A branch-anc-cut approach for solving line planning problems

TRANSPORTATION SCIENCE
Vol. 38, No. 3, August 2004, pp. 379–393
ISSN 0041-1655 | EISSN 1526-5447 | 04 | 3803 | 0379

DOI 10.1287/trsc.1030.0051

A Branch-and-Cut Approach for Solving Railway Line-Planning Problems

Jan-Willem Goossens, Stan van Hoesel
Department of Quantitative Economics, University of Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands {j.goossens@ke.unimaas.nl, s.vanhoesel@ke.unimaas.nl}

Leo Kroon
Rotterdam School of Management, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands, l.kroon@fbk.eur.nl

An important strategic phase in the planning process of a railway operator is the development of a line plan, i.e., a set of routes (paths) in a network of tracks, operated at a given hourly frequency. We consider a model formulation of the line-planning problem where total operating costs are to be minimized. This model is solved with a branch-and-cut approach, for which we develop a variety of valid inequalities and reduction methods. A computational study of five real-life instances based on examples from Netherlands Railways (NS) is included.

Key words: rail transportation; integer programming; branch and cut; combinatorial optimization
History: Received: September 2001; revision received: September 2002; accepted: September 2002.

Introduction

The planning problem faced by every railway operator consists of several consecutive stages, ranging from strategic decisions concerning, e.g., infrastructure development, to real-time traffic control. An overview of these stages is given in Figure 1(a). Strategic problems are driven by estimates for the long-term demand. The first problem concerns the determination of the infrastructure, such as railway tracks and stations. Both the infrastructure and demand data are input for the line-planning problem (LPP) considered in this paper. It involves the selection of paths in the railway network on which train connections are maintained. Thus, LPP focuses on determining a subset of all possible lines that together make up the line plan. Successive decision stages are the more detailed planning problems such as the construction of timetables (Schrijver and Steenbeek 1994, Odijk 1997, Nachtigall 1999, Lindner 2000), traffic planning (route assignment, platform assignment) (Zwaneveld 1997), rolling stock planning (Schrijver 1993), personnel planning (Caprara et al. 1997), and shunting planning (Gallo and Di Miele 2001).

Besides the operated paths, a line plan also specifies the hourly frequencies of the lines. Traditionally, the objective when constructing a line plan has been to find a set of lines that maximizes the number of direct travelers, i.e., the number of travelers that do not have to change trains during their journey; cf. Bussieck et al. (1996), Bussieck (1998). This is an obvious objective from a service perspective. However, this objective tends to generate geographically long train lines. Because the passenger flows usually differ substantially between regions, long train lines often result in an inefficient use of rolling stock. As an alternative objective, similar to Claessens et al. (1998) and Bussieck (1998), this paper will focus on the problem of minimizing the operational costs of a line plan. We will refer to this problem as the Cost-Minimizing Line-Planning Problem, or CLPP.

There are several approaches for formulating CLPP. In Claessens et al. (1998), an integer nonlinear programming model is presented as a natural model for CLPP. They transform this model into a linear programming problem on binary variables (BLP), which they then try to solve using branch and bound. In Bussieck (1998) an alternative formulation using general integer variables (ILP) is presented. In recent work of Bussieck et al. (2002), the original nonlinear model is reconsidered. They study a relaxation of the nonlinear formulation to obtain good primal solutions of the original problem.

Besides presenting the ILP formulation, Bussieck (1998) also compares the BLP and ILP formulations using a cutting-plane algorithm. They conclude that the more compact ILP formulation is preferable for generating good feasible solutions, compared to the large BLP formulation. However, the lower bounds provided by BLP are superior to the lower bounds of ILP, therefore the BLP formulation is preferable for proving optimality of a feasible solution. By extending the reduction methods and classes of cutting planes presented in both papers, we show that the BLP formulation can be used to solve large instances

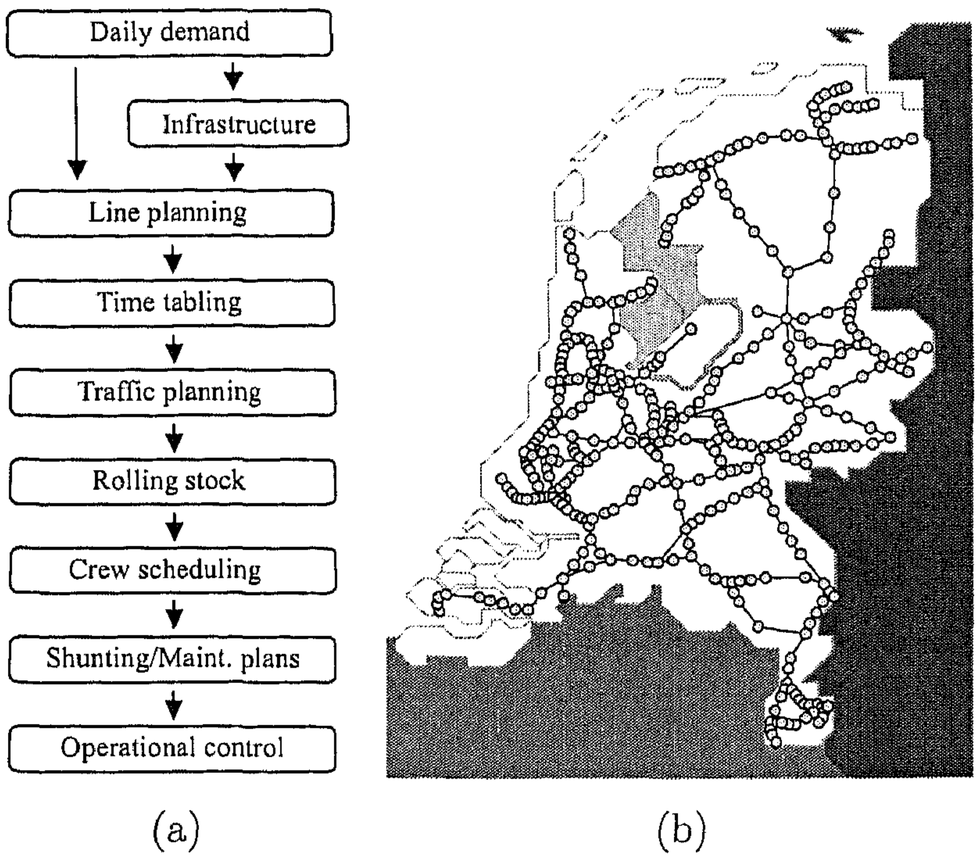


Figure 1 **The Different Stages in the Planning Process (1(a)) and the Dutch Railway Network (1(b))**

of the problem using branch and cut, or at least to obtain excellent upper bounds and lower bounds in reasonable time.

In the next section the model formulation is presented. Section 2 describes the solution methodology. Section 3 describes the implementational issues, and in §4 we describe a computational study based on instances of the Dutch railway operator NS (Nederlandse Spoorwegen).

1. Model Formulation

Fundamental in the modeling of the line-planning problem is the concept of a line. In railway terminology, a line specifies a route between an origin and a destination station and the subsequent stops, combined with the operated hourly frequency. A line plan is the set of operated lines. The line plan does not incorporate the exact timetable for the operated lines, though we assume that the timetable is cyclic with a cycle time of one hour, i.e., that the line plan is repeated every hour. This reflects the situation in the Netherlands, as well as in many other European countries. The model described here focuses on finding a line plan that minimizes the induced operational costs. The problem of finding a cost-minimizing line plan can be described as follows:

Given the railway infrastructure with the accompanying stations and the number of travelers between stations, determine a cost-minimizing line plan.

Personnel and rolling stock are the main cost factors that determine the operational costs of a line plan. Both the number of trains needed to operate the lines at given frequencies, as well as the number of conductors and drivers needed to operate them, depend on the circulation of the rolling stock. Following the practice at NS, we assume a circulation schedule in which all rolling stock is dedicated to specific lines. Thus, the switching of rolling stock between lines is kept to a minimum. The trains used to operate a single line are assumed to be identical, i.e., pulling the same number of carriages. Now, using the hourly frequency and the number of carriages per train, we can determine the operational costs that can be attributed to operating a specific line.

EXAMPLE 1.1. The required number of compositions (trains) for the trivial rolling-stock circulation is calculated using the total time needed to operate the line in both directions and the time needed at both end stations to prepare the train for the reverse journey. Consider a line with a travel time of 60 minutes from its origin to its destination station. Fifteen minutes are needed at either end station to prepare the train for the reverse journey. This amounts to a total cycle time of 150 minutes. If our line is operated once per hour, this would require $\lceil 150/60 \rceil = 3$ compositions (Figure 2(a)). However, operating our line at a frequency of two (Figure 2(b)), departing every 30 minutes, will call for $\lceil 150/30 \rceil = 5$ compositions.

The railway network is modeled as a graph $G = (V, E)$ of stations (vertices) and tracks (edges). The route through the network of line l is represented as a path in the graph. The lines and stations in the railway network are of one of three types.

