Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

RALF BORNDÖRFER MARKUS REUTHER
THOMAS SCHLECHTE CHRISTOF SCHULZ
ELMAR SWARAT STEFFEN WEIDER

Duty Rostering in Public Transport
-
Facing Preferences, Fairness, and Fatigue

Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany {borndoerfer,reuther, schlechte, schulz, swarat, weider}@zib.de

ZIB-Report 15-44 (September 2015)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

CASPT 2015

Duty Rostering in Public Transport - Facing Preferences, Fairness, and Fatigue

Ralf Borndörfer · Markus Reuther · Thomas Schlechte · Christof Schulz · Elmar Swarat · Steffen Weider

Abstract Duty rostering problems occur in different application contexts and come in different flavors. They give rise to very large scale integer programs which typically have lots of solutions and extremely fractional LP relaxations. In such a situation, heuristics can be a viable algorithmic choice. We propose an improvement method of the Lin-Kernighan type for the solution of duty rostering problems. We illustrate its versatility and solution quality on three different applications in public transit, vehicle routing, and airline rostering with a focus on the management of preferences, fairness, and fatigue, respectively.

Keywords rostering · integer programming · heuristics · preferences · fairness · fatigue

1 Introduction

Duty- or *crew rostering* is an important step in the planning processes of the public transit, railway, airline, and health-care industries. It deals with the concatenation of duties or pairings, corresponding to days of work of one personnel, to a sequence over a period of several weeks. Such a sequence of duties and days-off is called a *roster*; it can be anonymous or personalized, cyclic or fully dated, and scheme-based or free. Rosters must satisfy a variety of rules concerning qualifications, the distribution pattern of duties and days-off, and limits on quantities such as average and total working times.

This work was funded by the Federal Office for Goods Transport and by LBW GbR.

Ralf Borndörfer · Markus Reuther · Thomas Schlechte · Christof Schulz · Elmar Swarat · Steffen Weider
Zuse Institute Berlin, Takustr. 7, D-14195 Berlin
Tel.: +49-30-84185-244
Fax: +49-30-84185-269
E-mail: swarat@zib.de

The objectives in rostering are not only economic quantities such as costs or the size of the staff. Today, the generation of *employee friendly* rosters is also gaining importance as a way to improve the attractiveness of the professions of bus drivers, locomotive drivers, pilots, cabin crews, etc. This objective is not easy to measure, but the match with stated preferences and fairness among all employees are certainly key criteria. Corresponding regulations are often part of collective agreements between employers and labor unions. User friendly rosters can also have economic advantages by reducing the number of staff away sick. Fatigue management, i.e., the generation of rosters that fit with the biorhythm, also veers toward this direction, as rests on duty can make an otherwise illegal roster feasible.

The more complex the rules and objectives, the larger the need for advanced mathematical optimization methods becomes. These must, however, be versatile to deal with a wide variety of fast changing requirements in different companies and industrial sectors. Many algorithms have been proposed for this purpose, the two main approaches being set partitioning and covering based column generation algorithms and all kinds of local search methods. We refer the reader to the surveys of Kohl and Karisch (2004) and Ernst et al (2004), see also the 124 pages annotated bibliography of Ernst et al (2004).

We also propose a local search heuristic with various different neighbourhoods. One of them is based on a chained edge-exchange-approach as proposed by Kanellakis and Papadimitriou (1980) for the Asymmetric Traveling Salesman Problem (ATSP). In Kanellakis and Papadimitriou (1980) the original heuristic of Lin and Kernighan (1973) for the symmetric Traveling Salesman Problem (TSP) is adjusted to the ATSP. Our heuristic is based on interpreting a cyclic roster as a directed cycle through all duties. In this way, the cyclic rostering problem reduces to an asymmetric traveling salesman problem (ATSP) with additional constraints, for which good 2- and 3-opt procedures have been developed. We show how these methods can be improved to a variable depth search in the style of Lin and Kernighan (LK) for the solution of large scale rostering problems. We show the versatility of the method by discussing three applications on handling preferences, fairness, and fatigue in three different industries. We argue that our method produces high quality solutions for real-world rostering planning problems.

The paper is structured as follows. In Section 2 we characterize rostering problems according to their variants and rules. In the following Section 3 basic model formulations of rostering problems are introduced. In Section 4 the adaption of the LK-Heuristic to the rostering problems is discussed in detail. Three rostering applications are then presented in Section 5 and finally Section 6 gives computational results showing the applicability of the heuristic.

2 Rostering in Practice

In the following, we want to characterize variants of rostering, and show how these variants occur in applications stemming from projects at Zuse Institute Berlin.

2.1 Variants of Rostering

Rostering is the problem of assigning a set of *duties* to employees obeying certain rules. A duty is a specified set of tasks, which a single employee can perform on a single work day.

Variants of rostering can be distinguished by their following properties:

Calendar Days vs. Days of Operation The duties to be rostered are either duties of a certain planning period, e.g., the following month, or duties of idealized days of operation. Typically, planners group working days with similiar workload to so called *days of operation*. In public transport companies the days from Monday till Friday often have the same schedules, while Saturday, and Sunday may have very different schedules. This results in duties for a day of operation "Mon-Fri", and duties for the days of operation "Sat" and "Sun". The days of operation form the so called *standard week*. These duties are then inserted in a roster. When planning with days of operations, often also a "standard" roster is planned, which ignores deviations in the workload stemming from public holidays, road works or similar events.

Personalized vs. Anonymous Duties can be assigned to specific employees, considering their preferences for certain types of duties on certain days, preplanned duties (like training courses), or planned absences. Or duties can be assigned to rows (or in airline context also called lines) of a roster, which will be in a subsequent step assigned to specific employees, e.g. by preferential bidding or by another planning step.

