Faculty of Mathematical Sciences

University of Twente

University for Technical and Social Sciences

P.O. Box 217
7500 AE Enschede
The Netherlands
Phone: +31-53-4893400
Fax: +31-53-4893114
Email: memo@math.utwente.nl

Memorandum No. 1498

# Routing of railway carriages: A case study

P. Brucker,[1] J.L. Hurink and T. Rolfes[2]

September 1999

ISSN 0169-2690

[1]University of Osnabrück
[2]University of Osnabrück

# Routing of Railway Carriages: A Case Study

Peter Brucker
University of Osnabrück

Johann Hurink
University of Twente

Thomas Rolfes*
University of Osnabrück

Revised September 1999

**Abstract**

In the context of organizing timetables for railway companies the following railway carriage routing problem occurs. Given a timetable containing rail links with departure and destination times/stations and the composition of the trains, find a routing of railway carriages such that the required carriages are always available when a train departs. We will present a local search approach for this routing problem for the railway carriages. The approach uses structural properties of an integer multi-commodity network flow formulation of the problem. Computational results for a real world instance are given.

**Key words:** Railway scheduling, local search, multi-commodity flow

**Subject Classification:** 90B35

*Supported by Transport-, Informatik-, und Logistic Consulting GmbH (TCL), Wiesbaden.

# 1 Introduction

Railway companies normally divide the process of making a new timetable into several steps. In the first phase a possible timetable, containing rail links with departure and destination times and stations, and the trains' capacities is developed. Subsequently, the problem of finding a possible implementation of this timetable is considered. In this context, one of the arising subproblems is the question of how the railway carriages should be routed in order to always have the required carriages available when a train departs.

In this paper we will consider this railway carriage routing problem in more detail. A precise definition of the problem is given as follows. A timetable is given which contains a set of passenger train rail links. For each of these rail links the following data are given:

- departure station and destination station,
- departure time and arrival time at the destination, and
- regular composition of the train (type of locomotive, number of first and second class carriages, dining carriage, etc.).

Trains for all passenger train rail links must be available at the departure station in time. This implies that locomotives and railway carriages have to be rerouted from destination stations to departure stations.

The task is usually divided into two subtasks:

1. reroute railway carriages,
2. route locomotives.

There are two possibilities to accomplish the first subtask:

- connect limited numbers of carriages on trains serving the passenger train rail links (these links will be called **existing links**),
- create additional trains with the sole purpose of transporting carriages without passengers (these links will be called **additional links**).

To organize the traffic for short-distance trains the German railway network is divided into several subnetworks (regional areas). A planning team is responsible for scheduling the railway traffic in each subnetwork. As the task of rerouting railway carriages for the short-distance trains in regional areas is dealt with more or less manually, we were asked

to explore the possibility of doing this automatically. The purpose of this paper is to document the results of a corresponding study.

We formulated the rerouting problems as an integer multi-commodity network flow problem with a fixed cost objective function. Such a formulation is given in Section 2. Due to the fact that we failed to solve the problem by integer programming techniques we applied simulated annealing based on structural properties of the multi-commodity flow formulation. In Section 3 appropriate neighborhood structures are derived while Section 4 is devoted to implementations and computational results. Some conclusions can be found in Section 5.

We would like to mention two papers which are of interest. Radtke [1995] discussed a problem related to locomotive routing, while Schrijver [1993] considered the problem of minimizing the number of carriages for an hourly train service of the Nederlandse Spoorwegen (NS) on the Amsterdam-Vlissingen line under the assumption that link demands at the stations (Amsterdam, Rotterdam, Roosendaal, and Vlissingen) are known. The carriages are coupled and decoupled from the regular trains, or kept stationary overnight. For this reason no extra limits for transporting carriages are needed. A two-commodity network flow problem has been solved for the case of two different types of carriages.

# 2 Network Flow Formulations

In Section 2.1 we will formulate the railway carriage routing problem under the condition that there is only one type of railway carriage. This leads to a single-commodity network flow formulation. The generalization of this model to different types of carriages leads to a multi-commodity flow problem which is presented in Section 2.2. The final section deals with additional restrictions relevant to the problem.

## 2.1 The Single-Carriage Case - A Single-Commodity Flow Model

There are $m$ railway stations $i = 1, \ldots, m$ between which we have the passenger train rail links. Associated with each rail link there is a departure and a destination station, as well as a departure and an arrival time. Furthermore, the number of railway carriages needed for this link is known. We are interested in undertaking the routing for a fixed period (e.g. for one day or one week) which will be repeated periodically. Let

$$t_{i1} \leq t_{i2} \leq \ldots \qquad \leq t_{in_i}$$

be the ordered sequence of all departure or arrival times for rail links starting or ending at station $i$. These times define vertices $v_{iv} (i = 1, \ldots, m;\ v = 1, \ldots, n_i)$ of the network

under consideration. Thus, each vertex $v_{iv}$ corresponds with either the departure or the arrival of some train $k$ at station $i$. Let $l(k)$ be the number of carriages coupled to train $k$. If $k$ departs at station $i$ at time $t_{iv}$, we set $b_{iv} = l(k)$ (demand in vertex $v_{iv}$) and if $k$ arrives at station $j$ at time $t_{jv}$, we set $b_{jv} = -l(k)$ (supply in vertex $v_{jv}$). For each station $i$ we introduce two additional vertices $v_{io}$ and $v_{i,n_i+1}$ which indicate the starting time and ending time, respectively, of the period. The use of these vertices will be discussed later. Let $V$ be the set of all these vertices.

The arcs in the network represent the possibilities of rerouting carriages from destination to departure stations. There are three types of arcs.

- Transition arcs $(v_{ir}, v_{i,r+1})$ are used to allow carriages to wait at station $i$ for period $[t_{ir}, t_{i,r+1}]$ if they are needed at this station at a later stage.
- Associated with each existing link and its departure $t_{iv}$ and arrival $t_{js}$ there is an arc $(v_{iv}, v_{js})$ allowing empty carriages to be connected to the train. The flow in these arcs will only correspond to the number of carriages which are additionally moved from station $i$ to station $j$ by the existing link and not to the carriages which belong to the regular composition of the train (the supplies and demands in the vertices corresponding to the departure and arrival of the existing link take care of these carriages).
- Additional arcs $(v_{iv}, v_{js})$ are introduced to establish additional rail links with the purpose of bringing empty carriages from station $i$ to $j$. A flow of value $x > 0$ in such an arc indicates that a locomotive with $x$ carriages without passengers will leave station $i$ after time $t_{iv}$ and arrive at station $j$ before time $t_{js}$ (if the flow in such an arc is 0, nothing will happen). Therefore, such an arc may only be introduced if the times corresponding to the vertices $v_{iv}$ and $v_{js}$ satisfy the condition

$$t_{ir} + d_{ij} \leq t_{js}, \tag{2.1}$$

  where $d_{ij}$ is the travel time which a train needs to travel from station $i$ to $j$.

