A web-based Decision Support System for Crew Rostering in Public Transport

Lin Xie, Natalia Kliewer, Leena Suhl

Paderborn, Germany
Dezember, 2013
1 Introduction

Traditionally, in public bus transport the crew rostering problem is solved after solving the crew scheduling problem. The crew scheduling problem aims at generating anonymous duties while some constraints have to be satisfied, such as maximum length of a duty, maximum driving time without a break in a duty, minimum break time during a duty. The generated duties, as well as some other given activities, such as standbys, are required to be covered by drivers. Driver assignment occurs in the crew rostering problem, which is solved on a monthly or semimonthly basis. Law and labor union rules, as well as personal preferences of drivers, are considered during the assignment. The resulting schedule for each driver or group of drivers is called a roster. The details of the input information about activities and drivers, and the rules and regulations of the crew rostering problem, can be found in Section 2.

Compared with crew scheduling problems in the planning process, crew rostering has received much less attention in academic literature. One reason is that most of the cost benefits can be achieved in crew scheduling for generating duties. However, the generation of duties in crew scheduling does not include any information about drivers (i.e. anonymous duties). The assumption made in solving this problem is that all crews are equal. Therefore, it causes difficulties generating schedules of drivers in the crew rostering problem, since the drivers are individual people with different qualifications and preferences. It is possible that some duties cannot be covered on some days, while some drivers do not get any jobs on some other days. In crew rostering, the minimization of operational costs is still important; moreover, drivers’ preferences are considered as well. Rosters which are generated by considering the desires of drivers bring higher acceptance than rosters that ignore individual wishes (see Hanne et al. (2009)). That means fewer exchanges and less absence in operational days. Therefore, fewer recovery activities are expected, which implies lower operational costs. Besides that, the even distribution of workload is considered an important objective, which induces fewer payments for overtimes. For these reasons, the crew rostering problem becomes very difficult to solve, and more complex when dealing with large real-world problems.

The crew rostering problem considers the planning process for generating feasible and satisfactory rosters for bus drivers in public transport. Feasible rosters can be achieved by holding complex rules and regulations during the assignment. Moreover, such rules and regulations might differ from one bus company to another. Therefore, there is a demand for a flexible decision support system to capture variant constraints and objectives. Additionally, satisfactory rosters are hard to obtain, since the interests of both the management and the drivers are considered in crew rostering. The interests of the management include, for example, the
maximization of assigned duties, while the interests of drivers are concerned with getting desired and fairly distributed jobs. Thus, the crew rostering problem is a multi-objective problem, and the objectives might conflict with each other, such as maximizing the number of assigned duties and minimizing overtimes for drivers. A single optimal solution across all objectives does not exist. Consequently, a compromise solution should be found according to the preferences of the decision maker. It is often difficult to get an *a priori* specification of preferences since it is not clear in the beginning which specific objective values are achievable. It is very difficult to create rosters manually to keep both parties satisfied. A decision support system is required to help planners to navigate to a most preferred solution.

1.1 Decision Support System

The concept of a decision support system (DSS) was introduced in the early 1970s. Different definitions of DSS have been given by many researches. However, there is no single definition that is agreed by everyone. In this paper we adopt the definition that was proposed in Gorry and Morton (1971):

> A DSS is an interactive computer-based system that helps decision makers to utilize data and models to solve unstructured problems.

Due to the complex problem structure and the substantial amount of data in crew rostering, human planners need to put a lot of effort into the generation of rosters. Moreover, the schedules generated by a manual process may have many drawbacks, such as infeasible schedules or the lack of consideration of certain types of information. Therefore, there have been more DSSs appearing in literature or in practice in the last decade to help human planners to get good solutions without much human effort.

For the public transport area, the DISSY system in Emden-Weinert et al. (2000, pp. 16-21) is a decision support and simulation system for the bus and tram driver rostering problem. Solution approaches for sub-problems of the driver rostering problem are proposed. Additionally, different scheduling scenarios are evaluated in simulation to better adopt human resource management to new framework conditions. Moreover, different decision support systems have been developed in practice, such as MOVEO Profahr\(^1\) and IUV.crew\(^2\) in Germany. MOVEO Profahr provides solution for cyclic and non-cyclic driver rostering problems separately using a sequential approach. Drivers’ preferences and a balanced workload for all

---

2. IUV.crew: http://www.ivu.de/produkte-loesungen/ivusuite/disposition/ivucrew.html
drivers are considered during the optimization. IVU.crew provides a version IVU.crew.roster.apd to optimize the crew rostering problem under consideration of drivers’ preferences as well as a balanced workload, while another version, IVU.crew.Mobile, is used to allow drivers to enter their preferences and further information online, as well as access their plans using mobile devices.