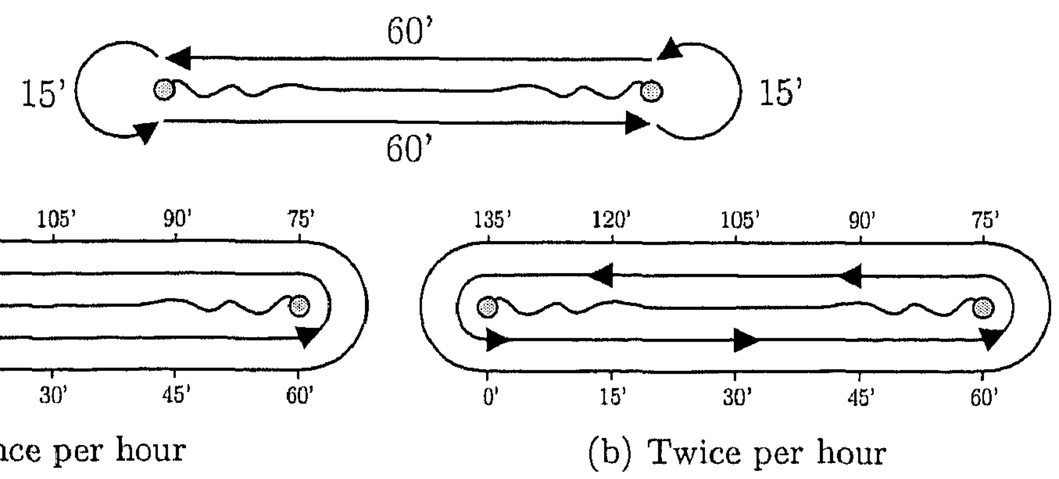


Figure 2 **Compositions vs. Frequency**

The intercity (IC) trains for longer distances and larger stations, the interregional (IR) trains for intermediate distances, and the aggloregional (AR) trains for the short distances and smaller stations. The types of a line and a station determine whether or not a line halts at a station; i.e., the IC trains stop only at the IC stations, the IR trains stop at IR and IC stations, and the AR trains stop at every station. The stations at which a line halts are thus given by the path through the network, together with the line and station types of the stations along this path. The total flow of travelers using the different train types can be decomposed into passenger flows for each type (Claessens et al. 1998). Travelers determine their route through the network, based on the associated travel time. They are thus motivated to switch to higher train types at the earliest opportunity. As described in Oltrogge (1994), this enables us to decompose the overall problem into separate line-planning problems. Using this approach, we consider only line-planning problems with all lines of exactly one type t. Line-planning models with multiple line and station types are discussed in Goossens et al. (2002). Additionally, we assume that for every type of line, there is exactly one type of carriage used to operate this line. Because all stations at which lines of type t do not halt can be removed from the network, the stops of a line l are thus given by the stations at the endpoints of all tracks used by l.

The demand data for the line-planning problem are given by the *origin/destination* (OD) matrix, specifying the number of travelers per hour between all pairs of stations. The route passengers take to go from their origin to their destination clearly influences the planning of the capacity of lines. As dictated by the ticket regulations, this route is in general the shortest path and can thus be fixed a priori. The assumption that there is exactly one fixed route is not important. The essence is that the route is known for every traveler. Moreover, we assume that the flow of passengers is symmetric. Thus, the number of passengers traversing some edge e can be calculated given the OD matrix. As a service to the passengers, we assume that each line is always operated in both directions. Therefore, as in Claessens et al. (1998), we can regard lines as being undirected.

The operated line plan in turn influences the behavior of travelers and thus the demand data. For example, a line plan offering more frequent connections might well encourage travelers currently using their cars to switch to using the train. As also mentioned by Claessens et al. (1998) and Bussieck (1998), the approach applied in practice is to use the resulting line plan to reallocate the passengers using, e.g., the system split method (Oltrogge 1994). The resulting OD matrix could be used to obtain a new line plan, etc.

The lines in the line plan are selected from a given set of feasible lines L through the network. Not every path between every pair of stations is a feasible route. Many infrastructural and operational restrictions have to be taken into account, such as the shunting possibilities for turning at the end stations and the maximum distance covered by a line (Härte 1995).

The level of service that the proposed line plan offers to the passenger is ensured by two conditions. The capacities of the lines should suffice to transport all passengers, and the line plan is enforced to guarantee a high number of connections between every pair of consecutive stations in the network. We consider this issue in more detail in the next subsection.

1.1. Formulation

Our formulation of CLPP is very similar to the formulation used by Claessens et al. (1998). Given are the undirected graph $G = (V, E)$ with vertex set V, edges E, and a set of potential lines L. Every line $l \in L$ corresponds to a set of edges forming a simple path between the end vertices of l. The consecutive stops of line l are given by the stations corresponding to the vertices along its path. Recall that we consider only line-planning problems with exactly one type of line. For every line, we have to decide whether to deploy it and, if so, at what hourly frequency and with how many carriages. The set of possible frequencies for the lines is denoted by $F \subset \mathbb{Z}_+$, the possible number of carriages by $C \subset \mathbb{Z}_+$. As mentioned before, the resulting line plan must meet two types of service restrictions. For every edge e in the network we are given the required number $\underline{f}_e$ of passing trains per hour and the total number of carriages $\underline{c}_e$ needed to meet the demand of all passengers who want to traverse e per hour.

For formulating the line-planning problem as an integer linear programming problem, we introduce a binary variable for every $(l, f, c) \in N$, with the set of triples as $N := L \times F \times C$. Every $i \in N$ will be used for referring to a particular (l_i, f_i, c_i) combination. For convenience, let us also introduce the set $N(e) \subseteq N$ for every edge e as $N(e) = \{i \in N : e \in l_i\}$. The line-planning problem CLPP can now be formulated as follows:

$$\min \sum_{i \in N} k_i x_i \tag{1}$$

$$\text{subject to } x \in S^{\text{CLPP}}, \tag{2}$$

where the set S^{CLPP} is defined as

$$\sum_{i \in N(e)} f_i x_i \geq \underline{f}_e \quad \forall e \in E, \tag{3}$$

$$\sum_{i \in N(e)} f_i c_i x_i \geq \underline{c}_e \quad \forall e \in E, \tag{4}$$

$$\sum_{i \in N | l_i = l} x_i \leq 1 \quad \forall l \in L, \tag{5}$$

$$x_i \in \{0, 1\} \quad \forall i \in N. \tag{6}$$

Before describing the objective function coefficients k_i, let us first consider the different restrictions. The constraints (3) enforce the minimum number of passing trains per hour on every edge $e \in E$. Restrictions (4) impose the lower bound $\underline{c}_e$ on the number of carriages crossing edge e in one hour. Typically, $\underline{c}_e$ is the smallest number of carriages necessary for transporting all passengers over edge e. Note that, as mentioned earlier, we consider all units of rolling stock to be identical. The third group of constraints ensures that every line is operated in at most one configuration. Contrary to Claessens et al. (1998) and Bussieck (1998), we have no upper-bound restriction on the number of passing trains per hour for every track. Including these restrictions to account for possible infrastructural bottlenecks would be part of a more operational study.

The objective function coefficient k_i represents the costs of operating the line associated to variable x_i. These costs are split into fixed and variable costs. Variable costs in this sense are hourly costs per kilometer associated to operating the line. They involve, for example, energy and maintenance costs. The fixed costs are costs that are incurred for the availability of the individual trains and carriages and involve, e.g., depreciation costs. The objective function coefficients k_i are defined as

$$k_{(l,f,c)} = \lceil cp_l \cdot f \rceil \cdot (k^{\text{tr}}_{\text{fix}} + c \cdot k^{\text{car}}_{\text{fix}}) + d_l \cdot f \cdot (k^{\text{tr}}_{\text{var}} + c \cdot k^{\text{car}}_{\text{var}}).$$

This formulation thus incorporates four classes of resource costs (Claessens et al. 1998, Bussieck 1998):

- $k^{\text{tr}}_{\text{fix}}$ the fixed hourly costs per train,
- $k^{\text{car}}_{\text{fix}}$ the fixed hourly costs per carriage,
- $k^{\text{tr}}_{\text{var}}$ the hourly costs per train per kilometer,
- $k^{\text{car}}_{\text{var}}$ the hourly costs per carriage per kilometer.

The distance in kilometers covered by a line l is given by d_l. The parameter cp_l is equal to the total cycle time of l divided by 60 minutes (e.g., $cp_l = 150/60$ for the line in Example 1.1).

Claessens et al. (1998) prove that finding a cost-optimal allocation of trains using only capacity restrictions is NP-hard. Because this problem is a special case of CLPP, we know that CLPP is also NP-hard.

Before proceeding with describing our branch-and-cut method, let us first introduce some general notation and visualization of CLPP instances throughout the remainder of this paper. Consider Figure 3. The instance consists of three stations, $V = \{v_1, v_2, v_3\}$, and two tracks, $E = \{e_1, e_2\}$. The overall set of candidate lines L for this network consists of three lines $L := \{l_1, l_2, l_3\}$. We refer to an individual variable x_i as being of line l_i with a frequency of f_i per hour, pulling c_i carriages, and, therefore, with an hourly capacity of $f_i \cdot c_i$ carriages.

