | check_od((rs:RoRiset))
  check_ps((rs:OvRS))
  Ros = SpecStfMbr → BRos

value
eRS → RS
  obs_Plnf: PersInfo → Info
  obs_SpWrkReq: PersInfo → WrkReq
  obs_PrevDuty: PersInfo → Duty*
  obs_PostDuty: PersInfo → Duty*
  obs_Qual: PersInfo → Qualification
  obs_PIPer: SCH → Nat

  bros_length: BRos → Nat
  bros_length(bros) ≡
  len (plos, numb) = bros in
  len plos end

end

The following function expresses all possible base rosters for a given duty set (related to a depot).

gendep_bros:
  SCH × SttF × Dep × eRS → BRos-set
gendep_bros(sc, stft, dep, rs) as bros
  post
  ∀ bros BRos •
  bros ∈ bros ⇒
  bros = genbrosdep(sc, stft, dep, rs) ∧
  ∃ bros BRos •
  bros = genbrosdep(sc, stft, dep, rs) ∧
  bros ∉ bros

gendep_bros: SCH × SttF × Dep × eRS → BRos
  genbrosdep(sc, stft, dep, rs) as bros
  post
  let ds = depdutyset (dep, stft) in
  cover_dfs(bros, ds) end ∧
  wf_bros(bros, sc, stft, rs)

cover_dfs: BRos × Duty-set → Bool,
wf_bros: BRos × SCH × SttF × eRS → Bool
  wf_bros(sc, stft, rs) ≡
  let (plos, numb) = bros in
  samequal(plos, stft) ∧
  conform_plros(plros, stft, rs) ∧
  len plos * numb = obs_PIPer(sc) in
  end

samequal: PIROS × SttF → Bool
  samequal(plros, stft) ≡
  ∀ i: Nat • i(i+1) ⊆ inds plos
  ⇒ samequal(plros(i), plros(i+1))

smqual: Duty × Duty → Bool

conform_plros: PIROS × SttF × eRS → Bool
  conform_plros(plros, stft, rs) ≡
  conform_plros(plros, stft, rs) ∧
  let cycros = (plros[len plos]) (hd plos)
  in conform_plros(cycros, stft, rs)
end

conform_plros: PIROS × SttF × eRS → Bool
  conform_plros(plros, stft, rs) ≡
  case rs of
  check_od((rs:RoRiset)) →
  ∀ r.R:RoR • r ∈ rs ⇒
  ⇒ r(plros, stft),
  false

end
end

Sets of base rosters have to conform to some rules, such as for example: Having a maximal percentage of base rosters with particular characteristics etc.

seldep_bros:
  SCH × SttF × Dep × eRS → BRos-set
seldep_bros(sc, stft, dep, rs) as bros
  post
  let bros1 = gendep_bros(sc, stft, dep, rs) in
  bros ⊆ bros1 ∧
  conform_bros_rules(bros, stft, rs)
end

conform_bros_rules:
  BRos-set × SttF × eRS → Bool
  conform_bros_rules(bros, stft, rs) ≡
  ∀ bros:BRos • bros ∈ bros ⇒
  case rs of
  check_od((rs:OvRS)) →
  ∀ ovr:OvR •
  ovr ∈ ovs ⇒
  ⇒ false
  end

Given a base roster, a staff type, and a depot we can express rosters for the specific staff members of the given staff type.

gensmros: BRos × SttF × Dep → Ros
gensmros(bros, stft, dep) as ros
  post
  let sms = dstfmg( dep, stft) in
  ros = assignment(bros, sms) ∧
  card dom ros = bros_length(bros)
end
dstfmg: Dep × SttF → Staff
dstfmg(dep, stft) ≡
  let anstaff = dstfmg (dep, stft) in
  getstaff (anstaff) end

getstaff: AnonStaff → Staff
getstaff(anstaff) as staff
  post
  ∀ asm:AnonStaff •
  asm ∈ rng anstaff ⇒
  ∃! ssm:SpecStfMbr •
  ssm ∈ rng staff ∧
  obs_Name(asm) = obs_Name(ssm)
Given a base roster and staff we can assign specific staff members to the base roster such that we receive a set of rosters. The number of rosters is equal to the length of the base roster. All the rosters are permutations of the base roster. So at this stage of planning we assign specific staff members to duties in the plan roster (ie., the cyclic part of the base roster).