In the airline and railway sectors, the Carmen Crew Rostering system is developed and applied to different practical crew rostering problems in eight airlines in Europe and North America, and in the Swedisch State Railways and Deutsche Bahn. More details can be found in Kohl and Karisch (2004, pp. 241-254). According to Bentsson et al. (2007, p. 126), this system is used by all major European airlines, and also used by several operators in North America and Asia. Another DSS in the airline and railway sectors, Harmony CDR, is developed in Freling et al. (2004, pp. 208-211) for crew scheduling and rostering. A state-of-the-art Branch-and-Price algorithm is developed to solve both problems sequentially and within one step. Other applications for airline crew scheduling and rostering including the AD OPT Altitude system in Desrosiers et al. (2000), which includes a three-module optimization package for aircraft routing, crew pairing and crew rostering problems. A DSS for airline crew recovery is developed in Guo (2005). Hartog et al. (2009) concentrates on solving the decomposed cyclic train driver rostering problem in an automatic way.

As documented in literature, there is an apparent lack of such a DSS in public bus transit that can handle variant constraints and objectives, as well as considering the interests of both the management and the drivers.

Traditionally, a DSS is supported by a specified operating system (OS), and has to be installed in a specified location. Due to the rapid development of web technologies, it is possible to deliver computerized decision support into a distributed environment, which supports, for example, remote data access. The users on the client side can use a web browser to access the DSS and to generate data at various geographical locations, using distributed data processing, i.e. independent of specific OS and hardware platforms (e.g. mobile phone, PC, or laptop). The installation of software can be avoided; instead, users without technical expertise can easily access the DSS using a browser and a user-friendly interface. Moreover, the web platform is unified. Therefore, the web-based DSS is considered as an extension of the traditional DSS to a global environment.

Most of the DSSs in literature for the crew rostering problems are traditional DSSs, apart from an application in the Carmen Preferential Bidding system (see Kohl and Karisch 2004, p. 235), which supports a web-based bidding interface for crews. The crews receive
immediate feedback while they bid. Moreover, some web applications for bus driver rostering are provided in practice, especially to support the functions for drivers, such as IVU.crew.Mobile.

By the mid-1990s, the first of a series of papers were published in Bhargava et al. (1995a) and Bhargava et al. (1995b) on DecisionNet, electronic marketplace of web-based DSSs, since they saw the new opportunities created by World Wide Web and Internet technologies for building and deploying decision support systems. More about the historical perspectives and how the web technologies influence the development of a DSS can be found in Bhargava et al. (2007, pp. 1084-1085). Moreover, many prototype applications of web-based DSSs in different areas were reported starting at the beginning of the 20th century, such as hospital management (Kohli et al. (2001)), personalized e-services (Yu (2004)), service contract management (Sundarraj (2004)), risk analysis for e-commerce development (Ngai and Wat (2005)), integrated pest management (Jones et al. (2010)), and assessing regional water quality conditions and management actions (Booth et al. (2011)). More examples of development of software for web-based DSSs can be found in Power and Kaparthi (2002, pp. 296-299) and Bhargava et al. (2007, pp. 1089-1092).

Due to wide applications of web-based DSSs in other sectors and the rapid development of web technologies, we develop in this paper a novel prototype of a web-based DSS for different users in bus companies, for the purpose of reducing the human effort in the planning and (simplified) operative process of the crew rostering problem.

We conclude this section with an outline of the remainder of the paper. Section 2 describes the crew rostering problem in more detail, thereby focusing on typical rules and regulations used in the crew rostering in public transport. Different ways to generate a roster and different approaches for solving this problem are described as well. In Section 3, the architecture and concept of the web-based DSS are described. A new model-solver integration framework is described. We conclude this paper with a summary and an outlook of further research.

2 Problem Description for Crew Rostering

In this section, we first give details about the input information for solving the crew rostering problem. After a definition of cyclic and non-cyclic crew rostering problems, a short description of sequential and integrated approaches follows.

2.1 Input Information
Different kinds of input information for solving the crew rostering problem, including information on duties, other activities, drivers, and rules and regulations, are shown below.

**Activities**

Not only must the duties, which are generated in the crew scheduling problem, be assigned to drivers, but also some other activities, such as standbys, days off, leaves, and training periods. Each generated *duty* has the following properties:

- a start time, when it begins,
- an end time, when it ends,
- the duration, which is equal to end time minus start time,
- a depot, where it begins and ends,
- a vehicle type it belongs to,
- the paid time, which is possibly not equal to the duration – it is calculated depending on the duty type,
- a shift type, which depends on its start and end time; for example, one duty beginning between 3 am and 6 am is an early shift, and
- the calendar date or day of the week that the duty belongs to.