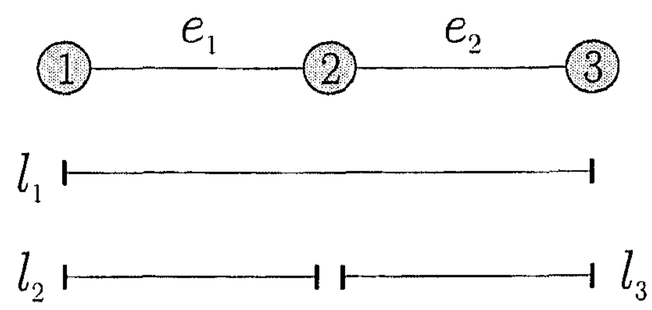


Figure 3 Example of a CLPP Instance

2. Branch-and-Cut Method for CLPP

The branch-and-cut techniques will be divided into three different sections: preprocessing (§2.1), cutting planes (§2.2), and tree search (§2.3). The preprocessing section focuses on reducing the initial size of the problem. This is done by removing superfluous variables and constraints, and by tightening the model restrictions. Next, we discuss a variety of classes of cutting planes, both generally applicable and cutting planes specifically derived for CLPP. Finally, the tree search section will cover branching rules and primal heuristics.

2.1. Preprocessing

The preprocessing techniques that will be described in this section will attempt to strengthen the initial formulation of the problem. Strengthening, in this context, covers both coefficient reduction techniques and methods for reducing the number of variables and constraints used to model a given CLPP instance. Both Claessens et al. (1998) and Bussieck (1998) describe several preprocessing techniques specifically for CLPP. We briefly recall these techniques and review them in a more general context.

2.1.1. Coefficient Reduction. The main constraints of the model—i.e., the capacity constraints (4) and the frequency constraints (3)—will be strengthened using the methods described below. The coefficient strengthening or reduction techniques used here are extensions of techniques and ideas described in Dietrich and Escudero (1994).

The coefficient-strengthening techniques will be described on a simplified line-planning problem (SLPP) with only two types of constraints:

$$\min cx \qquad \text{subject to } x \in S, \tag{7}$$

where the feasible set S is defined as all x for which

$$Ax \geq b, \tag{8}$$

$$Cx \leq \mathbf{1}, \tag{9}$$

$$x \in \{0, 1\}^n, \tag{10}$$

where N is the set of variables with $n = |N|$. The matrix A is an $m \times n$ matrix with nonnegative integer entries a_{ij}, and b is an m-dimensional vector of positive integers. Every variable $i \in N$ is associated with a unique $l \in L$, denoted l_i. For every $l \in L$, there is a constraint $\sum_{i|l_i=l} x_i \leq 1$ in the second constraint matrix C. Note that CLPP is contained in this class, where the constraints (8) contain the service constraints on the tracks.

We derive several methods for strengthening problems of this form. Using notation similar to Dietrich and Escudero (1994), the coefficient a_{kj} of row k and column j can in general be strengthened to

$$\bar{a}_{kj} \leftarrow \max\{0, a_{kj} - \Delta_{kj}\} = \max\{0, b_k - Q_{kj}\}, \qquad (11)$$

where $\Delta_{kj} = a_{kj} - b_k + Q_{kj} \geq 0$. Here, Q_{kj} represents a lower bound on the left-hand-side value of constraint k in any feasible solution $\bar{x} \in S$ with $\bar{x}_j = 1$. To maintain the nonnegativity property for a_{kj}, we have added the lower bound on $\bar{a}_{kj}$. Note that an initial strengthening can be obtained by setting $\bar{a}_{kj} = b_k$ for all $a_{kj} \geq b_k$ because we have assumed $a_{kj} \geq 0$ and all variables are binary. This reasoning also implies $Q_{kj} \geq 0$.

Next we give two techniques for determining valid values for Q_{kj}. Both techniques use an additional row from the constraint matrix A to strengthen its coefficients. Note that if Q_{kj} is valid for S, then $\widehat{Q}_{kj} \leq Q_{kj}$ is also valid. Alternatively, for any valid, though noninteger, $\widehat{Q}_{kj}$ we know that $\lceil \widehat{Q}_{kj} \rceil$ is also valid. Both techniques give lower bounds for the value of the following minimization problem.

THEOREM 2.1. *Consider an instance of SLPP as defined by* (7)–(10). *Strengthening the coefficient* a_{kj} *of the constraint matrix* A *according to* (11) *with*

$$Q_{kj}(h) = \min \sum_{i \neq j} a_{ki} x_i$$
$$\text{subject to} \quad \begin{cases} a_{hj} + \sum_{i|l_i \neq l_j} a_{hi} x_i \geq b_h, \\ Cx \leq \mathbf{1}, \quad x \in \{0,1\}^n \end{cases} \qquad (12)$$

if $a_{hj} < b_h$, *and* $Q_{kj}(h) = 0$ *otherwise, is valid for* S *for any constraint* h.

PROOF. We prove that for all $x \in S$, it holds that $(b_k - Q_{kj}(h))x_j + \sum_{i \neq j} a_{ki} x_i \geq b_k$. This is obviously true for all $x \in S$ for which $x_j = 0$ because now the new coefficient of x_j is unimportant. For all $x \in S$ for which $x_j = 1$, we have to show that $\sum_{i \neq j} a_{ki} x_i \geq b_k - (b_k - Q_{kj}(h)) = Q_{kj}(h)$. Clearly, any $x \in S$ with $x_j = 1$ satisfies $a_{hj} + \sum_{i|i \neq j} a_{hi} x_i \geq b_h$ and $x_j + \sum_{i \neq j | l_i = l_j} x_i \leq 1$, making x a feasible solution to the minimization problem in (12). □

The time needed to solve the integer-covering problem in (12) is too high for it to be used for coefficient strengthening. We discuss two relaxations of this problem that give good bounds for $Q_{kj}(h)$.

EXAMPLE 2.1. Consider the set of integer points

$$S := \{x \in \{0,1\}^5 : 3x_1 + 4x_2 + 7x_3 + 4x_4 + 6x_5 \geq 7$$
$$x_1 + x_2 + x_3 + x_4 + 2x_5 \geq 2$$
$$x_1 + x_2 + x_3 \leq 1$$
$$x_4 + x_5 \leq 1\}.$$

Let us strengthen the coefficient of x_3 in the first constraint. Although it is equal to the right-hand side, any feasible solution $x \in S$ with $x_3 = 1$ will still have at least one of the variables x_4 or x_5 set to 1 for the second constraint to be satisfied. Meeting the second constraint would therefore imply an additional left-hand side of $Q_{13} = \min\{4, 6\} = 4$ units in the first constraint, and therefore

$$3x_1 + 4x_2 + (7-4)x_3 + 4x_4 + 6x_5 \geq 7$$

is valid for S. Note that, for instance, the fractional solution $\{0, 0, \frac{1}{4}, 0, \frac{7}{8}\}$ to the LP relaxation of S is cut off by the strengthened constraint.

The following corollary generalizes the idea of the previous example.

COROLLARY 2.1. *Consider an instance of SLPP as defined by* (7)–(10). *Let us take two not necessarily distinct rows from the constraint matrix* A, *say rows* h *and* k. *Strengthening the coefficient of* a_{kj} *according to* (11) *with*

$$Q'_{kj}(h) \equiv \begin{cases} a_{kr} & \text{if } a_{hj} < b_h, \\ 0 & \text{otherwise,} \end{cases} \qquad (13)$$

where $r := \arg\min_{i|l_i \neq l_j,\, a_{hi} > 0} a_{ki}$, *is valid for* S.

PROOF. We prove the validity of $Q'_{kj}(h)$ by showing that it is the solution to a relaxation of the optimization problem in (12). Given that $a_{hj} < b_h$, then $\sum_{i|l_i \neq l_j} a_{hi} x_i > 0$ is a relaxation of the restriction in (12). Now observe that, for binary x, this new constraint is satisfied only if at least one x_i with $l_i \neq l_j$ and $a_{hi} > 0$ has value 1, and thus

$$a_{kr} = \min \sum_{i \neq j} a_{ki} x_i$$
$$\text{subject to} \quad \sum_{i|l_i \neq l_j} a_{hi} x_i > 0, \quad x \in \{0,1\}^n. \quad \square$$

Note that rows h and k need not be different. The above method for finding valid Q_{kj} works well on constraints with moderately sized $b_h - a_{hj}$. If this number is larger, we may consider another relaxation of (12), where we drop the integrality constraints imposed on the variables in the covering problem. As described later, the resulting continuous-covering problem can be solved to optimality using a greedy algorithm.

EXAMPLE 2.2. Consider the feasible set S as given in Example 2.1. We again strengthen the coefficient of x_3 in the first constraint. In any feasible solution x with $x_3 = 1$, Equation (12) tells us that strengthening a_{13} with

$$Q_{13} = \min\{4x_4 + 6x_5\}$$
$$\text{subject to} \quad x_4 + 2x_5 \geq 1, \quad x_4 + x_5 \leq 1, \quad x_4, x_5 \in \{0, 1\}$$

is valid for S. If we drop the integrality restrictions on x_4 and x_5, we can solve the remaining problem using a greedy algorithm. The optimal solution is $x_5 = 0.5$, $x_4 = 0$, which shows us that $Q_{13} \geq 3$. From this we know that

$$3x_1 + 4x_2 + (7 - 3)x_3 + 4x_4 + 6x_5 \geq 7$$

is valid for S.

The following corollary generalizes this idea.