  To avoid too many arcs of the third type being introduced we restrict this type of arc to those fulfilling the conditions

$$t_{i,r+1} + d_{ij} > t_{js} \tag{2.2}$$

  and

$$t_{ir} + d_{ij} > t_{j,s-1}, \tag{2.3}$$

  i.e. only arcs where the time difference $t_{js} - t_{ir} \geq d_{ij}$ is as small as possible belong to the arc set.

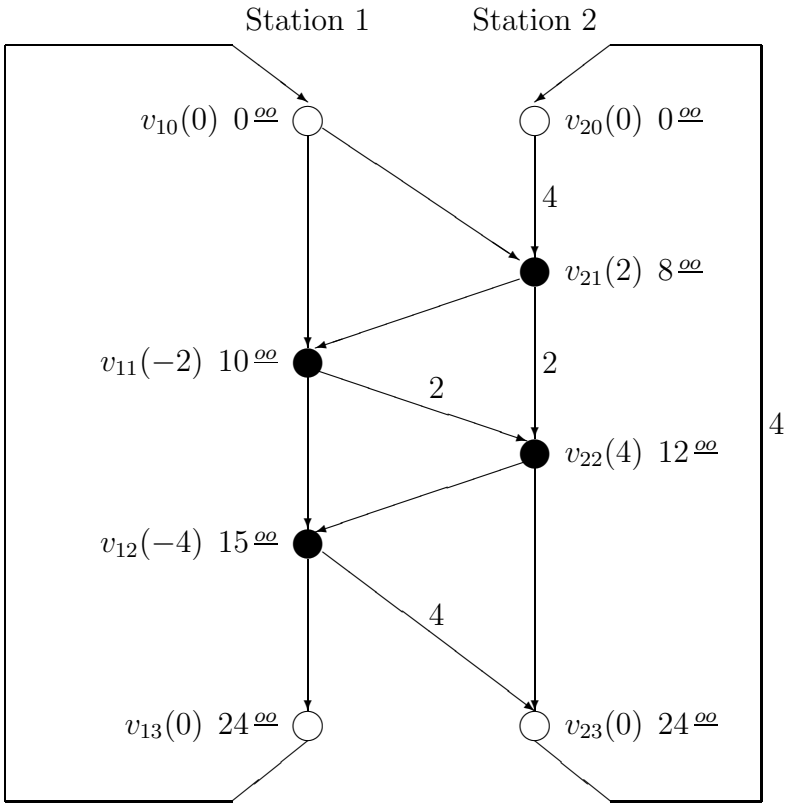


Figure 1: Network with flow

The set of all these arcs is denoted by $E$. Associated with each arc $(v_{ir}, v_{js})$ there is a capacity $u_{ir,js}$. If $i \neq j$, value $u_{ir,js}$ denotes the number of carriages which can be (additionally) coupled to the train; i.e. if the link corresponding to the arc $(v_{ir}, v_{js})$ is an existing rail link, $u_{ir,js}$ denotes the maximal number of carriages which can be coupled to the carriages which already belong to the train on this rail link. If the link corresponding to the arc $(v_{ir}, v_{js})$ is an additional rail link, $u_{ir,js}$ denotes the maximal number of carriages which can be moved on this additional link. If $i = j$ (the arc is of the form $(v_{ir}, v_{i,r+1})$), the value $u_{ir,(i,r+1)}$ denotes the capacity of station $i$. This capacity reflects the fact that due to the equipment of the station only a limited number of carriages can be parked there.

**An Example**

*Figure 1 shows an example with two stations and two trains linking station 2 to station 1. The first train departs at 8:00 from station 2 and arrives at 10:00 at station 1. It is coupled to two carriages. The second train with 4 carriages departs at 12:00 and arrives at 15:00. The planning period extends from 0:00 to 24:00. A journey with empty carriages*

*between these stations takes at least one hour and 30 minutes.*

*The vertices are labeled with nonzero demands/supplies whilst the labels at the arcs indicate nonzero flow values of a feasible solution. In this example, two additional links for carrying empty carriages from station 1 to station 2 are established. Note that the flow can be transformed into a circulation (which describes the routing of the railway carriages) by adding two units of flow to $x_{21,11}$ and four units to $x_{22,12}$ and setting all demands/supplies to zero.*

We have not yet stated how to define the supply/demand for the additional vertices $v_{io}$ and $v_{i,n_i+1}$. The definition of these values depends on the aim of the railway company. There are two possibilities.

- If the company wants to generate a solution for the planning period which can be repeated for the next period (e.g. a daily or a weekly schedule), we have to guarantee that at the end of the period the same number of carriages are at each station as at the beginning of the period. This can be achieved by adding a return arc $(v_{i,n_i+1}, v_{io})$ with infinite capacity and defining $b_{io} = b_{i,n_i+1} = 0$ (see the above example).
- If the company wants to have fixed numbers of carriages at the different stations at the beginning and end of the planning period (these numbers may be different), this can be achieved by assigning these distributions as supply/demand to the additional vertices. More precisely, if at station $i$ we want to have $b_i$ carriages available at the beginning and $b'_i$ carriages at the end of the period, we define $b_{io} = -b_i$ and $b_{i,n_i+1} = b'_i$.

The corresponding network flow problem is

$$\min \sum_{(v_{ir},v_{js})\in E} f_{ir,js}(x_{ir,js}) \tag{2.4}$$

s.t.

$$\sum_{ir:(v_{ir},v_{js})\in E} x_{ir,js} - \sum_{ir:(v_{js},v_{ir})\in E} x_{js,ir} = b_{js} \qquad v_{js} \in V \tag{2.5}$$

$$0 \le x_{ir,js} \le u_{ir,js} \text{ and integer} \qquad (v_{ir}, v_{js}) \in E \tag{2.6}$$

where $x_{ir,js}$ denotes the number of carriages to be additionally transferred from $v_{ir}$ to $v_{js}$. $f_{ir,js}(x)$ are the corresponding costs for transferring $x$ carriages. These costs depend on the type of the arc $(v_{ir}, v_{js})$.