*Days off* consist of a couple of rest days between working days. *Standby* activities are planned to cover the absences of drivers, while *leaves* are vacations. The training periods and leaves are pre-assigned and fixed for each driver and cannot be changed in the optimization. Their distribution is decided by bus companies and drivers.

**Driver information**

The data about drivers includes not only the depot (where a driver begins and ends) and vehicle types (depending on the capacity, speed, or equipment of each vehicle), but also the number of days off, the target working hours for the current planning period, the target number of standby activities, and the required training periods and leaves. All of these depend on the work contracts and the drivers’ current work-accounts. The work account of each driver includes their current overtime and their number of days off from previous periods. It describes the driver’s credit, e.g. a driver with more overtime in previous periods can get more jobs with shorter working hours in order to reduce overtimes gradually. Additionally, the drivers can express their preferences, including their daily desired activities and their possible combination of activities. The daily desires of a driver mean that the driver wishes to get one activity on one day, while the possible combination of activities can be, for instance, similar shifts within a working week, and/or not to get an early shift after a day off.
Rules and regulations

The rules and regulations can be labor rules, some of which are imposed by the bus company, and others are due to the agreement between the bus company and employee unions. Every bus company can also define its own internal constraints that are stricter than those required by law. There are three types of rules that can be considered: horizontal rules, vertical rules, and quality rules. Horizontal rules are rules that depend only on one roster, for example, the maximum consecutive number of working days or days off, while vertical rules combine the information among all rosters, such as the capacity of each work-related activity per date or day of the week. Work-related activities include duties and standbys. Quality rules are horizontal rules – without these rules the generated rosters are still legal - but they affect the quality of the rosters. The details of each type of rules are described in Xie and Suhl (2014).

2.2 Cyclic and Non-cyclic Crew Rostering

There are several ways to generate a roster. A cyclic roster is generated for a group of drivers who have the same qualifications and similar preferences. Such a roster includes several rows of duties from Monday to Sunday. The number of rows is equal to the number of drivers in the group. All drivers within a group use the same roster but begin with different rows. An example is shown in Figure 1, in which different activities, ES (early shift), MS (midday shift), LS (late shift), and F (day off), are assigned to two drivers.

Fig. 1 An example of a cyclic roster.

In cyclic rostering, the number of days in the planning period is equal to the number of drivers multiplied by seven. Each week is assigned to each driver in such a way that each weekly pattern is worked in parallel with a person. Cyclic rostering is a rather simple way to generate rosters, because instead of generating a roster for each driver a roster is generated for a group of drivers. Furthermore, the roster is fair since drivers within a group have the same duties, including unpopular duties, and the days off and weekends off are evenly distributed. However, this fairness is only achieved if absences and other occurrences do not arise in real life, otherwise the cycles are destroyed. Moreover, a cyclic roster considers duties from days of the week, not calendar dates. Therefore, it is not flexible enough to respond to changes in traffic, such as an increase in traffic on holidays. Moreover, a new employee can not be easily


<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MS</td>
<td>MS</td>
<td>F</td>
<td>F</td>
<td>ES</td>
<td>ES</td>
<td>ES</td>
</tr>
<tr>
<td>ES</td>
<td>ES</td>
<td>F</td>
<td>F</td>
<td>LS</td>
<td>LS</td>
<td>LS</td>
</tr>
</tbody>
</table>

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65
added to an existing group, since the planning horizon will be changed, and one more week must be planned.

The shortcomings of cyclic rostering can be avoided if an individual roster is generated for each driver within a given time period (e.g. two months). Such a non-cyclic roster may consider wishes with respect to a special single day off or vacation periods. Noncyclic rostering provides more freedom to take holidays and special events into account. Figure 2 shows an example of a two-week, non-cyclic roster for two drivers, in which the first five days are in the previous period. The horizontal rules between fixed and unfixed activities must be enforced. We assume that the maximum consecutive number of working days is restricted to five. The five fixed activities in the last period for driver d2 are work-related activities. Therefore, a day off activity should be assigned to the first day of the planning period (day 6 in the example). Additionally, we assume the maximum length of a days-off period is three days. A working day is therefore required for d1 on day 6. Note that the previous period is implicitly considered in the cyclic roster, if the roster is not disrupted.