COROLLARY 2.2. *Consider SLPP taking two rows from the constraint matrix A, say rows h and k. Strengthening the coefficient a_{kj} of column j in row k according to* (11) *is valid for S, with*

$$Q''_{kj}(h) \equiv \min \sum_{i \neq j} a_{ki} x_i$$
$$\textit{subject to} \quad \begin{cases} a_{hj} + \sum_{i \mid l_i \neq l_j} a_{hi} x_i \geq b_h, \\ Cx \leq \mathbf{1}, \quad x \in [0, 1]^n \end{cases} \tag{14}$$

if $a_{hj} < b_h$, and $Q''_{kj}(h) = 0$ otherwise.

PROOF. Clearly, (14) is a relaxation of the optimization problem in Theorem 2.1, and thus $Q''_{kj}(h) \leq Q_{kj}(h)$. □

The continuous-covering problem with nonoverlapping constraints (9) is closely related to the multiple-choice knapsack problem (see Martello and Toth 1990). It can be solved using a greedy algorithm very similar to the one for continuous knapsack problems.

The strengthening methods given above all build on the assumption that all coefficients are nonnegative. Note that this property is guaranteed by $\bar{a}_{kj} = \max\{0, b_k - Q_{kj}\}$ in (11).

2.1.2. Variable Reduction. We reduce the number of variables by deriving dominance relations between groups of variables. This is a preprocessing technique that uses an exchange argument between variables in feasible solutions. The essence of this technique is also described in Claessens et al. (1998). The variable reduction of Bussieck (1998) uses similar techniques. Neither party, however, links their variable reduction methods to coefficient-strengthening procedures. By introducing this link, we improve upon their results and give a more transparent description of these techniques. Note, as stated in Bussieck (1998), that their variable reduction is done without strictly preserving the feasibility of the elimination and the lower bounding.

If in any feasible solution x with $x_j = 1$ this solution can be altered to $x_j = 0$ and $x_i = 1$ for some i, while maintaining feasibility and without worsening the objective value, then x_j is said to be dominated and can be removed from the model. If there is a fixed x_i with this property, then x_i is said to dominate x_j.

LEMMA 2.1. *Given an instance of SLPP as defined by* (7)–(10), *consider two variables $i, j \in N$, with $l_i = l_j$ and $i \neq j$. Variable x_i dominates variable x_j and can therefore be removed from the problem if*

$$c_i \leq c_j \quad \textit{and} \tag{15}$$
$$a_{ki} \geq a_{kj} \quad \forall k \in \{1, \ldots, m\}. \tag{16}$$

PROOF. Straightforward. □

Note that the variable dominance relations described above are transitive; i.e., if i dominates j and j dominates k, then i dominates k. Variable dominance is tested between all variables of a line $l \in L$. The dominance technique described above is also used in Claessens et al. (1998), but without linking it to coefficient-reduction techniques. Without coefficient strengthening, the model will not contain many dominated variables.

2.1.3. Constraint Reduction. Next, we derive dominance rules for constraints. Again we assume the problem to be given in the form of SLPP. We give conditions under which constraints are redundant for the description of the set of feasible solutions.

THEOREM 2.2. *Suppose we are given an instance of SLPP and a pair of constraints, say h and k. Now, constraint k is redundant for the description of SLPP if $b_k \leq Q_{k0}(h)$, with*

$$Q_{k0}(h) \equiv \min \sum_{i \in N} a_{ki} x_i \qquad \textit{subject to} \quad x \in P', \tag{17}$$

where $P' := \{x \mid \sum_{i \in N} a_{hi} x_i \geq b_h, Cx \leq \mathbf{1}, x \in [0, 1]^n\}$.

PROOF. Clearly, the feasible region P' is a superset of the solution set S of the original problem, since it is defined by only one constraint of the constraint matrix A, and for P' we have $x \in [0, 1]^n$. From this definition, we know that for every solution $x \in P'$:

$\sum_{i \in N} a_{ki} x_i \geq Q_{k0}(h)$. Because $S \subseteq P'$, this also holds for feasible solutions $x \in S$. □

This theorem can also be used to strengthen the right-hand side b_k to $\lceil Q_{k0}(h) \rceil$ because all coefficients a_{ki} are integer.

EXAMPLE 2.3. Consider the feasible set S defined as

$$\begin{aligned} S := \{x \in \{0,1\}^5 : \; & 3x_1 + 4x_2 + 7x_3 + 4x_4 + 7x_5 \geq 7 \\ & x_1 + x_2 + 2x_3 + x_4 + x_5 \geq 2 \\ & x_1 + x_2 + x_3 \leq 1 \\ & x_4 + x_5 \leq 1\}. \end{aligned}$$

Defining the minimization problem (17) on the first two constraints, then $Q_{10}(2) = 7$ from the solution $(1,0,0,1,0)$. Thus, the second constraint induces a left-hand-side value for the first constraint of seven in the LP relaxation. Therefore, the first constraint is redundant.

Note that the constraint-reduction or strengthening techniques are not independent of one another. Similar to the coefficient strengthening and variable reduction techniques described earlier, they should be applied sequentially, updating the optimization problem in (17) accordingly every time.

Constraint dominance frequently occurs in CLPP instances, both between the capacity and frequency constraints of some track, as well as between the service constraints on connecting tracks. Especially for a dead-end track, it often occurs that all lines covering such a track also cover a neighboring track, causing overlapping nonzero elements in the service constraints of both tracks. If the required frequency and capacity for the dead-end track are higher than for the neighboring track, then the service constraints of the neighboring track are clearly redundant.

2.2. Cutting Planes

Besides preprocessing, we describe several classes of cutting planes. The separation algorithms for these classes will be discussed in §3.2.

2.2.1. Probing Cuts. Let us describe this class of cutting planes on problems of the form (7)–(10). We derive valid inequalities from information that is obtained from variable probing.

EXAMPLE 2.4. Let us assume $\tilde{x}^*$ is the optimal solution to the LP relaxation of an instance of SLPP. Here, $\tilde{x}^*_1 = 1$, $\tilde{x}^*_2 = 0.8$, and $\tilde{x}^*_i = 0$ for all $i \notin \{1,2\}$. One constraint in the associated problem is

$$5x_1 + 10x_2 + \sum_{i \in N \setminus \{1,2\}} a_{ki} x_i \geq 13.$$

Fixing x_1 to 1 in this constraint yields

$$10x_2 + \sum_{i \in N \setminus \{1,2\}} a_{ki} x_i \geq 8 \Rightarrow 8x_2 + \sum_{i \in N \setminus \{1,2\}} a_{ki} x_i \geq 8.$$

The left inequality is also valid in case $x_1 = 0$ because clearly it is weaker than the original inequality. Now by using, for example, the coefficient-strengthening techniques, we can reduce the coefficient of x_2, yielding the right tightened constraint. It is violated 20% by $\tilde{x}^*$.

The idea described in the example given here is generalized in the following lemma. The way in which these cuts are constructed preserves the problem-specific structure of the initial inequality. Therefore, they can be used to generate new cuts of known classes of valid inequalities. Note that the probing inequalities themselves will never be violated.

LEMMA 2.2. *Consider an instance of SLPP and some variable $j \in N$. Now, the inequality*

$$\sum_{i \mid l_i = l_j,\, a_{ki} > a_{kj}} a_{ki} x_i + \sum_{i \mid l_i \neq l_j} a_{ki} x_i \geq b_k - a_{kj} \tag{18}$$

is valid for S for any constraint k of the constraint matrix A.

PROOF. For any feasible solution, we know that $\sum_{i \mid l_i = l_j} x_i \leq 1$, and thus

$$\begin{aligned} b_k &\leq \sum_{i \in N} a_{ki} x_i \\ &= \sum_{i \mid l_i = l_j} a_{ki} x_i + \sum_{i \mid l_i \neq l_j} a_{ki} x_i \\ &= \sum_{i \mid l_i = l_j,\, a_{ki} \leq a_{kj}} a_{ki} x_i + \sum_{i \mid l_i = l_j,\, a_{ki} > a_{kj}} a_{ki} x_i + \sum_{i \mid l_i \neq l_j} a_{ki} x_i \\ &\leq a_{kj} + \sum_{i \mid l_i = l_j,\, a_{ki} > a_{kj}} a_{ki} x_i + \sum_{i \mid l_i \neq l_j} a_{ki} x_i. \quad \square \end{aligned}$$

Because inequality (18) is valid for the original problem, it is also possible to recursively apply this technique for a set of variables. Note that the resulting inequality is order independent as long as at most one variable per line is used for probing.

2.2.2. Two-Cover Cuts. This class of cutting planes is based on the covering constraints in the CLPP on a given track e. If variable i by itself does not satisfy both service constraints, then every feasible solution will contain at least one more line across e. This observation is made in the following example.

EXAMPLE 2.5. Consider the following polytope S of a CLPP instance

$$S := \{x \in \{0,1\}^5 : \; x_1 + x_2 + x_3 + 2x_4 + 2x_5 \geq 3\}.$$

Because at least two of these variables will have a nonzero value in any feasible solution,

$$x_1 + x_2 + x_3 + x_4 + x_5 \geq 2$$

is valid for S.

LEMMA 2.3. *For any instance of CLPP, the inequality*

$$\sum_{i\in C} x_i + \sum_{i\in \bar{C}} 2x_i \geq 2 \tag{19}$$

is a valid inequality for every $e \in E$. The set C and its complement $\bar{C}$ are defined as

$$C := \{i \in N(e) \mid f_i < \underline{f}_e \vee f_i c_i < \underline{c}_e\}$$

and $\bar{C} := N(e)\backslash C$.