**assignment : BRos×Staff → Ros**

**post**

- `card rng staff > bros|Length(bros)`
- `∀ dt: Duty •
  - duty|in|bros(dt,bros) ⇒
    - `∃ ssm: SpecStfMbr •
      - ssm ∈ dom ros ∧
      - dt=first|bros|duty(ros(ssm)) ∧
      - conform|ssm|ros(ssm),ssm ∧
      - permutation(ros(ssm)),bros)`

**duty|in|bros : Duty×BRos → Bool**

**first|bros|duty : BRos → Duty**

Each roster is assigned to a specific staff member according to qualifications, special work requirements, and previous duties, and such that they are performable by that staff member.

**conform|ssm : BRos×SpecStfMbr → Bool**

**sat|qual : BRos×Qualification → Bool**

**sat|predt : BRos×Duty → Bool**

**sat|wr : BRos×WrkReq → Bool**

Finally we express the rosters for the given depot and staff type. For each base roster generated in the previous stage we generate these rosters.

**gen|ross : SCH×StTp×Dep×eRS → Ros**

**post**

- `let bross=sel|dep|bross(sc, stf, dep, rs) in
  - ∀ bros:BRos • bros ∈ bross
  - ⇒ ros = gen|samros(bros, stf, dep)`

All rosters are generated taking into account staff types. So using the function above, we can generate all rosters per depot for all staff types related to this depot. To generate (all) rosters per depot we will need only the schedule, the depot and the rules.

**dep|rosters : SC×Dep×eRS → (StfTp, Ros)**

**dep|rosters(sc,dep,rs) as stf, ross**

**post**

- `∀ stft:StfTp •
  - stft ∈ dep|stftypes|dep) ⇒
    - let rset=gen|ross(sc, stft, dep, rs)
    - in stf|ross = [stf ↔ rset] end ∧
  - card dep|stftypes|dep)=card dom stf|ross`

**dep|stftypes : Dep → StfTp-set**

**dep|stftypes|dep) ≡

- `{stft|StfTp •
  - ∃ ss: SpecStfMbr •
    - obs|SM|Dep(ssm) = dep}`

Base rosters and respective rosters are generated per depot, under the assumption, that after the staff scheduling stage, all duties generated per depot are shifted to the depot. If this is not the case we can observe all the duties generated in the staff scheduling stage per depot (dep|dutyset): This will aid us in integrating the two stages of staff planning into one. So given a schedule, a staff type, a set of depots, and rules we can now express all rosters per each depot in the depot set for the given staff type.

**obtain|ross : SCH×StTp×Dep-set×eRS → Ros-set**

**obtain|ross(sc, stf, dep, rs) as rosset**

**post**

- `let dtss=sel|dutysys(sc, stf, dep, f(rs)) in
  - ∀ ross: Ros • ross ∈ rosset ⇒
    - `∃ dep: Dep • dep ∈ deps ⇒
      - ross=gen|ross(sc, stf, dep, rs) ∧
      - dep|dutyset(dep, stf) ∈ dtss end ∧
    - card rosset = card deps`

**dep|dutyset : Dep×StTp → Duty-set**

**dep|dutyset|dep, stf) ≡

- `{dt|Duty • dep=duty|dep(dt, stf)`

The rest are some of the possible functions for handling staff members in depots.

**hire|sm : SpecStfMbr×Staff ~ Staff**

**hire|sm|sm, stf) ≡

- `sst ∪ {obs|Name(sm) → sm)
  - pre
    - ∀ sm|′ SpecStfMbr •
      - sm′ ∈ rng stf ⇒
        - obs|Name(sm′) ≠ obs|Name(sm) ∧
      - sm′ ∉ rng stf`

**fire|sm : SpecStfMbr×Staff ~ Staff**

**fire|sm(sm, stf) ≡**
6. CONCLUSION

6.1 Summary

We have analysed the problem of staff rostering. This required a careful analysis of the topology of railway nets, including "what are" stations and depots, and the railway staff related to serving on trains and "located" at depots. Then followed a careful analysis of "what are" schedules, journeys, and trips, including a large number of auxiliary concepts relating schedules, journeys, trips and staff. After that followed a careful analysis of railway staff actions and duties — those "things" for which their roster is to be "built". Since rosters must satisfy many constraints we also found a need to analyse, ie., specify many auxiliary notions. We were then ready to define proper rosters and real staff members. And hence to define the main functions of the problem of staff rostering under many constraints.

6.2 Some Remarks

In this example of applying formal specification cum analysis techniques to understanding the domain and requirements for what is normally considered an operations research problem, our specifications became alarmingly detailed. But careful consideration reveals that in normal optimisation work many properties are not considered — they were here — or are overlooked, or are not even discovered. We do not claim to have discovered all necessary and sufficient properties — but to have made a great stride towards that goal.

References