![Fig. 2 An example of a non-cyclic roster.](image)

2.3 Sequential vs. Integrated Approach

The crew rostering problem is usually divided into several sequential sub-problems due to its high complexity (see Moz and Pato (2007) for the nurse rostering problem). A roster is considered as a schedule of combinations of work-related activities (standby, shift/duty) and days off activities. The sub-problems are days off scheduling (the optimal distribution of days off by considering fair distribution of days off), shift assignment (the allocation of shifts to drivers), and duty sequencing (according to the result of shift assignment, duties are assigned to drivers). The problem of integrated days off scheduling and shift assignment is named rota scheduling in Emden-Weinert et al. (2000) and Xie et al. (2012b).

The drawback of the sequential crew rostering problem, for example the approach we use in this paper, with rota scheduling first and duty sequencing second, can be found in our previous published work in Xie and Suhl (2014). It is worth noting that, duty sequencing problem is usually easier to solve compared to the rota scheduling problem, due to the assigned shifts and other activities. Moreover, fewer restrictions are considered in this problem (details see Sodhi and Norris (2004) and Xie and Suhl (2014)).
3 A web-based DSS for crew rostering

In this section, the functionality of our proposed web-based DSS for the crew rostering problem is firstly described. Secondly, we propose model and solver agents within a model-solver integration framework.

3.1 Functionality of the Web-based DSS

There are different classifications for a DSS: for example, at the technical level described in Power (1997), and at the conceptual level shown in Power (2002, pp. 12-14). Marakas (2003, pp. 8-24) defines the typical components of a DSS: the users, the user interface, the data management system (including components for gathering, storing and organizing the data that will be useful for the decision making), the model management system (providing variable models for the analytical capabilities in DSS), and the knowledge engine (which recognizes the problem, generates solutions, and functions related to the management of the solving process).

Some other authors identify different components in a DSS (such as in Sprague and Carlson 1982, Power 2002).

![Structure Scheme](image)

**Fig. 3** Structure scheme of the web-based DSS and the functions of each component.

We propose in this section the structure and functionality of a web-based DSS as illustrated in Figure 3, which shows its structure scheme and the functionality of each component. Our DSS system is divided into two parts: the client side and server side. The client side includes users and a web browser interface, while the server side consists of a web server, DSS server, and database server. Compared with the components of the DSS described in Marakas...
(2003), the user interface is replaced with the web browser interface in our web-based DSS. Additionally, the data management is performed on a database server. Moreover, the DSS server provides the model management functionalities (in “formulation”) and the knowledge engine (in “solution approach”). The components of our DSS and their functionalities are described below.

**The users**

The users of the DSS are the controllers, planners, managers and drivers. The functions of each group of people are described as follows:

- Figure 4a illustrates all of the functions of a planner; here we assume a planner to be an administrator. Therefore, he or she has access to all functions, such as viewing statistics for managers.

![Simply and Quickly generate and optimize duty plan.](image)

**Fig. 4a** The overview page for the planner.

The *Controller* and planner usually belong to the same group of people. However, the controller is responsible for the recovery process for repairing disrupted rosters, while the planner is responsible for the planning process for generating rosters. This paper concentrates on the planning process for the crew rostering problem, therefore only a simple function is provided for the controller in our DSS, i.e. sending SMS messages to drivers to notify them of changes (see Figure 4b).
Fig. 4b The page for recovery on an operational day.

Moreover, the planner can register new drivers (see Figure 4c), who receive information about accessing the website via email.

Fig. 4c The page for registration of a new employee.

Additionally, the planner can modify the information about a driver (see Figure 4d).

After the planner has uploaded the verified data for the generation process (Figure 4e),
he/she can provide the parameters that will help the solver agent to choose a suitable
solver, such as running time and gap to optimum, but this is optional (see Figure 4f).

Fig. 4d The page for managing all employee data.

Fig. 4e The page for uploading the data file for the generation of rosters.

Moreover, the planner can choose to receive a message via email and/or SMS if a so-
lution is available (see Figure 4f).
Fig. 4f The page with the options for the generation process.

Fig. 4g The results page, including the plans for all drivers, and additional result information, such as selected solver.

The detailed solutions are shown including the result details, the selected solver(s), and the resulting plans for all drivers (see Figure 4g). If the solution is not satisfactory, the planner can restart the optimization with different parameter settings. As long as the solution is accepted, the solution analysis is stored in the database.
The manager can access the database to get some analyzed data for supporting decisions, such as evaluating the costs of proposed rule changes or evaluating drivers’ satisfaction (see Figure 5).

Fig. 5 The website for the manager and the corresponding functions.