Cyclic vs. Acyclic If rostering is used to plan Days of Operations for anonymous employees it is possible to create cyclic rosters. I.e., the rows are planned such that an employee can subsequently perform the duties of all the rows without violating rules. Further, the rows form a cycle, that is, after performing the last row, employees start again with the first row. The result is a fair roster in which all employees have to conduct the same duties in the same sequence.

Fixed Off Days Days off can be planned in advance and given as input of the problem or they may be freely distributed by the optimizer obeying specific rules.

2.2 Rules

The rostering rules in all these planning variants are very similar. There are hard rules coming from laws and regulations and soft rules for employee friend-

liness and fairness. The soft rules, e.g., creating as much completely free weekends as possible, avoiding isolated duties, or minimizing unfavorable sequences of duties, can be enforced by penalties.

The most important hard rules are

- a maximal working time per week,
- a minimum break duration between two successive duties,
- and a minimum continuous weekly rest time of 48 hours.

These must not be violated. These hard rules are there for safety and health reasons.

The airline industry goes even one step further. Today, all planning processes in the airline industry are heavily supported by mathematical optimization methods, e.g., in the field of pairing optimization Barnhart et al (1996) and Borndörfer et al (2005). This well established technology allows for integrating new aspects such as fatigue which is based on a complex biorhythms model to identify sleep periods, instead of the simple rules stipulated by laws. In particular, an airline crew has to face several changing the clocks during a roster. The integration of fatigue and alertness in an airline rostering and crew management context is described in Rangan et al (2013). The authors of Goel and Vidal (2014) address the fatigue of drivers in road freight transportation.

3 Models

Rostering problems are typically formulated as multi-commodity flow problems with additional constraints or as their path-decomposition (see Dantzig and Wolfe (1960)) resulting in set partitioning problems. In the multi-commodity flow formulation, each duty is interpreted as a node in a graph. A directed arc in this graph means, that the adjacent duties can be performed subsequently by the same driver. Furthermore, we have one commodity per employee or, in the anonymous case, one per row of the roster. The additional constraints are typically knapsack constraints to ensure, e.g., maximum working times per row, or infeasible path constraints, which forbid unwanted combinations of duties in a row. Each feasible row of a roster corresponds to a feasible path in this graph. The objective of this problem is to find a cost minimal set of paths which covers each duty exactly once.

The flow formulation is more efficient if we have many feasible paths and only a few hard rules that can be formulated by a small number of additional constraints. In the case of many hard rules one should use the set partitioning formulation, because flow formulations become inefficient in our experience, if many additional constraints exist. That is the main reason why we use different models for our rostering applications.

3.1 Flow Model

In the following we shortly present the basic flow model for rostering. Let D be the set of duties which should be covered by a set of rows M. Our graph has then one node for every duty plus two additional nodes s and t as source and sink. We call this set of nodes V. If two duties u and v can be performed in direct succession in the same row, there is an arc (u, v) in the graph. Additionally, we have arcs (s, v) for every duty v that can be the first in a row, and arcs (u, t) for duties u which can be the last ones, respectively. Let A be the set of all feasible arcs of the underlying scheduling graph. Then our flow rostering model (FROSTER) looks as follows:

$$\min \sum_{m \in M} \sum_{a \in A} c_{am} x_{am} \qquad \text{(FROSTER)} \tag{1}$$

$$\sum_{m \in M} \sum_{a \in \delta^{\text{in}}(v)} x_{am} = 1, \qquad \forall v \in D, \tag{2}$$

$$\sum_{a \in \delta^{\text{in}}(v)} x_{am} - \sum_{a \in \delta^{\text{out}}(v)} x_{am} = 0, \qquad \forall m \in M, \forall v \in D, \tag{3}$$

$$\sum_{a \in A} b_{ar} x_{am} \leq u_{rm}, \qquad \forall m \in M, \forall r \in R, \tag{4}$$

$$\sum_{a \in I} x_{am} \leq |I| - 1, \quad \forall I \in \mathcal{I}, \forall m \in M, \tag{5}$$

$$x_{am} \in \{0, 1\}, \quad \forall a \in A, \forall m \in M. \tag{6}$$

Variables x_{am} are one if arc a is used in row m or zero otherwise. Coefficients c_{am} give the cost of using an arc a by row m. Constraints (2) guarantee that every duty is covered by exactly one roster. Constraints (3) guarantee that every row is a path in the network. These are equivalent to flow conservation constraints of a multi-commodity-flow problem. Constraints (4) are resource constraints. Coefficients b_{ar} define a resource consumption of a resource r on arc a. A resource can be, e.g., paid time, or a number of working days. These equations constrain the consumption of a resource per row to an upper bound u_{rm}. Finally, constraints (5) forbid infeasible sequences of arcs. The set $\mathcal{I}$ is a set of sets of arcs. Each of these sets of arcs must not be together in a single row.

3.2 Set-Partitioning Model

The corresponding column oriented set-partitioning model (CROSTER) is defined on the same graph $G = (V, A)$. Let P be the set of paths in G that

correspond to valid rows. Then our model (CROSTER) is:

$$\min \sum_{p \in P} d_p y_p \qquad \text{(CROSTER)} \tag{7}$$

$$\sum_{p \ni d} y_p = 1, \qquad \forall d \in D, \tag{8}$$

$$\sum_{p \in P} e_{rp} y_p \leq u_r, \qquad \forall r \in R, \tag{9}$$

$$y_p \in \{0, 1\}, \quad \forall p \in P. \tag{10}$$

The cost coefficients d_p give the cost for a row p. Constraints (8) guarantee that every duty is covered by one row. Constraints (9) ensure, that resource consumptions of a resource r for the whole roster do not exceed their upper bound u_r. Resource constraints for a single row and infeasible path constraints are not needed in (CROSTER), because the corresponding violating paths are simply not included in the set of all feasible paths P.