- If $(v_{ir}, v_{js})$ is a transition arc no costs will result from sending a flow through the arc.
- If $(v_{ir}, v_{js})$ is associated with an existing link (denote the set of all these arcs by $E_1$), a positive flow in the arc means that additional carriages are coupled to the regular composition of the train traveling on this link. Thus, no additional fixed costs occur (resulting from the use of a locomotive on the link). However, additional costs (energy, maintenance, etc.) occur depending on the number of carriages coupled to the existing train. We will assume that these costs are linear in the number of carriages and, therefore, that the costs of a flow of $x$ units in the arc may be defined by $W_{ij}x$, where $W_{ij}$ is a constant approximating the additional cost for one additional carriage coupled to a train traveling from station $i$ to station $j$.
- If $(v_{ir}, v_{js})$ is associated with an additional link (denote the set of all these arcs by $E_2$), a positive flow $x$ in the arc means that a new train will be established to bring $x$ carriages from station $i$ to station $j$. Besides the costs relating to the number of carriages $x$ (given by $W_{ij}x$ as in the previous case) fixed costs resulting, e.g. from the use of a locomotive and an engineer, also occur. If we denote these costs by $F_{ij}$, a flow of $x > 0$ units will result in costs $F_{ij} + W_{ij}x$.
- If the goal is to find a periodic schedule, the number of carriages used in the solutions may vary. Thus, in this case it is possible to also put costs on the use of a carriage. This may be achieved by defining the costs of a flow of $x$ units in a return arc $(v_{i,n_i+1}, v_{i0})$ (denote the set of return arcs by $R$) by $Kx$.

Summarizing, the costs $f_{ir,js}(x)$ are defined by

$$f_{ir,js}(x) = \begin{cases} Kx & \text{if } (v_{ir}, v_{js}) \in R \\ W_{ij}x & \text{if } (v_{ir}, v_{js}) \in E_1 \\ F_{ij} + W_{ij}x & \text{if } (v_{ir}, v_{js}) \in E_2 \text{ and } x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

## 2.2 The Multiple-Carriage Case - A Multi-Commodity Flow Model

If the trains are composed of several types of carriages, then for each carriage type $t = 1, \ldots, T$ we have a network flow problem with supply/demands $b^t_{js}$ and flows $x^t_{iv,js}$. The flow balance constraints (2.5) of the different types of carriages are independent of each other. However, the upper bounds (2.6) on the number of carriages which may be coupled

to a train on an existing link or on the number of carriages which may form an additional link are now common bounds over all types of carriages. Furthermore, the fixed costs which occur when an additional link is established have to be considered for all types of carriages simultaneously. Taking into account these restrictions, we get the following multi-commodity network flow problem.

$$\min \quad \sum_{(v_{ir},v_{js})\in E} \sum_{t=0}^{T} f^t_{ir,js}(x^t_{ir,js}) \tag{2.8}$$

$$\text{s.t.} \quad \sum_{ir:(v_{ir},v_{js})\in E} x^t_{ir,js} - \sum_{ir:(v_{js},v_{ir})\in E} x^t_{js,ir} = b^t_{js} \qquad v_{js} \in V; t = 1,\dots,T \tag{2.9}$$

$$x^0_{ir,js} = \sum_{t=1}^{T} x^t_{ir,js} \qquad (v_{ir}, v_{js}) \in E \tag{2.10}$$

$$x^0_{ir,js} \le u_{ir,js} \qquad (v_{ir}, v_{js}) \in E \tag{2.11}$$

$$x^t_{ir,js} \ge 0 \text{ integer} \qquad (v_{ir}, v_{js}) \in E; t = 1,\dots,T. \tag{2.12}$$

In this formulation we have introduced dummy variables $x^0_{ir,js}$, which express the total number of railway carriages to be transferred from $v_{ir}$ to $v_{js}$ (see (2.10)). These variables help to present the cost function $f^t_{ir,js}$ in a compact way:

$$f^t_{ir,js}(x^t_{ir,js}) = \begin{cases} c^t_{ir,js} x^t_{ir,js} & \text{if } t \in \{1,\dots,T\} \\ F_{ij} & \text{if } x^t_{ir,js} > 0,\ t = 0 \\ & \text{and } (v_{ir}, v_{js}) \text{ is an additional link} \\ 0 & \text{else,} \end{cases} \tag{2.13}$$

where

$$c^t_{ir,js} = \begin{cases} K_t & \text{if } ir = (i, n_i + 1) \text{ and } js = io \\ W^t_{ij} & \text{if } i \neq j. \end{cases}$$

In this formulation for an additional link $(v_{ir}, v_{js})$, the value $f^0_{iv,js}(x^0_{iv,js})$ with $x^0_{iv,js} > 0$ represents the fixed costs for running a train with empty carriages.

## 2.3 Additional Features

In the following three subsections we will indicate how some further relevant practical aspects can be dealt with. The first two subsections deal with new situations or restrictions which have not yet been covered, but which can be incorporated into the given

model without changing its structure. In the last subsection we will deal with a different representation of solutions which is useful in practice.

### 2.3.1 Overnight Trains

Until now all additional rail links were scheduled within the intervals $[t_{io}, t_{i,n_i+1}]$, where these intervals typically start at 0:00 and end at 24:00. If we also want to allow overnight rail links (with a travel time less than one day) to move carriages from destination stations to departure stations, we have to add arcs $(v_{ir}, v_{js})$ with $t_{ir} > t_{js}$. Such an arc represents a feasible possibility of moving carriages if the starting time $t_{ir}$ plus the minimal traveling time $d_{ij}$ between the corresponding stations is smaller than the time $t_{js}$ in the next time period, i.e. if $t_{ir} + d_{ij} \leq t_{j,n_{j+1}} + t_{js}$. For the cost function (2.13) we have to define the value $c^t_{ir,js}$ by $K_t + W^t_{ij}$ for arcs $(v_{ir}, v_{js})$ with $i \neq j$ and $t_{ir} > t_{js}$.

### 2.3.2 Turnaround Times at Stations

We assumed that passenger carriages arriving at some time $t$ at a station $i$ can leave $i$ with any train at departure time $t_{ir} > t$. This assumption is not always realistic. If the train departs at a remote track the time difference $t_{ir} - t$ may be too short. Also a passenger carriage could depart with a train at time $t_{ir}$, but not with a train at time $t_{is} > t_{ir}$. To model such a situation we may replace the chain of nodes associated with a station by a bipartite graph $(V_1 \cup V_2, A)$, where $V_1$ represents the arrival nodes and $V_2$ the departure nodes. Furthermore, we have an arc $(v_1, v_2) \in A$ if and only if a carriage arriving at $v_1$ can depart at $v_2$.