The driver can provide his/her preferences, including a desired day off or a working shift on a particular day; or a combination of jobs, such as similar jobs within one working week, in the next planning period (see Figure 6a). After the solutions have been accepted by the planner, the drivers can access their plans, and download them to their own calendars as shown in Figure 6b.

Web browser interface
The users on the client side can access the DSS simply by using a web browser. Different users have different website designs based on their available functions and different user permissions (see Figures 4a-4g, 5, and 6a, 6b). The websites were built entirely using HTML5, CSS3 and JavaScript (using the jQuery JavaScript library).

Web server
Web services provide the web-based user interface, written in the C# language using the Microsoft ASP.NET MVC4 (2014) (Active Server Pages.NET Model View Control 4) framework; this enables people to easily access other servers from remote locations.
**Fig. 6a** The page for a driver to enter his/her desires for the next period.

**Fig. 6b** The page for a driver, showing his/her plan in calendar format for the current period.

*Database server*

This component provides for the transferring of data between the web server and DSS server. The input data from planners and drivers are analyzed and gathered in the database server. Those data include activities, driver information, and rules and regulations (see Section 2.1).
Recall that activities include duties, standbys, days off, and others. Driver information includes not only the drivers’ qualification details (for example: vehicle types) and work-accounts, but also their stated preferences. The planner can access the database to modify the data, add new data, and upload modified data. Data that are ready for optimization are gathered into a .csv or .dat file. A specification of the format can be found at the website http://dsor.de/crewrostering. The solutions and analysis provided by the DSS server are stored on the database server. All historical solutions are categorized according to different problem types/sizes, to support the decision making by a model agent and a solver agent, which will be described in the next subsection. Our actual database includes 16 real-world instances with different problem sizes for the CCR and NCCR problems and their solutions. The number of duties of all instances varies between 1,013 and 19,486. The properties of these real-world instances can be found in Xie and Suhl (2014).

To create, configure, and use databases on the web for our web-based DSS, we chose to use the data management services of Windows Azure described in Calder et al. (2011) (Microsoft’s cloud application platform), which uses Microsoft SQL Server technology. Therefore, we only need to focus on our DSS services, since the processing of replication, encryption, patch management, and backups will be taken from the data management services of Windows Azure. The maintenance cost is reduced while manageability and scalability are increased by running SQL Server databases in the cloud.

DSS server

The planner can start the optimization as long as the input data are complete and verified. The optimization process occurs on our DSS server, providing the mathematical formulation, solution approaches and results analysis. The mathematical formulation involves the generation of a model based on the rules and regulations provided by the bus company and drivers’ preferences, and the problem type (such as CCR or NCCR). A model agent is provided to support the model formulation. Details of this can be found in the next subsection. The generated model is in the form acceptable to an optimization, a column generation or a metaheuristic solver. The intelligent selection of a solver is based on a solver agent. Details about the process of model-solver integration are described in next subsection. The evaluation of results is stored on the database server.

3.2 Model-Solver Integration

The architecture of a model-solver integration framework in our optimization process is shown in Figure 7, which is based on the idea described by Lee and Huh (2006) for autonomous and intelligent model solution. The planner firstly defines the problem (such as rules)
and the requirements for the solver (such as the running time). Model and solver agents are intelligent agents, consisting of computer software for supporting the automated selection of a suitable model and solver. The choices of agents are based on the historical data in the database, which will be described later.

Figure 7 Architecture of a model-solver integration framework.

Figure 8 shows a sequence diagram to describe how a planner obtains a solution. First of all, the planner sends a request for a model to the model agent based on the given information, including rules and drivers’ preferences, whilst he or she also sends a request for a solver to the solver agent, with details such as running time or gap to optimum. Based on the planner’s request, the model agent checks the rules and recognizes the problem type (such as CCR or NCCR), which sometimes also depends on the historical data, such as whether the (similar) plan was formulated as sequential or integrated planning. When all preparations are complete, the model is generated. In order to solve this model, the solver agent chooses a solver, based on the data about the model and the planner’s request. The selected solver solves the model, and the results with analyses, are stored as historical data.

The model-solver integration framework is implemented in the C# programming language, making full use of its object-oriented nature. Figure 9 illustrates an overview of the
implementation. The rules checker identifies the rules while the problem-type recognizer

![Diagram](image)

**Fig. 8** Process of getting a required solution for the planner.

![Diagram](image)

**Fig. 9** Abstract implementation of the model-solver integration framework.
identifies the problem type. The link between the problem-type recognizer and network generator corresponds to the generation of a problem-specified network. Constraints corresponding to the rules and to the network are generated by the model generator. As long as the model is generated, the solver type is chosen in the solver-type recognizer. Also, the corresponding solver will be selected to run the model.