In practical applications we have to extend the basic models to incorporate their peculiarities. We will discuss these extensions in the corresponding sections.

4 The DEX approach

Real world instances of rostering problems cannot be solved directly by general-purpose solvers in acceptable time. Thus, we present a multi-phase-heuristic for the construction of roster schemes which is suited for all variants presented in Section 2.1.

Our approach, called DEX (for Dynamic-depth-EXchange heuristic), is a heuristic which consists of chaining different types of k-opt improvement steps.

At first, we construct a feasible acyclic roster scheme with a *greedy-heuristic* (START). This is done sequentially by solving a constrained shortest path problem for each row in the planning graph. We remove in each iteration the duties from the graph which are assigned to the row. Main problem of this heuristic is that for the last rows are too few duties left to build "good" rows. Therefore, we improve this solution in the next steps.

In the anonymus case the solution obtained by START is improved by a sequence of k-opt steps based on an assignment problem. For the cyclic rostering problem we connect the acyclic schedules to one cyclic roster. For the acyclic rostering problem we connect the end node of each acyclic schedule with its start node via an artificial edge. Then the solution is gradually improved by a heuristic derived from a heuristic by Kanellakis and Papadimitriou (1980) for the Asymmetric-Traveling-Salesman-Problem. This heuristic is an edge-exchange-approach based on a heuristic by Lin and Kernighan (1973) for the traveling-salesman-problem. We call it GENLK. GENLK tries to improve the start solution by a sequence of three- and four-edge-exchanges. Deteriorations of the solution quality is allowed. The concatenation of the-three-edge

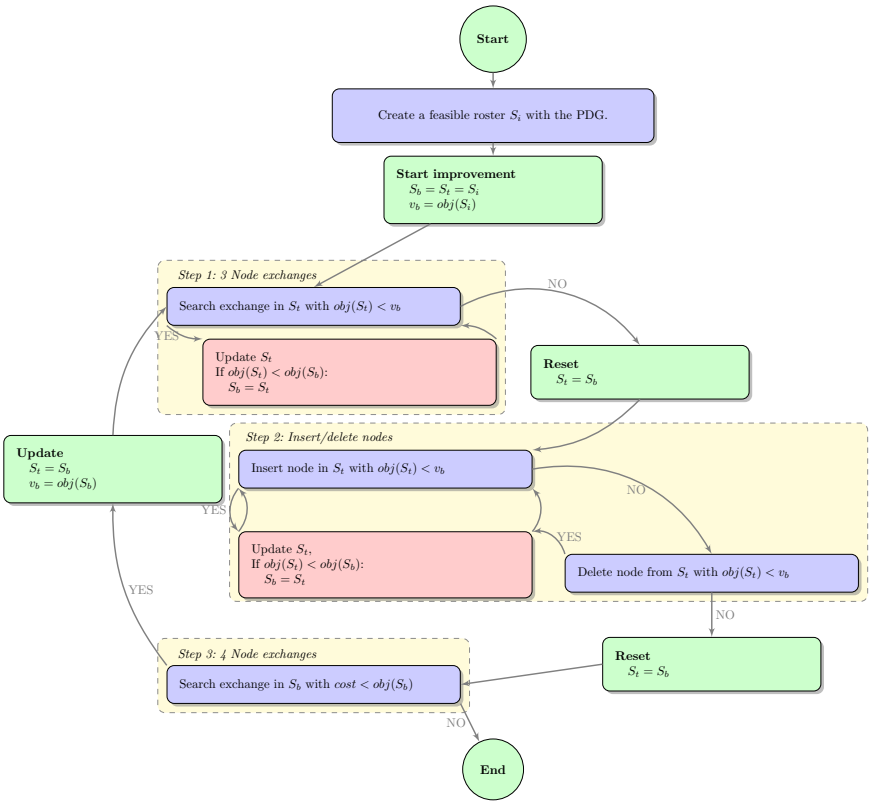

Fig. 1: LK heuristic for personalized rostering.

exchanges is constructed in such a way that the last added edge is removed in the next exchange. If we do i three edge exchanges in a row, we have done a $2 * (i - 1) + 3$ edge exchange. With this method we can perform a series of edge exchanges that results in a k-opt step.

For the personalized case we use a similar approach, but we replace edge-exchanges by node-exchanges. Figure 1 shows a description of this heuristic. In order to allow temporary deteriorations of the solution, GENLK saves not only the best incumbent solution S_b with cost v_b, but also a temporary solution S_t. All exchanges are done on the temporary solution S_t. Each iteration I_j consist of three different types of node exchanges. These are *three node exchanges*, *four node exchanges* and *node insertion and deletion*. We start with iteration 0 ($j = 0$). The set of nodes is denoted by V. The first step of each iteration j is to find feasible *three node exchanges* in S_t. The upper bound u for the deterioration is the objective value before iteration j. Let $n_1, n_2, n_3 \in V$ be the three nodes which should be switched with $v_1, v_2, v_3 \in S_t$, $S_{\hat{t}}$ the temporary solution after the exchange and $obj(S_{\hat{t}})$ the objective value of $S_{\hat{t}}$. An exchange is feasible if and only if $S_{\hat{t}}$ satisfies all hard rules, $(n_1, n_2, n_3) \neq (v_1, v_2, v_3)$ and

$obj(S_{\hat{t}}) < u$. If an exchange is feasible S_t is set to $S_{\hat{t}}$. If also $obj(S_{\hat{t}}) < obj(S_b)$, S_b is set to the new temporary solution $S_{\hat{t}}$. Then we search for a new *three node exchange* in S_t. The first step is terminated if no suitable *three node exchange* can be found anymore.