### 2.3.3 Routing of Carriages

A solution of the flow models presented in Sections 2.1 and 2.2 only gives limited information to a railway company. They only obtain information on how many carriages they have to use and how many carriages are moved on which rail link. But they do not get routings for individual carriages, which would be very important information since this will form the basis of planning safety checks, maintenance, etc. In the case of fixed distributions of carriages at the beginning and end of the planning period, the company wants to have a path (set of rail links plus transition arcs) for each carriage (better for groups of carriages) from the station where the carriage is at the beginning of the period to a station where it will be at the end of the period. In the case of a periodical schedule (daily, weekly, etc.) the company will want to have a cycle for each carriage (group of carriages) from the station where the carriage is at the beginning of the period back to this station. In the latter case the cycle may contain several periods (e.g. several return

arcs $(v_{i,n_i+1}, v_{i0})$).

To achieve such a collection of paths or cycles, we have to decompose a given flow solution into path and cycles. However, since not all movements of the carriages are represented by flows (the movement of carriages which form the regular composition of a passenger train rail link is not part of the flow!), we will first incorporate these movements into the flow solution.

1. For each regular rail link departing with $b^t_{ir}$ carriages of type $t$ at station $i$ at time $t_{ir}$ and arriving at time $t_{js}$ at station $j$ (i.e. $b^t_{js} = -b^t_{ir}$), we increment the flow $x^t_{ir,js}$ in $(v_{ir}, v_{js})$ by $b^t_{ir}$ units. Furthermore, we eliminate demand $b^t_{ir}$ and supply $b^t_{js}$. (These changes have already been mentioned in connection with the example in Section 2.1.)

2. Decompose the resulting flow solution into flows on paths and in cycles (see Ahuja et al. [1993], Section 3.5).

Usually such a decomposition is not unique. Thus, we have a certain amount of freedom to construct the circulations, which can be used to find 'good' circulations of groups of railway carriages.

# 3 Local Search Approach

In this section we will present a local search approach to calculate routings for railway carriages. The method will be based on the multi-commodity flow formulation (2.8)-(2.13) presented in the previous section. Firstly, in Subsection 3.1 we will define appropriate neighborhoods and, afterwards, in Subsection 3.2 we will present a two-phase local search method based on simulated annealing. For the sake of easy notation we will describe the methods based on the following formulation of the multi-commodity flow problem with fixed costs:

$$\min \quad \sum_{(i,j)\in E} \sum_{t=0}^{T} f^t_{ij}(x^t_{ij}) \tag{3.1}$$

$$s.t. \quad \sum_{i:(i,j)\in E} x^t_{ij} - \sum_{i:(j,i)\in E} x^t_{ji} = b^t_j \qquad j \in V; t = 1, \ldots, T \tag{3.2}$$

$$x^0_{ij} = \sum_{t=1}^{T} x^t_{ij} \qquad (i,j) \in E \tag{3.3}$$

$$x^0_{ij} \leq u_{ij} \qquad (i,j) \in E \tag{3.4}$$

$$x^t_{ij} \geq 0 \text{ and integer} \qquad (i,j) \in E; t = 1, \ldots, T \tag{3.5}$$

with

$$f_{ij}^t(x_{ij}^t) = \begin{cases} c_{ij}^t x_{ij}^t & \text{if } t \in \{1, \ldots, T\} \\ F_{ij} & \text{if } x_{ij}^t > 0 \text{ and } t = 0 \\ 0 & \text{if } x_{ij}^t = 0 \text{ and } t = 0. \end{cases} \tag{3.6}$$

We assume that all data are integers and that the fixed parts of the costs, $F_{ij}$, are non-negative. Obviously, the above formulation is a reformulation of (2.8)-(2.13).

## 3.1 Neighborhood Structures

The integer multi-commodity flow problem with fixed costs (3.1) to (3.6) reduces to a classical min-cost flow problem if only one commodity is considered (i.e. $T = 1$) and no fixed costs are involved (i.e. $F_{ij} = 0$). Therefore, for the definition of appropriate neighborhoods for problem (3.1) to (3.6) we will use ideas from the network simplex method for min-cost flow problems (for a description of the network simplex method see, e.g. Chvatal [1983]). The network simplex method considers spanning tree solutions and moves from a given spanning tree solution to the next by inserting one nontree arc into the tree and deleting one tree arc. To adapt these concepts to problem (3.1) to (3.6) we will generalize the corresponding definitions and techniques.

**Definition 3.1** *For a solution $x = (x^1, \ldots, x^T)$ of (3.1) to (3.6) an arc $(i,j) \in E$ is called* **fixed** *with respect to (w.r.t.) $x^t$ if either the flow of commodity $t$ in the arc $(i,j)$ is 0 or the total flow in the arc $(i,j)$ is equal to the upper bound of this arc; i.e. if either $x_{ij}^t = 0$ or $x_{ij}^0 = u_{ij}$. Otherwise, the arc $(i,j)$ is called* **free** *w.r.t. $x^t$.*

*A solution $x$ is called* **cycle-free** *if for $t = 1, \ldots, T$ the set of free arcs w.r.t. $x_t$ contains no (undirected) circle.*

Cycle-free solutions play an important role since, according to the next theorem, problem (3.1) to (3.6) always has an optimal solution which is cycle-free.

**Theorem 3.2** *An arbitrary solution $x$ of problem (3.1) to (3.6) can be transformed into a cycle-free solution $\overline{x}$ without increasing the objective value.*

**Proof**: Let $x = (x^1, \ldots, x^T)$. To prove the theorem it is sufficient to show that a solution $\overline{x}(t-1) = (\overline{x}^1, \ldots, \overline{x}^{t-1}, x^t, \ldots, x^T)$, where $\overline{x}^1, \ldots, \overline{x}^{t-1}$ do not contain circles of only free arcs, can be transformed into a solution $\overline{x}(t) = (\overline{x}^1, \ldots, \overline{x}^t, x^{t+1}, \ldots, x^T)$, where $\overline{x}^1, \ldots, \overline{x}^t$ do not contain circles of only free arcs.

If $x^t$ contains no circle of free arcs w.r.t. $x^t$, we have finished. Thus, assume that $C$ is a circle of free arcs w.r.t. $x^t$. Let $j_1, \ldots, j_k$ be the arcs in $C$. Since the arcs in $C$ are free

we have

$$x^t_{j_l} > 0 \text{ and } x^0_{j_l} < u_{j_l};\ l = 1, \ldots, k. \tag{3.7}$$

Furthermore, let $E_1(C)$ be the set of arcs in $C$ which are directed in the same way as $j_1$ is directed in $C$, and let $E_2(C)$ be the remaining arcs in $C$.