PROOF. Clearly, by definition of C, any feasible integer solution $\bar{x}$ with all nonzero variables in the left summation must consist of at least two distinct $i, j \in C$ with $\bar{x}_i = \bar{x}_j = 1$. □

The next corollary describes an extension of Lemma 2.3 in which the variables are not divided into two but into several parts. In particular, the resulting multicovering (MC) cuts consider $n+1$ subsets.

COROLLARY 2.3. *Consider an arbitrary instance of CLPP and a track e. Given a positive integer n for which $2 \leq n+1 < \underline{c}_e$, then the inequality*

$$\sum_{s=1}^{n} \sum_{i\in C^s} s x_i + \sum_{i\in \bar{C}} (n+1) x_i \geq n+1 \tag{20}$$

is valid for CLPP, with C^s and $\bar{C}$ given as $C^s = \{i \in N(e) \mid \underline{c}_e(s-1) \leq f_i c_i n < \underline{c}_e s\}$, and $\bar{C} = N(e)\backslash \cup_s C^s$.

PROOF. First, let us denote $f_i c_i$ by a_i and $\underline{c}_e$ by b. Note that all numbers are integers. Thus, $s_i = \min\{s \in \mathbb{N} \mid b(s-1) \leq a_i n < bs\}$ is equal to $s_i = \min\{s \in \mathbb{N} \mid (b-\epsilon)(s-1) < a_i n < (b-\epsilon)s\}$ for suitably small $\epsilon > 0$. To prove that (20) is valid, we show that it is a Gomory cut obtained from (4) with multiplication factor $n/(b-\epsilon)$. This is clear for the left-hand-side coefficients. It remains to be shown that $\lceil bn/(b-\epsilon)\rceil = n+1$ for all applicable values of b and n. Clearly, $\lceil nb/(b-\epsilon)\rceil \geq n+1$. It is also easy to show that $\lceil nb/(b-\epsilon)\rceil \leq \lceil nb/(b-1) \leq n+1$: $\lceil nb/(b-1)\rceil - (n+1) = \lceil nb/(b-1) - (n+1)\rceil \leq \lceil n/(b-1)\rceil - 1 \leq 1-1=0$. □

EXAMPLE 2.6. Consider the following polytope S of a CLPP instance.

$$S := \{x \in \{0,1\}^5 : 3x_1 + 5x_2 + 7x_3 + 10x_4 + 14x_5 \geq 14\}.$$

If we consider $n=2$, then $C^1 = \{1,2\}$, $C^2 = \{3,4\}$, and $\bar{C} = \{5\}$, and thus from Corollary 2.3 we know that the inequality

$$x_1 + x_2 + 2x_3 + 2x_4 + 3x_5 \geq 3$$

is valid for S.

2.2.3. Flow-Cover Cuts. The flow-cover cuts are described in Bussieck (1998). These cuts use the line structure of CLPP, in which lines typically cover more than one track.

LEMMA 2.4. *Let $C \subset E$, $e_0 \in E\backslash C$ for which $\underline{c}_{e_0} > \sum_{e\in C} \underline{c}_e$. Then, the inequality*

$$\left(\underline{c}_{e_0} - \sum_{e\in C} \underline{c}_e\right) \sum_{i\in N(e_0)\mid l_i\cap C=\varnothing} x_i + \sum_{e\in C} \sum_{i\in N(e)} f_i c_i x_i \geq \underline{c}_{e_0} \tag{21}$$

is valid.

PROOF. First, suppose $\sum_{i\in N(e_0)\mid l_i\cap C=\varnothing} x_i \geq 1$. Because adding the capacity constraints for all tracks in C gives $\sum_{e\in C}\sum_{i\in N(e)} f_i c_i x_i \geq \sum_{e\in C} \underline{c}_e$, validity is obvious. Next, assume $\sum_{i\in N(e_0)\mid l_i\cap C=\varnothing} x_i = 0$. All lines that pass e_0 thus also use at least one track in C. This implies that

$$\underline{c}_{e_0} \leq \sum_{i\in N(e_0)} f_i c_i x_i \leq \sum_{e\in C}\sum_{i\in N(e)} f_i c_i x_i. \quad \square$$

Corollary 2.4 identifies redundant choices for e_0 and C, which is useful for the separation algorithm (§3.2).

COROLLARY 2.4. *Consider a flow-cover inequality defined by the pair (e_0, C) with $e_0 \in E$ and $C \subseteq E\backslash e_0$. If the capacity constraint for e_0 has been strengthened using the techniques described in §2.1, then flow-cover inequalities with C, such that no lines across e_0 cross any edge in C, i.e., $\{l \mid e_0 \in l, l \cap C \neq \varnothing\} = \varnothing$, cannot cut off any feasible solution to the LP relaxation.*

PROOF. Strengthening the capacity constraint for e_0 implies that for all variables i using e_0, $f_i c_i \leq \underline{c}_e$ holds. Thus, $\sum_{i\in N(e_0)} x_i \geq 1$. Now, because $\{l \in L: e_0 \in l\} = \{l \in L: e_0 \in l, l \cap C = \varnothing\}$, this implies that $\sum_{i\in N(e_0)\mid l_i\cap C=\varnothing} x_i \geq 1$. We can thus prove this corollary along the same lines as used in the proof of Lemma 2.4. □

2.3. Tree Search

In this section we consider several problem-specific branching rules and primal heuristics.

2.3.1. Branching. As reported in Linderoth and Savelsbergh (1999), implementing problem-specific branching rules can significantly reduce the size of the branching tree and thus speed up the solution process. In general, the applied branching rule when considering binary problems is to branch on a single variable, thus creating two new subproblems. In this section we discuss several alternative rules.

Branching rules are used to split a problem into several—usually two—new subproblems. Similar to cutting planes in a class of cuts, every class of branching rules contains many possible branching instances. For example, for the class of variable branching, a branching instance is a variable. The

quality of a branching instance in our minimization problem is measured by the lowest bound of the new subproblems. The branching instance for which this lowest bound is maximal is considered best. See also §3.3.

Variable Branching. Using the solution to the LP relaxation $\tilde{x}$ at a subproblem, the choice of a branching variable is made by taking the variable j for which $\tilde{x}_j$ is closest to $\frac{1}{2}$. Ties are broken by considering the variable for which the objective coefficient is largest.

Generalized Upper Bounds and Special Ordered Set Branching. An alternative class of branching rules for problems containing generalized upper bound (GUB) constraints of the form

$$\sum_{i \in C} x_i \leq 1 \tag{22}$$

for $C \subseteq N$ with $x \in \{0, 1\}^{|N|}$ is to branch on such a GUB constraint (Linderoth and Savelsbergh 1999). In any feasible solution, (22) ensures that at most one of the variables in C can be set to one. We can use (22) to split the problem as follows:

$$\sum_{i \in \bar{C}} x_i = 1 \quad \text{versus} \quad x_i = 0 \text{ for all } i \in \bar{C}. \tag{23}$$

The problem is to determine the set $\bar{C} \subseteq C$. In special ordered set (SOS) branching, an ordering of the variables in C is used to determine $\bar{C}$. The variables in the GUB constraints (5) for every line l can be ordered according to their capacity. Now, a subproblem for which the LP solution $\tilde{x}$ is fractional is split into two new problems as in (23), using $C := \{i \mid l_i = l\}$ and

$$\bar{C} := \{i \in C \mid f_i c_i \leq \delta\} \quad \text{where } \delta := \sum_{i \in C} f_i c_i \tilde{x}_i. \tag{24}$$

The parameter δ is the so-called branching value of the SOS.

Line Branching. Similar to generalized upper-bound branching rules, the line-branching rule also splits a subproblem into two new subproblems using the GUB constraints (5) on the lines. In line branching, we take $\bar{C} = C$. Thus, the branching dichotomy for some line l will be

$$\sum_{i|l_i=l} x_i = 0 \quad \text{versus} \quad \sum_{i|l_i=l} x_i = 1. \tag{25}$$

The effect of line branching on the new subproblems is two-sided. On one hand, the added restriction itself has a direct effect on the solution to the LP relaxation. On the other hand, the LP problem in the left (D^{l-}) subproblem becomes smaller because we can remove any variable of line l from the model. The upward branch allows us to obtain a higher value for Q_{kj} for strengthening the coefficients of the constraint matrix. When $\sum_{i|l_i=l} x_i = 1$, then for all variables j that do not belong to line l,

$$Q_{kj} = \min_{i|l_i=l} a_{ki}$$

is clearly valid.

Capacity Branching. A subproblem is split into two new problems by taking a set of integer variables C and enforcing

$$\sum_{i \in C} x_i \leq g \quad \text{versus} \quad \sum_{i \in C} x_i \geq g + 1 \tag{26}$$

for some integer $0 \leq g < |C|$. The set C is constructed to contain variables with an identical capacity. Given a track e, a capacity b, and a number of variables g, the current problem is split using (26) with $C := \{i \in N(e) \mid f_i c_i = b\}$. Because x is restricted to integer values, the two branches cover the complete solution space.