3.2.1 Model Agent

The model agent is responsible for checking rules, determining the problem type, and generating the network, which are requirements for the formulation of a model.

Rules Checker

As mentioned before, the working regulations often differ from one bus company to another. The most-used working regulations are described in Xie and Suhl (2014). Moreover, one bus company might enforce the rules more strictly than another. Example 1 is provided to show this case.

Example 1: A company rule states that the maximal block length of work-related days can not exceed \( L_m^w \) for each driver \( m \in M \), where \( M \) is a set of all drivers (or driver groups for CCR), and \( L_m^w \) is defined as maximal length of consecutive work-related activities for employee \( m \). Bus company A wants to hold to this rule, while bus company B allows it to be slightly violated. The mathematical formulation of this rule will be shown later in this subsection.

Problem-Type Recognizer

The possible problem types are listed below. A recognizer is developed to identify the problem types, some of which can be recognized by the given information. Figure 10 shows the input information for different problem types.

- Cyclic crew rostering (CCR) can be recognized by its feature, i.e. restricted capacities of activities based on days of the week.
- Non-cyclic crew rostering (NCCR) can be recognized by its feature, i.e. restricted capacities of activities based on calendar dates.
- Multi-objective crew rostering (MOCR) is provided to identify the desired weight of each objective for the planner. If the weights of objectives are not provided as input, then the multi-objective approach is required.
- Rota scheduling (RS) is identified by having no duties as input.
- Duty sequencing (DS) is identified by the duties and rotas provided; these rotas specify the distributions of assigned shifts and other activities.
• Sequential planning (SP) and Integrated planning (IP) share the same input information.

The choice will be left to the agent solver. Although the results of integrated planning show better solutions compared with the ones solved by sequential planning, a sequential approach might be preferred in some cases, where using the integrated approach is too time-consuming, or is unsolvable due to limited running time or memory. Combinations of different problem types are possible, such as CCR and DS, or NCCR and IP.

![Diagram](Fig. 10 Input information for different problem types.)

Different problem types might reflect the formulations of the same rule differently, i.e. the same rule might be formulated differently if different problem types are identified. An example is shown below.

Example 2: A company rule states that the units of each activity are restricted. Bus company A uses the rosters generated by the NCCR problem, while bus company B uses CCR to generate rosters. Both companies consider the IP problem. The mathematical formulation of this rule will be shown later in this subsection.

**Network Generator**

Once the problem type has been identified, the multicommodity flow network for the corresponding problem is generated, where each network layer represents the valid activities on
each day for a driver, and possible connections between them. Recall that an activity can be a shift, a standby, a day off, or a fixed activity, such as a training period or annual leave. The activity arcs are only connected if they are compatible, i.e. the sequence of both activities does not violate any work regulation, such as the minimum rest period between two consecutive shifts. A path on the network corresponds to a schedule for an employee for the planning horizon (for example: two months). On each network layer, the cost of each activity is defined to reflect the daily desire of the driver, while the cost of each connection arc is used to reflect the combination of desires of activities for the driver. An example of the combination of desires for a driver is that similar shifts are connected. The arcs representing the less-desired activities and less-desired combinations of activities receive higher costs. An illustration of the network layers for drivers $d_1$ and $d_2$ in the NCCR example in Figure 2 for integrated planning is shown in Figure 11, in which the different types of activity arcs are represented as nodes for illustration purposes. The nodes with letters E, M, and L represent sorted shift types: early, midday, and late shifts respectively; while the ones with numbers 1 and 2 represent the single-offs and double-offs respectively. Double-off is defined as at least two consecutive days off, while a single-off is a single day off between two work-related activities. The marked nodes are the activities that drivers prefer. Additionally, a start and an end node are defined on each layer.

Fig. 11 Illustration of the reduced network layers of the NCCR problem in Figure 2.
Figure 12 illustrates the network layer for the CCR problem (integrated planning) in Figure 1, which a network layer is illustrated for a group of drivers, instead of for one driver. Moreover, a sequence of artificial days is appended to the planning period, with a length equal to the largest block length we consider, to maintain the acyclic network layer. Additionally, each network layer might have a different planning period, which is depended on the number of drivers in this group. Therefore, we define $T_m$ be a parameter, which defines the count of days on the planning horizon for each driver/driver group $m$. The details about differences between generating networks for CCR and for NCCR, as well as for sequential planning and for integrated planning, are described in Xie and Suhl (2014). The reduced techniques for the generation of the networks are included in that paper as well. Figures 11 and 12 show the reduced networks for the CCR and NCCR problems.