If that is the case S_t is set to S_b and we start the second step. Let W be the set of nodes that have been inserted in S_t in this step and U be the set of nodes which have been deleted from S_t in this step. The heuristic tries iteratively to insert a node $n \in V$ in S_t with $n \notin S_t$ and $n \notin U$. Let $S_{\hat{t}}$ be the solution after the insertion. The insertion is done if $obj(S_{\hat{t}}) < u$. Then S_t is set to $S_{\hat{t}}$ and n is added to W. S_b is set to $S_{\hat{t}}$ if $obj(S_{\hat{t}}) < obj(S_b)$. If no more insertion is possible we try to delete one node $d \in S_t$ with $d \notin W$. The deletion is done if the new objective value is less than u. Then d is added to U and we start the insertion again. If no deletion can be found step two terminates and S_t is set to S_b.

The last step of each iteration is to find a feasible *four node exchange*, which improves the objective value of the best solution S_b. If no such exchange exists the LK heuristic terminates, otherwise an exchange is done and the iteration $j + 1$ starts with the first step again.

For the anonymus case we do also a k-opt node-exchange step. Iteratively all duties that are planned for a specific single day of the planning horizon are removed from the roster. Then a bipartite network is constructed. One set of nodes of this network is formed by the removed duties. The other set of nodes is the set of possible rows. An arc connects a duty with a row, if the duty can be inserted in this row. The cost of the arc is the cost difference between the original roster and the roster which is created by inserting the duty in the new row. We then solve the assignment problem on this graph. This is repeated until no improvement of the roster occurs.

Finally, also a column generation approach can be used to improve the solution found by DEX. Here we use the heuristic proposed in Borndörfer et al (2008) to solve the model (CRoster). However, in most cases the solution of DEX is already near the optimum, such that the column generation approach has difficulties to improve it.

5 Applications

In this section we present three real-world rostering applications, their specific characteristics and how the corresponding optimization problems are modeled.

5.1 Rostering in Toll Enforcement

First, we want to address a rostering problem arising in the Federal Office for Goods Transport (BAG). One of the tasks of BAG is to enforce the truck toll on German long-distance roads BAG (2012). In a project with BAG we developed a method to optimize mobile control tours (shortly called *tours*), that

are conducted by *control groups* of one or two inspectors. In addition, feasible rosters of the inspectors should be generated. This relates to an integrated tour planning and duty rostering problem; it is called *Toll Enforcement Problem* (TEP). In (Borndörfer et al, 2012) a case study is presented that indicates the impact of the mathematical optimization approach on the enforcement planning. In Borndörfer et al (2013) we gave an extensive description of the modeling power of our approach, including an analysis of the bi-criterion character of the TEP. A main difference to other rostering approaches is that crews can only conduct controls in the area of their home depots. Hence, tours and crews must be planned simultaneously to prevent the planning of tours where no crew is disposable for.

Rostering in the area of enforcement or security inspection planning has some peculiarities in comparison to our basic models: The duties are not given in advance, they have to be created by *tasks* consisting of controlling a certain *section* of a highway in a certain time interval. For every section a minimum control frequency and an indicator of its importance is given. A duty is then a combination of such tasks, in our case of exactly two of them. TEP consists of finding duties and rosters simultaneously. Every duty will be performed by a control group.

Fairness and employee preferences are important criteria for the rostering of the employees. This includes, that inappropriate sequences of duties should be avoided. In particular, a change of the duty starting time on two subsequent days, i.e., a *rotation*, is not desired. Especially *backward rotations*, i.e., the duty on the next day starts earlier, alter the human biorhythm and affect the sleep. This is similar to the recently studied fatigue criterion in the airline industry, which will be presented in Section 5.3. Since avoiding certain sequences of duties represents a soft rule, there are penalties in the objective function on arcs representing rotations.

An important example for preferences are requests of employees to get a "Free", that is a day without a duty, on specific days. Since these requests must be respected, they are modeled as hard constraints in our model. Fairness, on the other hand, is considered by distributing unwanted duties, such as night and weekend duties, rather equally among the inspectors. However, if an employee prefers to work on weekends or at night, this will be also taken into account.

5.1.1 Model of TEP

A duty in the TEP is now a tour starting at a certain time in a certain depot. Every tour ends in the same depot where it started. In our scenarios exactly two sections are controlled during a tour each for a fixed time interval, in our case 4 hours. Let J be the set of days in the planning horizon of TEP. Further, we denote the set of sections by S. Let here D be the set of all control tours, the control tours are equivalent to the duties of model (FRoster). The control tours are also the nodes of our planning graph plus again artificial nodes for source and sink of the network. Let $D_f \subset D$ be the set of all control tours that

are feasible for control group $f \in F$ and D_j the set of control tours at day $j \in J$. D_s denotes all control tours that contain a section $s \in S$, and D_i all control tours at time interval $i \in T$. A day is partitioned in six time intervals of 4 hours each. Control groups are sets of one or two employees $m \in M$. By κ_s the minimum control quota, i.e., the minimum number of controls on section s during the planning horizon, is denoted and in addition by β_{ds} the number of controls of s during control tour d (could be one or two).

The overall goal is to compute a reward-maximal set of control tours. Hence, each tour gives a reward depending on the amount of traffic on the controlled sections and on the time and day of the week when the tour is scheduled. The profit w_d of a tour d is defined as the sum of the rewards of its controlled sections depending on time and day. We introduce binary variables z_d, $d \in D$, to decide if a control tour d is chosen or not.