2.3.2. Primal Heuristic. Our primal heuristic is based on the LP relaxation of the problem in a subproblem. From this LP solution $\tilde{x}$, a new CLPP is constructed, using only the lines $L' := \{l \in L \mid \sum_{i|l_i=l} \tilde{x}_i > 0\}$. This significantly smaller problem is given to CPLEX, while bounding the available CPU time to a default value of 60 seconds. Clearly, integer solutions to these problems are also feasible integer solutions to the original problem. This primal heuristic is by default applied at the root node and every tenth node with a depth in the enumeration tree of at least five.

3. Implementation Issues

We now discuss the implementation of the preprocessing, cutting-planes, and branching techniques of the previous sections.

3.1. Preprocessing Implementation

All preprocessing techniques of §2.1 are applied at every subproblem of the enumeration tree. The implementation of the coefficient reduction techniques of §2.1.1 is not described here, but in §3.2.

3.1.1. Variable Reduction. The dominance rules for variable reduction are tested between all pairs of variables belonging to the same line. For identifying dominance relations, it suffices to compare the coefficients of both variables only in all (strengthened) service constraints present in the model, not the added cutting planes. By definition, this is sufficient for the validity of the reduction in any optimal integer solution.

Instead of using the strengthened constraints as given by the coefficient-reduction techniques to identify variable dominance, we use the following, stronger approach. Consider strengthening the coefficients for every line separately. Given some line l, strengthening all service constraints only for this

line will result in valid inequalities that are at least as tight as the previous restrictions. Thus, we can derive dominance relations based on these new constraints. Because removing these variables is also valid for the original system, we can repeat this individual approach for all lines, thereby circumventing the order dependence of the strengthening procedure.

3.1.2. Constraint Reduction. At the root problem, the constraint-reduction technique is applied to all pairs of service constraints h and k for tracks that have one vertex in common. Recall from Theorem 2.2 that constraint k is redundant for the description of SLPP if $b_k \leq Q_{k0}(h)$ for some constraint h. If $b_k < Q_{k0}(h)$, then constraint k will not be removed, but b_k will be set to $\lceil Q_{k0}(h) \rceil$. In all other subproblems of the enumeration tree, we only apply this technique on constraints h and k belonging to the same track.

3.2. Cutting Planes Implementation

Next we describe the separation algorithms to find violated inequalities of the proposed classes of cutting planes. In every iteration, the separation algorithms of all classes of cutting planes are called, and violated inequalities are added to the LP relaxation. The LP is then reoptimized. These two steps are repeated until no more cuts are found. Note that for all classes of cutting planes we impose a minimum violation of 1% for the valid inequality to be added to the system.

3.2.1. Coefficient-Strengthening Cuts. The results of the coefficient-strengthening techniques of §2.1.1 depend on the order of strengthening. The later a coefficient is strengthened, the smaller the effect of the strengthening will be. We describe a heuristic for determining a permutation of the variables. In the root problem, this order is used to alter the coefficients in the constraints. In subsequent subproblems a new cut built up from strengthened coefficients is added to the problem description, replacing the initial constraint.

Recall that strengthening the coefficient a_{kj} of variable j for constraint k involves finding a valid value for $Q_{kj}(h)$, for some constraint h. For these h-k pairs, we only consider the frequency and capacity restrictions on a track e. The reduced coefficient $\bar{a}_{kj}$ is then determined by

$$\bar{a}_{kj} \leftarrow \min\{a_{kj}, (b_k - Q_{kj})^+\}, \quad \text{where}$$
$$Q_{kj} := \max\{Q'_{kj}(k), Q'_{kj}(h), Q''_{kj}(h)\}. \tag{27}$$

For every track e, we first strengthen the coefficient in the frequency constraint, then in the capacity constraint.

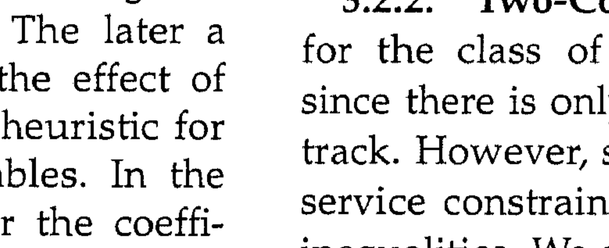
The strengthening is done in two phases. First, the coefficients of variables with nonzero values in the current solution of the LP relaxation $\tilde{x}$ are strengthened. The ordering of the variables is done in blocks of the same line. These line blocks are sorted according to $\sum_{i|l_i=l} \tilde{x}_i$, strengthening first those lines for which this number is high. The order of the tracks is given by the sequence of the tracks that are passed by the line from its origin to its destination station. In the second phase, we reuse this ordering of the lines, but now strengthen all coefficients of the remaining variables.

The resulting cutting planes are added to the LP of the current subproblem. Because all coefficients are either not changed, or strictly smaller than the coefficients in the original constraint, we can safely remove the original constraints from the current LP. From that point on, these new constraints are considered to be the service constraints of the various tracks.

3.2.2. Two-Cover Cuts. The separation algorithm for the class of two-cover cuts is straightforward, since there is only one two-cover inequality for every track. However, substituting for fixed variables in the service constraints gives rise to new two-cover (2C) inequalities. We also use the probing techniques from §2.2.1 for finding new violated 2C inequalities.

EXAMPLE 3.1. Consider the instance with two tracks shown in Figure 4. The displayed fractional solution $\tilde{x}$ has two variables at nonzero values, $\tilde{x}_j = 1$ and $\tilde{x}_r = \frac{1}{2}$. The 2C inequality for track e_2 is $x_j + 2 \cdot x_r + \cdots \geq 2$. Clearly, $\tilde{x}$ satisfies this 2C cut. Now consider probing on x_j. This shows that substituting for x_j in the capacity constraint results in a valid inequality with

i	(l_i, f_i, c_i)	$\tilde{x}_i$
j	$(l_1, 1, 5)$	1
r	$(l_2, 1, 10)$	$\frac{1}{2}$



e	$(\underline{f}_e, \underline{c}_e)$
e_1	$(1, 5)$
e_2	$(1, 10)$

Figure 4 Two-Cover Cuts

$\underline{c}_{e_2} = 10 - 5 = 5$. The corresponding 2C cut $2 \cdot x_r + \cdots \geq 2$ is violated 50% by $\tilde{x}$.

For every track e, we probe on a set of variables, also as described in §2.2.1. This set is built in two steps. First, all variables j for which $\tilde{x}_j = 1$ and $e \in l_j$ are added. Second, we add one additional variable that results in the maximally violated new 2C cut by considering all i for which $e \in l_i$ and $0 < \tilde{x}_i < 1$.

3.2.3. Multicover Cuts. The probing set $\{i \mid e \in l_i, \tilde{x}_i = 1\}$ is used for finding violated multicover cuts for every track e. Furthermore, we limit the search for the number of cover intervals to $3 \leq n+1 \leq \min\{6, \underline{c}_e - 1\}$. For every track, we only add the multicover constraint for n for which the violation is maximal.

3.2.4. Flow-Cover Cuts. Solving the separation problem for flow-cover cuts is done heuristically. As mentioned before, simply enumerating all possible choices for e_0 and C is not an option. Corollary 2.4 shows us that the edge e_0 and the edges in the set C should not be too far apart in the graph G. From the proof of Lemma 2.4, it is clear that, given the solution $\tilde{x}^*$ to the LP relaxation, violated flow-cover cuts can only be found for tracks e_0 and C for which $0 < \sum_{i \in N(e_0) | l_i \cap C = \varnothing} x_i < 1$. The separation algorithm considers only all possible e_0 and C that are subsets of $\delta(v)$ for all stations $v \in V$, where $\delta(v)$ is the set of edges for which v is one of the two endpoints. For every combination of v and e_0, we only add the cut that is violated most by $\tilde{x}^*$.

3.3. Branching Rules Implementation

As mentioned in §2.3.1, we compare different branching instances by using estimates for the resulting increase of the lower bound in the new subproblems. Determining these lower bounds can be done exactly by explicitly solving the new LP relaxations (*strong branching*), or heuristically by determining estimates for the new objective values. In our implementation we heuristically find one candidate instance for every class of rules and then use strong branching to decide among these candidates.

3.3.1. Special Ordered Set Branching. From all the lines in a given CLPP instance, we select the line l with the largest number of fractional variables as the line to branch on. Similar to Linderoth and Savelsbergh (1999), we only consider SOS branching on lines with at least three variables at a fractional value. If there is more than one line that attains the largest number of fractional variables, we break ties by choosing the line with the highest-weighted objective value $\sum_{i | l_i = l} k_i \tilde{x}_i$. The set $\bar{C}$ is set to be the largest of $\bar{C}$ as in (24) and $C \backslash \bar{C}$. Note the possibility of a branching cycle, i.e., that the current solution $\tilde{x}$ remains feasible in one of the new subproblems. This would imply that for this new problem the previously chosen branching rule would again be best. This is prevented by ensuring that $\varnothing \neq \bar{C} \neq \{i \in C \mid \tilde{x}_i > 0\}$.