![Network Diagram for CCR Problem](image)

**Fig. 12** Illustration of the reduced network layer of the CCR problem in Figure 1.

**Model generator**

Based on the rules, problem type, and generated network, the mathematical model will be generated. Continued with Example 1, for company A, the rule is formulated as a set of hard constraints (see constraints (1)); while for company B it is formulated as a set of soft constraints (see constraints (2)). $x_e^a$ is the binary variable that depicts the flow over the activity arc $e \in A^s$, where $A^s$ is the set of all activity arcs $e$ including the extended source edge $e_m^s$ and sink edge $e_m^e$ per driver/driver group $m \in M$. $D_f$ is defined as a set of all day off activities, while $A_{m,t,d}$ is a set of activity arcs available to a given driver/driver group $m \in M$ on day $t \in T_m$ referring to an activity $d$. Constraints (1) ensure that the maximal block length of work-related days is less than $L_m^w$, i.e. at least one day off activity should appear within each $L_m^w + 1$ days. In constraints (2), $\alpha_m$ is defined as a binary variable that indicates if at least one day off activity appears within each $L_m^w + 1$ days. If it is not that case, each of such variable will be punished in the objective (3), with an penalty cost $C^b$. 

---

...
\[ \sum_{t' = t}^{t + I^w_{m,t}} \sum_{e \in A_{m,t',d}} x^a_e \geq 1 \ \forall m \in M, t \in \{1, \ldots, T_m - L^w_{m,t}\}, d \in D_f \]  

(1)

\[ \sum_{t' = t}^{t + L^w_{m,t}} \sum_{e \in A_{m,t',d}} x^a_e + \alpha_m \geq 1 \ \forall m \in M, t \in \{1, \ldots, T_m - L^w_{m,t}\}, d \in D_f \]  

(2)

\[ \min \sum_{m \in M} C^b \cdot \alpha_m \]  

(3)

In Example 2, for company A, the rule for the NCCR problem is formulated as a set of hard constraints (see constraints (4)). \( D_w \) is defined as a set of all work-related activities. A parameter \( T_d \) defines the day of the work-related activity \( d \in D_w \). \( I_d \) is a parameter for defining the count of units of an activity \( d \) which are planned to be allocated. For company B, the CCR problem is differed from the NCCR problem, especially that the capacity of each work-related activity or day off activity is no longer restricted by calendar dates, but by days of the week (see constraints (5)). Therefore, a new set \( T_{m,t} \) is used to define the days (but not including artificial days) in the network for the day \( t \) of the week for the driver \( m \). For example, the day \( t \) of the week is equal to 1, so a set \( T_{m,t} \) might be \{1,8,15,\ldots\}. \( T'_d \) is defined as the day of the week for the work-related activity \( d \).

\[ \sum_{m \in M} \sum_{e \in A_{m,t,d}} x^a_e \leq I_d \ \forall d \in D_w, t = T_d \]  

(4)

\[ \sum_{m \in M} \sum_{t \in T_{m,t}} \sum_{e \in A_{m,t',d}} x^a_e \leq I_d \ \forall d \in D_w, t = T'_d \]  

(5)

3.2.2 Solver Agent

As described earlier, the planner can send the parameter values, such as running time or gap to optimum, to the solver agent. Based on those parameters, the solver agent can choose a suitable solver for the generated model. The solvers available include an optimization solver described in Xie and Suhl (2014); a metaheuristic (MH) solver as shown in Xie et al. (2013b); a column generation (CG) solver as described in Xie et al. (2013a); and a multi-objective metaheuristic (MOMH) solver as described in Xie et al. (2013b). Figure 11 shows the possible solver types for each problem type; for example, DS can be solved using the MIP solvers within short running times.
**Optimization Solver**

The optimization solvers used in our DSS refer to commercial MIP solvers, such as the IBM ILOG CPLEX Optimizer 12.0 (see IBM Corporation (2014) and the Gurobi Optimizer (see Gurobi Optimization (2014)). Such solvers are suitable for small and medium-sized problems.

As shown in Xie and Suhl (2014), the values of (Bin; Cols; Rows; NZs) indicate the difficulty of the crew rostering problem. Due to our experiments, MIP solvers are suitable for the instances with values lower than (200,000; 200,000; 150,000; 500,000) for rota scheduling, and the instances with values lower than (150,000; 150,000; 1,000,000; 1,000,000) for integrated planning.