We use now our basic model (FRoster) and extend it to incorporate the control tours as follows:

$$\max \sum_{d \in D} w_d z_d - \sum_{m \in M} \sum_{a \in A} c_{am} x_{am} \qquad (TEP) \tag{11}$$

$$\sum_{d \in D_f \cap D_j} z_d \leq 1, \qquad \forall f \in F, \forall j \in J, \tag{12}$$

$$\sum_{d \in D_s \cap D_j \cap D_i} z_d \leq 1, \qquad \forall s \in S, \forall j \in J, \forall i \in T \tag{13}$$

$$\sum_{d \in D_s} \beta_{ds} z_d \geq \kappa_s, \qquad \forall s \in S, \tag{14}$$

$$\sum_{a \in \delta^{\text{in}}(v)} x_{am} - \sum_{a \in \delta^{\text{out}}(v)} x_{am} = 0, \qquad \forall m \in M, \forall d \in D, \tag{15}$$

$$n_d z_d - \sum_{m \in M} \sum_{a \in \delta^{\text{in}}(d) \in A} x_{am} = 0, \qquad \forall d \in D, \tag{16}$$

$$\sum_{a \in A} b_{ar} x_{am} \leq u_{rm}, \qquad \forall m \in M, \forall r \in R \tag{17}$$

$$\sum_{a \in I} x_{am} \leq |I| - 1, \quad \forall I \in \mathcal{I}, \forall m \in M \tag{18}$$

$$x_{am} \in \{0, 1\}, \quad \forall a \in A, \forall m \in M, \tag{19}$$

$$z_d \in \{0, 1\}, \quad \forall d \in D. \tag{20}$$

In the objective function (11) the profit of the selected tours minus the costs of the sequence arcs are maximized. Contraints (15) and (17) till (19) are equivalent to model (FRoster). Constraints (16) couples the tour selection variables z_d with the arc variables x_{am}. Here n_d gives the number of employees needed for tour d. Constraints (12) guarantee that every control group performs only one tour per day, constraints (13) guarantee that a section is not controlled by more than one group at every point in time and constraint (14)

ensure the minimum control frequency. The costs c_{am} represent penalties on the duty sequence arcs.

We call the subpart of the model, that only involves the tour variables z *Tour Planning Problem (*TPP*).*

5.1.2 Solution method

In contrast to many other rostering problems model (TEP) is directly solvable by a general-purpose solver such as CPLEX. Since the beginning of 2014, our algorithm and software based on this model and CPLEX is in production to plan all toll enforcers of BAG in Germany. In this report we focus on the results obtained by our approach DEX proposed in Section 4 and compare them to the results obtained by CPLEX. We will show, that DEX is able to compute good solutions in a very short time.

The main difference to standard rostering is, that we have to optimize duties, which are here equivalent to tours, and a roster simultaneously. The set of possible tours can become very large, so we tried to restrict its size: We generate tours by solving TPP with relaxing constraints (12) such that each group can perform three or four tours on a day. We compare that with explicitly enumerating all tours. Because of the local control restriction for each employee and the fact that a tour consists of only two sections, it is possible to generate all tours by a simple enumeration.

After tours are generated, a slightly modified variant of DEX is used to compute a feasible roster. DEX has to be modified, because not all duties have to be covered. The number of duties used is a result of the minimum control constraints (14), the number of control groups available, and the rewards per duty in the solution.

5.2 Cyclic rostering in public transport

We present here a cyclic rostering problem of a public transport company denoted by (RPT). It is modeled as a set partitioning problem because of the multitude of difficult rules on paths. We use the terminology of model (CRoster). Here one path, called *row* of the roster, is a feasible duty schedule for one (anonymous) employee for the planning period. The goal is to find a cost minimal set of rows which covers all duties exactly once. In the cyclic case we need also subtour elimination constraints (24), because we want to find a single cycle for all rows. To ensure that each row in the solution has exactly one successor and one predecessor, we define binary variables y_{rs}. For two rows $r, s \in P$ variable y_{rs} is one if and only if row s is the successor of r. The constraints (22) and (23) ensure that there is exactly one predecessor and one successor for each row r with $x_r = 1$. Note that for the acyclic case constraints (22) to (24) can be omitted. The majority of the rostering rules can be checked within the construction of P. Thus, they are implicitly fulfilled by the model. We need only explicit constraints for the interval rules (25) that span more

than one row. For this the set of interval rules is denoted by I. For each pair of interval rule and row we have given a resource consumption b_{ri}. So there are certain sets of rows σ_i, an upper bound u_i and a lower bound l_i for the resource consumption for each interval rule $i \in I$. The set S_i contains all sets σ_i for interval rule i. Constraint (25) ensures that the resource consumption in σ_i is between l_i and u_i. Finally, (26) and (27) ensure that x and y are binary. The value of the objective function is the sum of the costs of the selected rows and the sum of the costs of the links between the rows. The costs consist of soft rule violations and fix costs per employee.

$$\min \quad \sum_{p \in P} c_p x_p + \sum_{r,s \in P} d_{rs} y_{rs} \qquad \text{(RPT)}$$

$$s.t. \quad \sum_{p \ni d} x_p = 1 \qquad \forall d \in D \qquad (21)$$

$$\sum_{s \in P} y_{rs} = x_r \qquad \forall r \in P \qquad (22)$$

$$\sum_{s \in P} y_{sr} = x_r \qquad \forall r \in P \qquad (23)$$

$$\sum_{r \notin S, s \in S} y_{rs} \geq x_p + x_q - 1 \forall p \notin S, q \in S, S \subset P \qquad (24)$$

$$\sum_{r \in \sigma_i} b_{ri} x_r \in [l_i, u_i] \qquad \forall \sigma_i \in S_i, i \in I \qquad (25)$$

$$x_p \in \{0,1\} \qquad \forall p \in P \qquad (26)$$

$$y_{rs} \in \{0,1\} \qquad \forall r, s \in P \qquad (27)$$

5.3 Rostering for air traffic considering fatigue

In this section, we will briefly discuss how to integrate fatigue requirements into model (CROSTER) which is solved by a column generation algorithm DEX.