3.3.2. Line Branching. We pick the candidate line by calculating estimates for the new LP lower bounds of the new problems in case this line would be chosen to be branched on. These degradation estimates for branching on line l are based on the objective function coefficients for a fractional solution $\tilde{x}$:

$$D^- := \sum_{i | l_i = l} k_i \tilde{x}_i \qquad D^+ := \frac{1 - \sum_{i | l_i = l} \tilde{x}_i}{\sum_{i | l_i = l} \tilde{x}_i} \sum_{i | l_i = l} k_i \tilde{x}_i,$$

where D^- and D^+ are the estimates for the left and right branch, respectively. From the lines for which $\sum_{i | l_i = l} x_i$ is fractional, we select the line with the highest value for $\min\{D^-, D^+\}$. This rule represents the multiple-variable variant of the standard objective-function-based degradation estimate.

3.3.3. Capacity Branching. Given an LP solution $\tilde{x}$ in a subproblem, we choose the track e with the largest number of fractional variables $|\{i \mid e \in l_i, 0 < \tilde{x}_i < 1\}|$ to be used for capacity branching. We only consider capacity branching for tracks with at least five fractional variables. Note that the parameter g is completely specified, given a track e and a branching capacity b because it must satisfy $g < \sum_{i \in C} \tilde{x}_i < g + 1$ to prevent a branching cycle. The estimates for the increase in the objective function value for the left and right branch are now calculated for every capacity b as

$$D^- := \frac{\sum_{i \in C} \tilde{x}_i - g}{\sum_{i \in C} \tilde{x}_i} \sum_{i \in C} k_i \tilde{x}_i \qquad D^+ := \frac{g + 1 - \sum_{i \in C} \tilde{x}_i}{\sum_{i \in C} \tilde{x}_i} \sum_{i \in C} k_i \tilde{x}_i,$$

for both branches, respectively. The branching capacity b is selected to be such that the value $\min\{D^-, D^+\}$ is highest.

4. Computational Results

The effectiveness of the techniques described in the previous sections is examined using five test instances. Characteristics for these instances can be found in Table 1. The graphs of SP97AR and SP971C are shown in Figure 5. The last four instances are also described by Bussieck (1998). However, the instances and models are not identical. Therefore, comparing results is not useful. The last two characters in the

Table 1 Instance Description

Instance	SP97AR	SP97IC	SP98AR	SP98IR	SP98IC
#stations	141	40	118	44	41
#tracks	177	52	134	44	46
#lines	1,212	831	913	420	627
Set F	{1, 2, 3, 4}	{1, 2}	{1, 2, 3, 4}	{1, 2}	{1, 2}
Set C	{1, ..., 5}	{3, ..., 15}	{2, ..., 10}	{3, ..., 12}	{3, ..., 15}

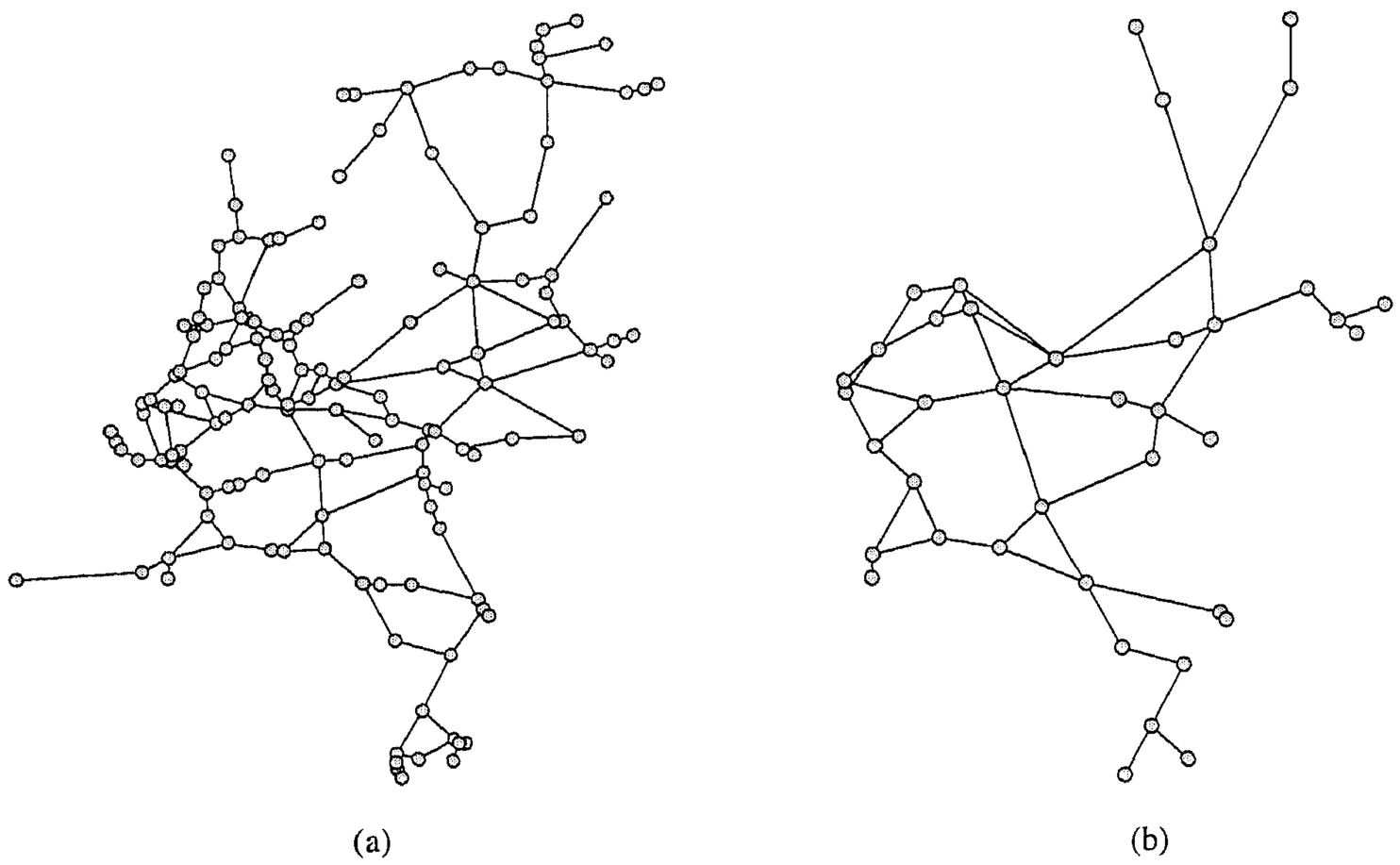


Figure 5 The Graphs for the Instances SP97AR (5(a)) and SP97IC (5(b)).

names of the instances give the train type that is considered. All techniques have been implemented in the branch-and-cut framework ABACUS (Thienel 1995, ABACUS 1998). The linear programming relaxations arising in the subproblems of the branch-and-cut tree are solved by CPLEX 6.6.1. All computations were done on an Intel Pentium III 866 Mhz PC with 128 MB internal memory running Windows 98SE.

The network reduction methods as described in Claessens et al. (1998) were already applied to the mentioned instances. The results of the preprocessing techniques for reducing the number of variables and constraints are given in Table 2. The initial number of constraints reflects the two service constraints for every track.

The effectiveness of the different classes of cutting planes is measured in two steps. First, we consider the effect on the root problem as shown in Table 3. The percentages between brackets show what fraction of the gap between the best-known upper bound and the initial lower bound is closed. The best-known upper bounds are obtained from the branch-and-cut process. Table 3 shows that, at least for the root node, the coefficient-strengthening cuts are superior. The effects of other classes of cuts are similar. The combined results for all classes of cuts together still show a large increase in the percentage of gap closed compared to the individual results.

Before presenting the branch-and-cut results, let us review the three proposed branching rules of §3.3, together with the standard variable branching. To analyze the effect of a specific branching rule, we keep track of the increase in the objective function value due to the application of the branching rule, i.e., before adding new cutting planes. All rules have been tested on SP98IC, with a maximum computation time of one hour and applying all classes of cutting planes. Table 4 shows both the average increase and the standard deviation. The overall effect of a branching rule in the tree search is measured by the increase of the overall lower bound at the end of the hour. A low standard deviation compared to the average increase is necessary for obtaining a balanced enumeration tree. See also Linderoth and Savelsbergh (1999) for more details. Table 4 illustrates this. The capacity-branching rule of §3.3.3 has by far the highest average increase. However, since the standard deviation of the increase for this rule is also high, the tree will be far from balanced. As predicted, the line-branching rule of §3.3.2 combines both a high mean increase with a relatively low standard deviation, and thus causes the enumeration tree to indeed be more balanced. That the overall effect of the variable branching rule is slightly better can be explained by the higher number of processed subproblems, due to the simplicity of this branching rule.