**Column Generation Solver**

As shown in Figure 12, in the process of column generation the crew rostering problem is divided into two sub-problems, i.e. the restricted master problem and the pricing problem. In the restricted master problem, a master problem is solved with limited variables for activities to get their reduced costs. Based on these reduced costs, the pricing problem is solved in parallel for each driver to generate new variables for activities with negative reduced costs. Then the master problem restarts with new variables being entered, until no new variables for activities can be found. Details of our column generation models for CCR and NCCR, and for the sequential and integrated approaches, are shown in our previous work in Xie et al. (2013a).

According to Xie et al. (2013a), column generation is proved to be the best solver for problems of these sizes, which the MIP solvers cannot solve optimally within 24 hours. However, in the case of solving very large instances with integrated planning, the column generation approach is unable to solve them optimally due to running out of memory.
Fig. 12 Illustration of the process of the column generation approach.

**Metaheuristic Solver**

Three metaheuristic solvers have been implemented in our previous work in Xie et al. (2013b), i.e. simulated annealing, ant colony optimization, and tabu search. The simulated annealing approach is proved to be more suitable than tabu search and ant colony optimization for our problems. Their running times are short (within two hours), however, the quality of their results cannot compete with the quality of those obtained using the MIP solvers and, especially, the column generation solver. Therefore, it is worth using simulated annealing to solve the very large instances with integrated planning, which the column generation solver is unable to solve.

**Multi-Objective Metaheuristic Solver**

A multi-objective metaheuristic solver is chosen to assist the human planner to see the differences in behavior between different specific objective values. Figure 13 illustrates an example solved by multi-objective simulated annealing (see Xie et al. (2013b)). The planner can improve or impair one objective to see relationship between this objective and the others, and the actual solution is marked in blue. After the planner chooses a solution, he/she can save it.

The weight of each objective can easily be calculated based on the following equation:

\[ w_1 * f_1 = w_2 * f_2 = \cdots = w_n * f_n \]

where \( w_i \) is defined as a variable that indicates the weight of the objective \( f_i, \forall i = \{1, \ldots, n\} \). If \( f_i \) is equal to 0, then let \( w_i \) be \( \max\{f_i: i = 1, \ldots, n\} \). After determining the weights, we can solve this problem with only one objective using different solvers described previously in this subsection. As an example, we assume that the planner accepts the marked solution in Figure 13. Table 1 shows the result achieved with the selected weights when using the CPLEX solver. This result is similar to, and even better than, the result shown in Figure 13.
Fig. 13 The set of non-dominated solutions for one instance solved by multi-objective simulated annealing.

<table>
<thead>
<tr>
<th>Objective</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected activity costs</td>
<td>216,780</td>
</tr>
<tr>
<td># underrun days off</td>
<td>48</td>
</tr>
<tr>
<td># overflowed single-offs</td>
<td>0</td>
</tr>
<tr>
<td># unassigned duties</td>
<td>4</td>
</tr>
<tr>
<td># double-off distance violation</td>
<td>0</td>
</tr>
<tr>
<td># underrun standbys</td>
<td>0</td>
</tr>
<tr>
<td># used moved days off</td>
<td>0</td>
</tr>
<tr>
<td># used moved days off on weekends</td>
<td>0</td>
</tr>
<tr>
<td>Maximal overtime (in hours)</td>
<td>4,876</td>
</tr>
</tbody>
</table>

Tab. 1 The solution for rota scheduling for one instance (CPLEX solver).
4 Summary and Outlook

A web-based decision support system, such as the one whose development is, reported in this paper holds several important practical implications for bus companies.

Firstly, a model-solver framework was developed to support the planners’ job. Our mathematical models and solution approaches were embedded in the framework, which can solve the problem automatically according to the planners’ requirements, such as the given running time. The planners do not need to make the decisions about which mathematical model and solution approach should be chosen, and do not even need to know the details of the optimization. Moreover, they will be informed via email or SMS, once a solution is proved by the solver agent.

Secondly, some functions are provided to the drivers, managers, and controllers. Drivers are able to submit their preferences online and download their schedules. Some simple functions are developed for managers and controllers; for example, a manager can see some statistics about the generated schedules, while a controller can send messages to drivers about changes on the day of operations.

However, our prototype does not support all of the needs of a bus company, such as recovery methods on the day of operations. It is difficult for controllers to find alternative crew rosters in a short period of time, without incurring substantial additional costs and causing major disturbance to further operations. Therefore, practical research could also be devoted to developing a web-based decision support system to assist them and propose suitable solutions.

References