We now consider the following solution $\overline{x}^t(\Delta)$ for commodity $t$ which results from $x^t$ by a change in $C$ only:

$$\overline{x}^t_k(\Delta) = \begin{cases} x^t_k + \Delta & \text{if } k \in E_1(C) \\ x^t_k - \Delta & \text{if } k \in E_2(C) \\ x^t_k & \text{else.} \end{cases} \tag{3.8}$$

If we choose $|\Delta|$ sufficiently small (no arcs of $C$ are fixed by the change of $\Delta$), the solution $\overline{x}^t(\Delta)$ is also feasible due to (3.7) and the change in the objective value is linear in $\Delta$ (only the linear parts of the objective function play a role if we choose $|\Delta|$ sufficiently small). Thus, for either small positive or small negative values of $\Delta$ the solution $\overline{x}^t(\Delta)$ is at least as good as $x^t$. Assume w.l.o.g. that this is the case for positive values of $\Delta$. We now choose $\Delta$ as the smallest possible positive value for which in $\overline{x}^t(\Delta)$ at least one of the arcs in $C$ becomes fixed. Since the fixed parts of the costs are assumed to be non-negative, this change does not lead to an increase of the costs. Thus, $\overline{x}^t(\Delta)$ is at least as good as $x^t$ and in $\overline{x}^t(\Delta)$ circle $C$ no longer contains only free arcs. Repeating this process yields a solution $\overline{x}^t$ which is at least as good as $x^t$ and which contains no circle of free arcs w.r.t. $\overline{x}^t$ only.

It remains to show that the above described change for the flow of commodity $t$ (replacing $x^t$ by $\overline{x}^t$) did not produce a circle of only free arcs for one of the flows $\overline{x}^1, \ldots, \overline{x}^{t-1}$ for the commodities $1, \ldots, t-1$. However, since in the above change only free arcs were involved and since only the solution for commodity $t$ was changed, no fixed arc w.r.t. some $\overline{x}^l;\ 1 \leq l \leq t-1$ will become free w.r.t. $\overline{x}^l$ and, therefore, no new circle of only free arcs may occur. □

Based on the above theorem, we will consider only cycle-free solutions for our local search approach. To define neighborhoods we first have to find a good representation for such solutions.

Considering a partial solution $x^i$ of a cycle-free solution $x$, we observe that the free arcs w.r.t. $x^i$ form a forest in the graph $G = (V, E)$. Extending this forest by some fixed arcs w.r.t. $x^i$ we get a spanning tree $S(x^i)$ with the property that all arcs not in $S(x^i)$ are fixed w.r.t. $x^i$. On the other hand, if we have a spanning tree of $G$ and given values $x^i_l$ for all nontree arcs $l$, then values $x^i_l$ for all tree arcs $l$ satisfying (3.2) are uniquely determined. Thus, in the following we will use spanning trees to describe solutions.

**Definition 3.3** *A* **spanning tree solution** *of (3.1) to (3.6) is given by a set $S^1, \ldots, S^T$ of spanning trees of $G$ and by given values $x_l^t$ for all nontree arcs $l$ of $S^t$; $t = 1, \ldots, T$. The unique solution $x = (x^1, \ldots, x^T)$ which fulfills all the constraints (3.2) is called the* **extension** *of the spanning tree solution. If this extension also fulfills constraints (3.3) to (3.5) and if all nontree arcs $l$ of $S^i$ are fixed w.r.t. $x^t$; $t = 1, \ldots, T$, the solution is called a* **feasible spanning tree solution**.

Obviously, each spanning tree solution is cycle-free. Furthermore, as indicated above, for each cycle-free solution $x = (x^1, \ldots, x^T)$ we can find corresponding spanning trees $S^1(x_1), \ldots, S^T(x_T)$ such that the resulting spanning tree solution is given by $x$. Thus, we can reduce the search space for our local search method to the set of all feasible spanning tree solutions. It remains to define a neighborhood structure on this set.

Analogous to the network simplex method, we will define neighbors by inserting and deleting arcs in the spanning trees. Let $S = (S^1, \ldots, S^T)$ be a set of spanning trees and let $x(S)$ be the corresponding feasible spanning tree solution. The set of operators which may be applied to $S$ is given by

$$OP(S) = \{insert_{kl}^t | t = 1, \ldots, T;\ (k,l) \notin S^t\}.$$

The application of an operator $insert_{kl}^t \in OP(S)$ to the spanning tree solution $x(S)$ is defined as follows: The arc $(k,l)$ will be inserted into the spanning tree $S^t$ yielding a circle $C$ in $S^t$. To determine the arc leaving $S^t$, the flow in the circle $C$ is changed in a similar fashion to (3.8). We define the set $E_1(C)$ as the set of arcs directed in the same way as $(k,l)$ in $C$ and $E_2(C)$ as the remaining arcs in $C$. If we have $x(S)_{kl}^t = 0$ we choose $\Delta$ as the smallest possible positive value such that in the resulting solution at least one arc of $C$ becomes fixed w.r.t. $x(S)^t$. Otherwise, if $x(S)_{kl}^0 = u_{kl}$ we choose $\Delta$ as the largest possible negative value such that in the resulting solution at least one arc of $C$ becomes fixed w.r.t. $x(S)^t$. The arc in $C$ which will become fixed is the arc leaving $S^t$ (if more than one arc becomes fixed, we may choose one of these arcs arbitrarily). Obviously, for commodity $t$ the set of free arcs still forms a tree or a forest. However, it may happen that by the above process the flow $x_{ij}^t$ in an arc $(i,j)$ of $C$ with $x(S)_{ij}^0 = u_{ij}$ is reduced ($S^t$ can contain fixed arcs!) and thus the arc $(i,j)$ swaps from being fixed to free for the other commodities $k \neq t$. Therefore, we can not guarantee that the resulting solution is a spanning tree solution. However, if this happens we may transform this solution into a cycle-free solution by the method described in Theorem 3.2. Since this elimination procedure is rather costly, in practice we will execute this 'repair' only when commodity $k$ is considered next, i.e. not before an operator on commodity $k$ is applied.