In classical column generation approaches for crew scheduling, the pricing problem can be interpreted as a resource constrained shortest path problem w.r.t. the current duals, see Barnhart et al (1996). In general, dynamic programming algorithms are used as pricers, because such methods are very fast and able to produce several substantially different columns (with hopefully large reduced costs) at once. The flexibility of dynamic programming and labeling approaches provides the possibility to consider more complicated non-linear resource rules, see e.g. Smith et al (2012) for the concept of replenishment arcs. Sub-paths are extended by adding new tasks at the end, leading to new sub-paths with the corresponding updated label states. Either the new labels are better than the ones stored at the entering node, then the label set is updated or the search is stopped because of pruning by an already existing

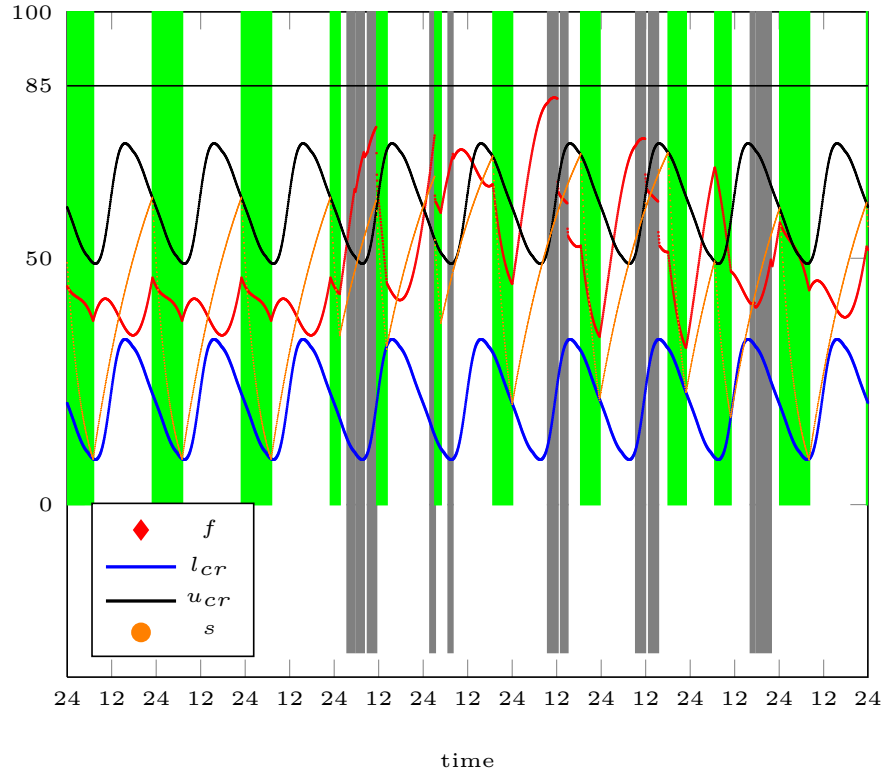

Fig. 2: Sleep process s, circadian rhythm l_{cr} and u_{cr}, and fatigue function f.

dominant label or because of an infeasible label state. (Assume a maximum is given and the resource consumptions exceed this value after the extension.) This classical search procedure can also be applied if a more complex calculation than just adding (or resetting) resources is needed which is obviously the case for fatigue.

In general a maximum fatigue rule has to be checked; it is denoted by U. By storing the "fatigue" state at the end of a sub-path, the fatigue calculation has only to be done for the part of the extension. This is possible because the fatigue state can be exactly described by a few values (circadian component, sleep-related component, and work-related component). Thus, if the local maximum of this interval does not violate the given U the constructed sub-path is feasible, otherwise we can stop because of a trivial rule violation. Figure 2 shows a roster example for nine days of some crew member. Flights are marked as gray areas and sleeps are marked as green areas. Thus, the first three days no duties or flights are assigned to the crew member. We denote the circadian biorhythm by l_{cr} and u_{cr}. The value of the sleep process is plotted as s and the fatigue value as function f. In case that the value of the

sleep process s intersects the function u_{cr} the crew member starts sleeping. We assume $U = 85$ which is fulfilled because the maximum of the function f is achieved on day 6 around 11:20 a.m. with 82.6. The natural behavior of the fatigue function and the human biorhythm can be observed in Figure 2, that is:

- during sleep one recovers,
- staying awake makes one tired,
- working makes one tired even faster,
- and one cannot sleep on command.

For more details on the precise fatigue function and their components we refer to Samel et al (1997). The determination of the local fatigue maximum is rather complicated taking different time-zones into account and definitely time-consuming due to the non-linearity. However, a significant speed-up can be achieved by storing result values for always recurring and cumulative calculations of the components. Note that the objective (b.o. the dual values) of a sub-path changes in each iteration of the column generation method, but the fatigue calculation will not differ. A cumulative fatigue function is integrated within the column generation based solution approach and leads to a reasonable and acceptable increase of the total computation time. Thus, the crew scheduling and optimization software NetLine/Crew Optimizer Suite of Lufthansa Systems GmbH & Co.KG provides the integration of fatigue evaluation into the complete process of crew scheduling and optimization.

Imposing fatigue in our heuristic DEX is in principle easy: after each exchange the fatigue function has to be computed for the respective rows. The main problem is to do this computation effectively, because in every run of DEX millions of potential exchanges are checked. However, we see a huge potential to anticipate redundant checks by using simple approximations, i.e., overestimation of the fatigue values.