Obviously, the above process transforms a feasible spanning tree solution into a spanning tree solution which fulfills conditions (3.2) - (3.4) and the non-negativity constraints. It remains to show that the integral constraints from (3.5) also remain valid. For the single-commodity case ($T = 1$) this is obvious, since for all extensions of spanning tree solutions,

where the nontree arcs are fixed, the integrality constraints are satisfied (note that the nontree arcs can take only values 0 and $u_{ij}$ and, therefore, the values resulting from the extension must be integral, too). Thus, for the single-commodity case we can ignore the integrality constraints. For the real multi-commodity case ($T > 1$) the nontree arcs may already take nonintegral values and, thus, also nonintegral extensions of spanning tree solutions may exist. Since easy examples can be found where nonintegral solutions to (3.1) - (3.6) exist which are better than each integral solution, we have to take care of the integrality constraints. However, if we start our neighborhood search with a feasible (and thus integral) spanning tree solution, we never lose the integrality since the calculated $\Delta$ values within a neighborhood step (including the circle elimination) are always integral.

## 3.2 Two-Phase Local Search Method

Based on the neighborhood described in the previous subsection a direct application of standard local search methods like simulated annealing or tabu search is possible. However, due to the large number of possible neighborhood operators (for each combination of a commodity and a corresponding nontree arc we have one operator) this approach will not be efficient. We decided to apply a two-level local search approach. On the first level a choice for one of the commodities is made. On the second level for the commodity chosen on level 1 a series of steps with operators defined on this commodity is applied. More precisely, we execute a fixed number of iterations of a simulated annealing based local search approach. Afterwards, we switch back to level 1, change the choice for the commodity, and continue the search with this new commodity. This process is repeated until some stop condition is fulfilled. Summarizing, we execute a series of local search applications to single commodity problems.

Besides a structural effect, the two-level approach also has a computational effect. For the time we focus on one commodity, we only have to keep in mind the data structure belonging to this commodity. The effect of the changes for this commodity on the other commodities will be calculated by the time these commodities will be fixed. Thus, especially the costly elimination of circles with only free arcs (such circles may occur due to the influence on other commodities) has to be executed only seldom.

The above sketched two-level approach is the basis for a two-phase local search method. In the first phase we try to cover a large part of the search space and to identify a good solution. Afterwards, in the second phase, we start with the best solution found in the first phase and try to improve this solution. Both phases use the presented two-level approach. They are applied several times (multiple restarts).

In the following we will describe in more detail how we calculated an initial solution, how we realized the two phases, and how these elements are combined to an overall local search approach.

- **Initial solution**
  The structure of the initial solution has an important influence on the behavior of the local search approach. This follows from the fact that our local search approach has problems to remove additional links if more than one commodity is involved in this link. In this case we can only get rid of the additional link (and, thus, the fixed costs) if the flows for all commodities in this link become 0. However, we do not get a large profit in any of the single-commodity problems if we only reduce this flow to 0, since the positive flow values for the other commodities still produce the fixed costs for this arc.

  Based on this observation, we calculated the initial solution such that only very few new additional links were used. To achieve this we allowed the use of a larger number of railway carriages even if this led to large costs. More precisely, we assume that we have an arbitrarily large amount of carriages available at each station and for each train departing at a station we either use a carriage which arrived earlier at this station or we take one of the carriages available at this station. Thus, we only need to establish additional links in the case where different numbers of carriages are available at the beginning and end of the period at some stations. The creation of these links is done in a greedy way. This induces that only additional links which are inevitable due to the given data of the instance are installed (i.e. if in a station the number of arriving carriages is smaller than the number of departing carriages). Reducing the number of carriages is much easier for our local search approach, since reducing the number of carriages is mainly a single-commodity problem.

- **Phase 1**
  In this phase we randomly select the commodities on level 1 according to their importance. More precisely, commodity $t$ is fixed with probability $B(t)/B$, where $B(t)$ is the total demand of commodity $t$ (sum of all positive $b_j^t$ values) and $B$ is the total demand over all commodities. However, if a commodity has not been fixed the last $2T$ times, it will be fixed within the next iteration. On the second level of the local search approach we apply a special version of simulated annealing to the single-commodity problem corresponding to the commodity fixed on level one. This special version consists of a fixed number of iterations (the used number is given in the next section) with a fixed cooling parameter (for a description of simulated annealing and its acceptance mechanism, see van Laarhoven and Aarts [1987] or Dowsland [1993]). The cooling parameter is experimentally chosen in such a way that it is unlikely to open a new additional link (i.e. to use an arc $(i, j)$ with $x_{ij}^0 = 0$ and $F_{ij} > 0$) without deleting some other additional link. More precisely, the probability that a decrease of more than $\min F_{ij}$ is accepted is below 0.01%.

  We have chosen a constant cooling parameter instead of a decreasing cooling parameter (which is standard) since in the latter case the search would end in some

good local optimum for one commodity. However, practical tests have shown that this often makes it difficult to get further improvements for the other commodities (the 'optimized' structure for one commodity allows no room for good changes for the other commodities). For the same reason, we started the local search for a new commodity with the last solution found for the previous commodity rather than the best solution.

These two variations may help to ensure that the search does not get stuck in some small region of the search space, but occupies larger areas of it (diversification).

Phase 1 is terminated after a fixed number of changes of the commodity on the first level (the concrete number is given in the next section).

- **Phase 2**
  In this phase we select on level 1 the commodities in a fixed and circular order which has been prespecified in a random way. The phase stops if we subsequently have applied local search to all commodities without improving the current best solution. Concerning the single commodity problems we apply a fixed number of iterations of simulated annealing with a linear decreasing cooling parameter. Already the initial value of the cooling parameter is small to ensure that it is not very likely that a new additional link is opened. The number of executed iterations depends on the importance of the given commodity (it is linear in the total demand $B(t)$ and a detailed description is given in the next section). Furthermore, the search for the next commodity starts with the best solution found for the previous commodity. The reason for these differences to the first phase is that we now intensively want to search for good solutions starting from the best solution found in the first phase (intensification).

- **Overall local search approach**
  For the overall approach we will apply the two phases several times always using the same initial solution. Computational experiments have shown that it is not necessary to calculate different initial solutions for the restarts. This may result from the randomized character of the basic local search method used (simulated annealing) and the structure of the initial solution (generous use of carriages). Therefore, the different restarts seem to have enough potential to occupy different regions of the search space.