6 Results

6.1 Cyclic rostering in urban public transport

Our rostering algorithm for public transport companies is integrated in the commercially available planning suite IVU.plan from the IVU traffic technologies AG. We have tested our heuristic on real scenarios from a German urban public transit company. Table 1 shows results of one scenario from a medium size company. For the computations we used a Dell Precision T1700 workstation with an 8-core Intel Xeon CPU with 3,40 GHz and Ubuntu 14.04 as operating system. Target working time for the employees is 39 hours per week. If they work more than 39 hours, overtime must be paid. For employee satisfaction there are three rules which are especially important. First, the employees want either to work a whole weekend or not at all at a weekend. Second, the roster should contain as less short off days as possible. A short off day is an off

Aspect	Manual solution	Heuristic approach
Employees	39	39
PT/Week (optimal 39:00)	[37:56,40:11]	[38:45,40:05]
Separated weekend duties	12	9
Free weekends	12	16
Stand-alone duties	0	0
Number of short frees	13	6

Table 1: Medium urban cyclic rostering scenario with 157 duties

day with less off time than a normal one. E.g. a day off is regular if there are at least 45 hours between adjoining duties, a shortened day off is allowed to have as less as 36 hours between the shifts. By German law shortened off days are allowed, but must be compensated within 3 weeks. Third, there should be no stand-alone duties between two off days.

Table 1 shows that in our solution the working time is closer to 39 hours than in the manual solution provided by the public transit company. The results show that one can reach a better employee satisfaction without additional costs. The number of free weekends in the planning period increases from 12 to 16 and the number of short free days decreases from 13 to 6. Like in the manual solution there are no stand-alone duties in the optimized plan. The running time of this scenario was about 45 minutes.

6.2 Acyclic rostering in toll enforcement

For the TEP we have compared the results obtained by DEX with the solutions from the integrated IP model. For the computational comparison, we consider real-world instances from the test environment as well as from the production operation at the BAG. First, we give some basic settings of the computations. All computations were done on a Dell Power Edge M620 workstation with an 8-core Intel Xeon CPU with 2,70 GHz and Ubuntu 14.04 as operating system. For the IP Cplex 12.6 by IBM is used as solver with the default parameter setting and maximal eight threads. The memory limit for the branch and bound tree was set to 40 GB.

We choose nine scenarios from seven different *control areas* also called *regions*, denoted by r_1 until r_7. The regions are optimized separately at BAG. They have a planning horizon of several weeks each. Table 2 gives the key characteristics of all instances. The first column gives the name of the instance, the second the control area, and the third the number of inspectors of this region. Column four shows the number of control sections belonging to each region. Another important aspect is the number of fixed duties like days-off, vacations, free requests or other obligatory duties like staff meetings. They must not be changed. Therefore, fixed duties reduce the problem complexity by implying to use certain arcs in the underlying planning graph. They are indicated in the fifth column. The number of different duty types of a scenario

Instance	Region	Inspectors	Sections	Fixed Duties	Duty Types	IP Rows	IP Columns
I1	r_1	21	17	253	6	7738	96526
I2	r_1	22	22	272	4	8010	101791
I3	r_1	22	22	170	7	13095	392563
I4	r_2	23	24	189	8	15417	402285
I5	r_3	22	22	8	12	20366	1611980
I6	r_4	19	17	177	8	11246	295388
I7	r_5	23	19	182	9	15067	501340
I8	r_6	24	28	57	8	15246	712228
I9	r_7	21	16	0	10	17369	904878

Table 2: Key characteristics of the instances to compare the LK algorithm with a pure IP approach.

	IP			DEX		
instance	obj 5 min	obj 12 hours	gap(%)	obj	time(s)	gap(%)
I1	359,077.13	359,077.13	0.00	353,181.93	36	1.67
I2	332,283.74	332,556.68	0.00	326,612.72	82	1.82
I3	–	513,998.85	0.00	499,804.91	127	2.84
I4	–	346,788.20	0.88	340,130.17	179	2.85
I5	–	805,294.03	1.08	790,459.06	420	2.98
I6	154,142.04	154,270.86	0.03	151,769.96	151	1.68
I7	–	335,015.01	0.72	331,301.40	162	1.85
I8	–	373,509.57	85.84	671,729.68	415	3.33
I9	–	437,426.84	4.76	441,274.11	474	3.85

Table 3: Results for real-world instances of the TEP with the IP method and DEX without solving the TPP before.

also influences its complexity. It is shown in column six. An increasing number of duty types leads to more duties and therefore to a more complex scenario. In particular, in our application a duty type is defined by the allowed starting times of the corresponding duties. The last two columns show the number of constraints and variables in the TEP-IP model.

In our experiments, two different time settings for the IP were applied. First, we used a time limit of five minutes to compare results, to have a fair comparison with DEX, which also runs about 5 minutes. In the second experiment, a time limit of 12 hours was set to check if the IP method can find significantly better or eventually even optimal solutions. As described in Section 5.1.2 we ran DEX with all possible duties or with an subset of them.

Hence, we performed four runs for each instance, two with the IP method using different time limits and two with DEX using different duty sets. Then, we compared the solution quality of the IP method with the ones from DEX in terms of running time and objective value. Table 3 shows the results of DEX with all possible duties. Table 4 shows the results of DEX with a subset of duties generated by TPP. In these tables the first column gives the name of the instance. Column two shows the IP solution after five minutes of runtime and

instance	IP obj 5 min	IP obj 12 hours	IP gap(%)	DEX obj	DEX time(s)	DEX gap(%)
I1	359,077.13	359,077.13	0.00	352,939.36	30	1.74
I2	332,283.74	332,556.68	0.00	328,985.85	62	1.09
I3	–	513,998.85	0.00	485,343.35	40	5.90
I4	–	346,788.20	0.88	342,407.95	91	2.17
I5	–	805,294.03	1.08	776,836.70	196	4.78
I6	154,142.04	154,270.86	0.03	150,841.02	28	2.30
I7	–	335,015.01	0.72	330,247.18	34	2.17
I8	–	373,509.57	85.84	668,771.62	185	3.79
I9	–	437,426.84	4.76	427,642.76	175	7.16

Table 4: Results for real-world instances of the TEP with the IP method and DEX with solving the TPP before.

column three the IP solution after 12 hours of runtime. In the fourth column the optimality gap of the IP solution after 12 hours is displayed. Columns five to seven show characteristics of the solution of DEX. Column five gives the objective value, column six the runtime and column seven the optimality gap of DEX with respect to the best known dual bound from the IP runs.