# 4 Computational Results

In this section we report on some results of a computational study achieved with the developed local search approach. The aim of the study was to provide some idea of which

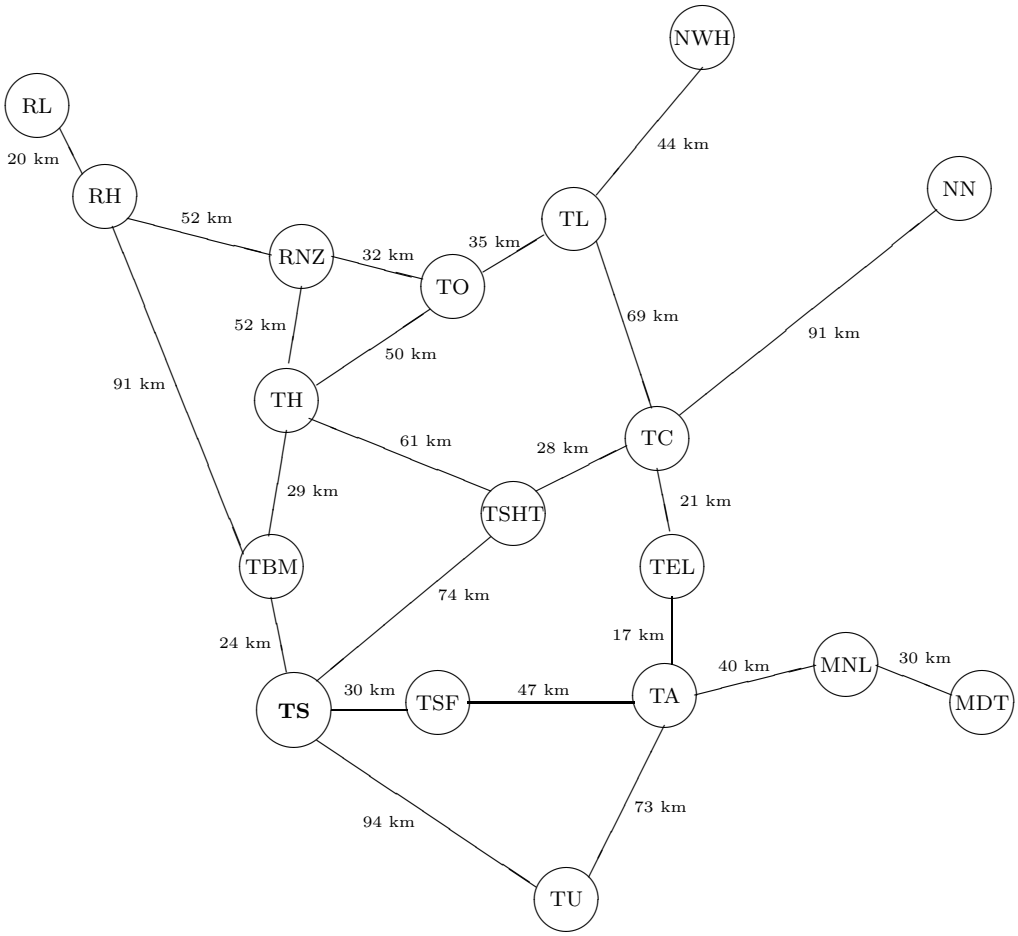


Figure 2: Real world instance

type of solutions we were able to generate with our method and how long it takes to generate them. To achieve this we constructed three scenarios using an instance which was generated on the basis of the winter term 1996/97 schedule for the regional area 'Würtemberg (Germany)'. The three scenarios only differ in the way in which the costs are chosen. One scenario focuses on the goal of minimizing the number of used carriages, the second on minimizing the costs for additional links and the third forms a compromise between these two goals (note that only the ratio between the fixed costs and the linear costs are of importance, rather than the exact values for the costs). Using the three scenarios, solutions of different structures can be generated for the given instance.

For the instance in the regional area 'Würtemberg' we have 18 relevant stations (stations where trains arrive or depart), approximately 200 passenger train links, and 6 different types of railway carriages. The railway stations, possible connections between them, and the distances are given in Figure 2. Transforming this instance into an instance of our

integer multi-commodity flow problem leads to a network with 440 vertices and 958 arcs. From these arcs 371 correspond to possible additional train links. The total demand for the different types of railway carriages varies from 12 to 361.

Based on this network, we generated three test instances by assigning different costs to the arcs. More precisely, we varied the ratio between the costs for using carriages and the costs for additional links. With the first instance $I_1$ we mainly tried to minimize the number of carriages used by assigning relatively high costs for the use of a carriage. The second instance $I_2$ focuses on minimizing the costs of additional train links by assigning only small costs for the use of carriages. The last instance will form a compromise such that for two carriage types the costs of using these carriages are smaller, whereas for the four remaining types the costs are again high.

The computational tests were executed on a PC 486 DX with 33 MHz and the presented methods were coded using the programming language $C$. Before describing these computational results, we shortly give some details on how the described components of the local search method were built together. In phase 1 a new commodity on level 1 is chosen $n_1^{ph1}$ times. For each fixed commodity $n_2^{ph1}$ iterations of simulated annealing with a constant cooling parameter are executed on level 2. This results in a total of $n_1^{ph1} \cdot n_2^{ph1}$ iterations of simulated annealing in phase 1. In phase 2 we can not prespecify the number of times we change the commodity on level 1 since we will only stop if the current best solution has not been improved in $T$ subsequent attempts. The number $n_2^{ph2}$ of iterations which are executed on level 2 for a fixed commodity is given by $n_2^{ph2} = c_1 + c_2 \cdot B(t)$, where $c_1$ is a constant much larger than $n_2^{ph1}$. Thus, for a fixed commodity the number of iterations in phase 2 is always much larger than in phase 1.

The structure of the algorithm we used to achieve the test results can be summarized as follows:

**Algorithm** *Local Search*

**FOR** $i = 1$ **TO** $num\_restarts$ **DO**
**BEGIN**
phase_1($n_1^{ph1}, n_2^{ph1}$);
phase_2($n_2^{ph2}$);
**END**.

In a series of tests we tried to determine good values for the parameters of our local search approach (details of these tests can be found in Rolfes [1998]). For phase 1 it appeared to be good to execute around 25,000 iterations per restart. Reducing this value made the results worse. Increasing this value led to better results. However, it reduced the chance of obtaining improvements in phase 2. Putting both phases together, a higher number of

| | additional links | | used carriages | | | | | |
|---|---|---|---|---|---|---|---|---|
| | # | km | Type 1 | Type 2 | Type 3 | Type 4 | Type 5 | Type 6 |
| practice | 12 | 495 | 99 | 19 | 11 | 29 | 6 | 4 |
| $I_1$-average | 13 | 606 | 93 | 19 | 11 | 28 | 6 | 4 |
| $I_1$-best | 11 | 526 | 93 | 19 | 11 | 28 | 6 | 4 |
| $I_2$-average | 4 | 151 | 99 | 22 | 12 | 29 | 6 | 4 |
| $I_2$-best | 4 | 134 | 99 | 22 | 12 | 29 | 6 | 4 |
| $I_3$-average | 6 | 225 | 99 | 19 | 11 | 29 | 6 | 4 |
| $I_3$-good | 5 | 155 | 99 | 19 | 11 | 29 | 6 | 4 |
| $I_3$-best | 6 | 207 | 98 | 19 | 11 | 28 | 6 | 4 |

Table 1: Computational Results

iterations in phase 1 did not improve the overall quality significantly, but increased the computational times. The 25,000 iterations in phase 1 were divided over 50 runs for fixed commodities (i.e. $n_1^{ph1} = 50$) each with 500 iterations (i.e. $n_2^{ph1} = 500$). Reducing the value for $n_1^{ph1}$ made the results worse and increasing this value increased the computational times too much (resulting from the necessary repair work after a change of commodity). Using this constellation, phase 1 took approximately one minute of computational time for our test instance.

For phase 2 we determined the number of iterations for a fixed commodity by 2000 plus 5 times the total demand of the fixed commodity (i.e. $c_1 = 2000$ and $c_2 = 5$). Increasing these values will lead to a slight increase of the quality, but also to a lengthening of the computational times. The tests indicate that it is better to use this time for additional restarts. In combination with the above-mentioned setting for phase 1 one execution of phase 2 also takes approximately one minute.

Using the mentioned values for the parameters, it arose that we always obtained good solutions after at most 10 restarts. To indicate the quality of the achieved results we will compare our solutions with the solution used in practice. The main characteristics of the practical solution and our solutions are reported in Table 1. The table contains the following information: For each solution we give the number of additional links used per day (#) and the total length of these additional links (km). Furthermore, the number of carriages used of each type is given.

In Table 1 for each instance an average ($I_j$-average) and the best found solution ($I_j$-best) is given. The average solution describes an outcome whose quality was achieved after 10 restarts by almost all runs of our local search method with the above-mentioned parameters (due to the random structure of the method we made several runs with the same parameter constellation). The best solutions correspond to the best solutions we found during the whole tests. For instance $I_2$ these solutions were found with the above-

mentioned parameter setting in many runs. However, for instance $I_1$ this solution was only found in a more time-consuming run (with $n_1^{ph1} = 500$ in the first phase). For $I_3$ the best solution was found more easily than for $I_1$, but slightly harder than for $I_2$ (i.e. it was found several times and also by different parameter settings, but not as often as for $I_2$). For instance $I_3$ the solution $I_3$-good represents an outcome which has often been achieved with the above-mentioned parameter setting.

The results for instance $I_1$ show that it is possible to reduce the number of carriages used for the practice solution by 7 using a similar number of additional links and without increasing the distance for additional links considerably (only 31 km in the best solution). On the other hand, instance $I_2$ gives us solutions where we can reduce the number of additional links from 12 to 4 and the distance of these links from 495 km to approximately 150 km. To realize these solutions we have to use 4 more carriages than for the practice solution. Finally, for instance $I_3$ we get solutions which dominate the solution from practice in all aspects. Using the same amount of carriages, we can reduce the number of additional links from 12 to 5 and the distances of these links from 495 km to 155 km (solution $I_3$-good) or using 2 carriages fewer we can reduce the number of additional links from 12 to 6 and the distances of these links from 495 km to 207 km (solution $I_3$-best).

Summarizing, we can state that our local search approach gave excellent results for the considered real world instance within 20 minutes on a slow PC. Using the restarts, the method is very stable since the quality of the results achieved after at most 10 restarts is always very similar. The calculated solutions will reduce the costs for operating a given timetable considerably.

# 5 Conclusions

In connection with a case study we have formulated the problem of routing railway carriages for a given timetable as an integer multi-commodity flow problem with linear fixed-cost objective functions and solved this problem by a simulated annealing procedure. Test results for short-distance connections in a subnetwork of the German railway network have shown that the method is very promising. Besides calculating good routings it can be used to calculate the influence of investment into additional carriages. However, long-term planning was not the topic of the paper.

The method can also be used to solve the single-commodity network flow problem with linear fixed-cost objective functions, which is known to be $NP$-hard. Thus, it would be an alternative to a branch-and-bound algorithm developed by Arlt [1994] for this special problem. The general method can also be used to solve network design problems. Another approach to solve the problem is to apply mathematical programming methods to solve the integer multi-commodity network flow problem. For a multi-commodity flow

problem with binary variables and linear objective function Barnhart et al. [1996] have developed a method based on branch-and-bound and column generation. They provide approximative solutions for problems with up to 50 nodes and 130 arcs, but a large number of commodities. It remains to determine how these methods, if extended to the more general situation, are able to compete with the simulated annealing approach.

# References


**R.K. Ahuja, T.L. Magnanti, J.B. Orlin [1993]** Network flows; theory, algorithms, and applications, Prentice Hall, New Jersey.

**Ch. Arlt [1994]** Die Lösung großer Fixkosten-Netzwerkflußprobleme, Gabler Verlag, Wiesbaden.

**C. Barnhart, C.A. Hane, P.H. Vance [1996]** Integer multicommodity flow problems, Working paper, Massachusetts Institute of Technology, Center for Transportation Studies.

**V. Chvatal** Linear Programming, W.H. Freeman and Company, New York/ San Francisco.

**K.A. Dowsland [1993]** Simulated annealing, in: Modern Heuristic Techniques for Combinatorial Problems, edited by C. Reeves, Blackwell Scientific Publications, London, 20-69.

**P.J.M. van Laarhoven, E.H.L. Aarts [1987]** Simulated annealing: theory and applications, D. Reidel Publishing Company, Dordrecht, Holland.

**A. Radtke [1995]** Dispositionsmodell für den optimierten Betriebsmitteleinsatz der Eisenbahn, Wissenschaftliche Arbeiten Nr. 43, Fachbereich Bauingenieur- und Vermessungswesen; Institut für Verkehrswesen, Eisenbahnbau und Betrieb, Universität Hannover.

**T. Rolfes [1998]** Ein lokales Suchverfahren für die Fahrzeugeinsatzplanung im Schienenpersonenverkehr, Diplomarbeit, Fachbereich Mathematik/Informatik, Universität Osnabrück.

**A. Schrijver [1993]** Minimum circulation of railway stock, CWI Quarterly 6(3), 205-217.