The instances are very different, especially with regard to fixed duties and duty types. This is also indicated by the results of the test. Table 3 shows that the instance I1 can be solved to optimality with the IP method within 5 minutes. For two other instances, I2 and I6, feasible solutions can be found within five minutes that are better than the solutions by the heuristic. Hence, for these three instances the default IP method clearly outperforms the heuristic. For all other instances no feasible integer solution is found within five minutes, while the heuristic always finds a solution no matter if tours are generated by the TPP or not. Hence, we can state that the heuristic is the first choice for medium-size and large instances to find feasible solutions in good quality very fast.

If one compares the results of the 12-hour runs with the heuristic it becomes clear that the strength of the IP method lies in finding the overall best solutions. Four instances can (almost) be solved to optimality. For further three instances, I4, I5 and I7, there remains only a gap of about 1% or less between the primal and the dual bound. For the first seven instances, i.e., also for the difficult instance I5 with almost no fixed duties and the highest quantity of duty types, the objective value of the 12-hour-run is better than the one obtained by the heuristic. In contrast to that, especially in case of the difficult instance I8 the performance of the IP is very poor. One can summarize that the heuristic is able to produce good solutions for each instance, but they are never optimal. The IP method could perform poorly with some large and difficult instances. There the heuristic is even with respect to the solution quality the first choice.

Comparing the results of Tables 3 and 4 is seems that enumerating the tours by the heuristic itself leads to better results for most of the instances but there are two instances where it is different. The approach to compute the

tours with the TPP as a first step represents a more sequential view on the problem. Hence, integration of rostering and tour generation is an important characteristic of the TEP.

Acknowledgments

We want to thank our partners from IVU AG and our users from BAG, in particular Sebastian Adlers-Flügel, Hans-Stefan Madlung, Christian Hoffmann, Thomas Dankert, Daniel Schneider, Ralf Haas and Uta Sperling.

We want to thank our cooperation partner Lufthansa Systems GmbH & Co.KG, in particular Andreas Soehlke and Julika Mehrgardt.

References

BAG (2012) Bundesamt für Güterverkehr - Lkw-Maut. available online, URL `http://www.bag.bund.de/DE/Navigation/Verkehrsaufgaben/Lkw-Maut/lkw-maut_node.html`

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1996) Branch-and-Price: Column Generation for Solving Huge Integer Programs. Operations Research 46:316–329

Borndörfer R, Schelten U, Schlechte T, Weider S (2005) A Column Generation Approach to Airline Crew Scheduling. In: OR, Springer, pp 343–348

Borndörfer R, Löbel A, Weider S (2008) A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit. In: Hickman M, Mirchandani P, Voß S (eds) Computer-aided Systems in Public Transport, Springer Verlag, Lecture Notes in Economics and Mathematical Systems, vol 600, pp 3–24, URL `http://opus.kobv.de/zib/volltexte/2004/790/`, ZIB Report 04-14

Borndörfer R, Sagnol G, Swarat E (2012) A Case Study on Optimizing Toll Enforcements on Motorways. In: Ravizza S, Holborn P (eds) 3rd Student Conference on Operational Research, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, OpenAccess Series in Informatics (OASIcs), vol 22, pp 1–10, DOI http://dx.doi.org/10.4230/OASIcs.SCOR.2012.1, URL `http://drops.dagstuhl.de/opus/volltexte/2012/3541`

Borndörfer R, Sagnol G, Schlechte T, Swarat E (2013) Optimal Toll Enforcement - an Integration of Vehicle Routing and Duty Rostering. Tech. Rep. ZIB Report 13-79, Zuse-Institut Berlin, Takustr. 7, 14195 Berlin

Dantzig GB, Wolfe P (1960) Decomposition Principle for Linear Programs. Operations Research 8:101–111

Ernst A, Jiang H, Krishnamoorthy M, DSier (2004) Staff Scheduling and Rostering: A Review of Applications, Methods and Models. European Journal of Operational Research 153(1):3 – 27, DOI http://dx.doi.org/10.1016/S0377-2217(03)00095-X

Goel A, Vidal T (2014) Hours of Service Regulations in Road Freight Transport: An Optimization-Based International Assessment. Transportation Science 48(3):391–412, DOI 10.1287/trsc.2013.0477, URL http://dx.doi.org/10.1287/trsc.2013.0477, http://dx.doi.org/10.1287/trsc.2013.0477

Kanellakis PC, Papadimitriou CH (1980) Local Search for the Asymmetric Traveling Salesman Problem. Operations Research 28(5):1086–1099

Kohl N, Karisch S (2004) Airline Crew Rostering: Problem Types, Modeling, and Optimization. Annals of Operations Research 127(1-4):223–257

Lin S, Kernighan BW (1973) An Effective Heuristic Algorithm for the Traveling Salesman Problem. Operations Research 21(2):498–516

Rangan S, Bowman J, Hauser W, McDonald W, Lewis R, Dongen HV (2013) Integrated Fatigue Modeling in Crew Rostering and Operations. Canadian Aeronautics and Space Journal 59:1–6

Samel A, Wegmann H, Vejvoda M, Drescher J, Gundel D, Manzey D, Wenzel J (1997) Two-Crew Operations: Stress and Fatigue during Long-Haul Night Flights. Aviat Space Environ Med 68:679–687, URL http://elib.dlr.de/27561/, lIDO-Berichtsjahr=1997,

Smith OJ, Boland N, Waterer H (2012) Solving Shortest Path Problems with a Weight Constraint and Replenishment Arcs. Computers and Operations Research 39(5):964–984, DOI 10.1016/j.cor.2011.07.017, URL http://dx.doi.org/10.1016/j.cor.2011.07.017