The branch-and-cut algorithm was tested on the five instances using both different classes of cutting planes and different branching rules. The results for all 55 problems are shown in Table 5. The cutting

Table 2 Preprocessing Results for Variable and Constraint Reduction

Instance	SP97AR	SP97IC	SP98AR	SP98IR	SP98IC
#var. initially	24,240	21,606	32,868	8,400	16,320
#var. after (3.1.1)	14,101	12,497	15,065	3,651	10,894
#con. initially	354	104	268	88	92
#con. after (3.1.2)	181	60	191	65	63
Size reduction (%)	70	67	67	68	54

Table 3 The Objective Function Values of the Best Solution Along with the Different LP Relaxations at the Root Problem when Applying Only a Specific Class of Cutting Planes

	Best UB	No cuts	All cuts	(3.2.1) (%)	(3.2.4) (%)	(3.2.2) (%)	(3.2.3) (%)
SP97AR	6,728	6,456	6,526 (26)	6,499 (16)	6,486 (11)	6,475 (7)	6,472 (6)
SP97IC	4,302	4,169	4,221 (39)	4,204 (26)	4,183 (11)	4,173 (3)	4,191 (17)
SP98AR	5,307	5,151	5,244 (60)	5,219 (44)	5,190 (25)	5,193 (27)	5,193 (27)
SP98IR	2,182	2,115	2,159 (66)	2,145 (45)	2,140 (37)	2,128 (19)	2,141 (39)
SP98IC	4,495	4,364	4,443 (60)	4,413 (37)	4,413 (37)	4,375 (8)	4,385 (16)

planes are tested using all classes, only the coefficient-strengthening cuts, and the coefficient-strengthening cuts combined with either the flow-cover cuts, the two-cover cuts, or the multicover cuts. All combinations of cuts are tested using only variable branching ("Var.") and using variable branching combined with line branching ("V + L"). When applying all classes of cuts, we additionally tested all four branching rules ("All") at the same time. The last column shows the performances of the ILP solver of CPLEX 6.6.1 for every instance, without applying any of the techniques described in this paper.

We enforced a maximum computation time of two hours. Computation times are only reported when an instance is solved within this time bound. For every problem, the table shows the best feasible (integer) solution that was found ("UB"), the remaining over-all lower bound ("LB"), the implied gap ("Gap"), the total number of created nodes in the tree ("#Nodes"), and the number of nodes that has already been processed ("#Done"). The best upper and lower bounds for every instance are typed in bold. The only instance that can be solved to optimality is SP98IR. An optimal solution for their version of this instance was already found by Bussieck (1998).

From Table 5 it is clear that the best lower bounds within two hours are obtained using all classes of cuts simultaneously, even though the number of subproblems that can be processed in this time is lower compared to using only a subset of the classes. Again we see that the effect of using line branching on the balance of the enumeration tree does not outweigh the larger number of subproblems that can be processed when using only variable branching. This is illustrated best by SP98IR. Solving this problem to optimality using variable branching requires more than 20% more nodes, yet the solution time is lowest.

The differences in the reported gaps after two hours are not only a result of the differences in the overall lower bounds ("LB"). Also, the best-known solutions differ significantly in some of the instances. This can best be explained by the fact that these solutions are mostly found using the primal heuristic of §2.3.2. Because the LP solution is input for the heuristic, results can vary when different classes of cutting planes are used.

Finally, the results in Table 5 show that our branch-and-cut algorithm outperforms the off-the-shelf ILP solver of CPLEX. For all five instances and for all the tested combinations of cutting planes, both the upper and lower bounds are significantly better than those obtained with CPLEX. For example, instance SP98IR, which can be solved to optimality in around three minutes, cannot be solved by CPLEX within the time limit of two hours.

Table 4 The Increase in the Lower Bound Due to the Branching Rule

	Mean (%)	Std. dev (%)	Init. LB	After 1 h	Increase (%)
Variable	3.8 (7)	4.8 (9)	4,443	4,465	22 (42)
SOS (3.3.1)	2.0 (4)	4.0 (8)	4,443	4,458	15 (29)
Line (3.3.2)	3.6 (7)	3.4 (6)	4,443	4,464	21 (40)
Cap (3.3.3)	16 (32)	27 (52)	4,443	4,451	8 (15)

5. Summary and Conclusions

In this paper we have outlined a branch-and-cut approach for solving the problem of allocating lines to passenger flows. Two integer programming models were given for this problem in Claessens et al. (1998) and Bussieck (1998). Where these previous papers use branch and bound and cut and branch, respectively, to solve the models, we have switched to branch and cut. The algorithm is described by introducing several classes of preprocessing rules (§2.1), cutting planes (§2.2), and branching rules (§2.3). These techniques have been tested on five real-life instances of NS. From these tests we can conclude that the described techniques perform very well on practical instances and significantly better than the ILP solver of CPLEX 6.6.1. While the preprocessing techniques considerably reduce the size of the initial problem, the mentioned classes of cutting planes effectively strengthen the LP lower bounds. Combined with the primal heuristic, we are now not only able to obtain excellent lower bounds, but also to find good primal solutions in reasonable time. Of the five test instances, we can prove to have the optimal solution of one instance. Of the remaining four instances, two were solved to within 1%, and two to within 3% of optimality.

Future research on the line-planning topic will involve the consideration of several train types simultaneously, without splitting the passenger flows a priori.

Table 5 Computational Results

	All cuts			(3.2.1)		(3.2.1) + (3.2.4)		(3.2.1) + (3.2.2)		(3.2.1) + (3.2.3)			
	All	Var.	V + L	Var.	V + L	Var.	V + L	Var.	V + L	Var.	V + L	Best	CPLEX
SP97AR													
UB	6,924	6,841	6,924	6,919	6,901	**6,728**	6,867	6,949	7,056	6,929	6,894	6,728	7,090
LB	6,547	**6,551**	6,547	6,534	6,531	6,550	6,545	6,537	6,532	6,536	6,533	6,551	6,497
Gap (%)	5.76	4.43	5.76	5.89	5.67	2.72	4.92	6.30	8.02	6.01	5.53	2.70	8.37
#Nodes	633	923	665	1,485	1,285	1,293	1,027	1,005	803	1,339	1,039		
#Done	321	461	332	742	643	646	514	502	402	669	519		
SP97IC													
UB	4,308	4,304	4,337	4,314	**4,302**	4,325	4,307	4,319	4,309	4,316	4,312	4,302	4,694
LB	**4,244**	**4,244**	**4,244**	4,228	4,230	4,236	4,234	4,232	4,232	4,236	4,235	4,244	4,175
Gap (%)	1.51	1.41	2.19	2.03	1.70	2.10	1.72	2.06	1.82	1.89	1.82	1.37	11.05
#Nodes	1,176	1,519	1,029	2,503	2,473	2,291	2,425	1,633	1,271	2,155	1,609		
#Done	616	760	514	1,252	1,236	1,145	1,213	817	636	1,078	804		
SP98AR													
UB	5,438	5,380	5,311	5,308	**5,307**	5,312	5,317	5,439	5,459	5,314	5,313	5,307	5,647
LB	5,262	**5,263**	5,261	5,243	5,244	5,252	5,255	5,252	5,254	5,249	5,247	5,263	5,216
Gap (%)	3.34	2.22	0.95	1.24	1.20	1.14	1.18	3.56	3.90	1.24	1.26	0.84	7.63
#Nodes	635	929	707	2,074	1,655	1,819	1,479	1,019	757	1,375	1,097		
#Done	340	464	354	1,039	828	910	739	510	378	688	549		
SP98IR													
UB	**2,182**	**2,182**	**2,182**	**2,182**	2,183	**2,182**	**2,182**	**2,182**	**2,182**	**2,182**	**2,182**	**2,182**	2,231
LB	**2,182**	**2,182**	**2,182**	**2,182**	2,178	**2,182**	**2,182**	**2,182**	**2,182**	**2,182**	**2,182**	**2,182**	2,162
Gap (%)	0.00	0.00	0.00	0.00	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.1
#Nodes	236	293	239	5,821	9,033	665	689	1,601	2,827	2,925	3,283		
#Done	236	293	239	5,821	6,503	665	689	1,601	2,827	2,925	3,283		
Time	0:03:00	0:02:38	0:03:23	0:54:12	—	0:03:00	0:03:59	0:14:28	0:31:52	0:20:00	0:28:17		
SP98IC													
UB	4,501	4,499	**4,495**	4,518	4,511	4,506	4,508	4,510	4,503	4,509	4,512	4,495	5,008
LB	4,469	**4,470**	4,469	4,441	4,438	4,465	4,465	4,446	4,445	4,446	4,441	4,470	4,384
Gap (%)	0.72	0.65	0.58	1.73	1.64	0.92	0.96	1.44	1.30	1.42	1.60	0.56	12.36
#Nodes	1,488	2,567	1,685	2,833	2,923	2,827	2,921	2,493	1,941	2,835	2,835		
#Done	785	1,300	844	1,418	1,461	1,428	1,467	1,249	970	1,419	1,161		

Acknowledgment

This research was partly sponsored by the Human Potential Program of the European Union under Contract HPRN-CT-1999-00104 (AMORE).

References

ABACUS. 1998. *ABACUS, A Branch-And-CUt System*. Universität zu Köln, Universität Heidelberg, Germany.

Bussieck, M. R. 1998. Optimal lines in public rail transport. Ph.D. thesis, University of Braunschweig, Braunschweig, Germany.

Bussieck, M. R., P. Kreuzer, U. T. Zimmermann. 1996. Optimal lines for railway systems. *Eur. J. Oper. Res.* **96**(1) 54–63.

Bussieck, M. R., T. Lindner, M. E. Lübbecke. 2002. Cost optimal line planning: A combined linear/nonlinear integer programming approach. Technical report, Department of Mathematical Optimization, Braunschweig University of Technology, Braunschweig, Germany.

Caprara, A., M. Fischetti, P. Toth, D. Vigo, P. L. Guida. 1997. Algorithms for railway crew management. *Math. Programming* **79** 125–141.

Claessens, M. T., N. M. van Dijk, P. J. Zwaneveld. 1998. Cost optimal allocation of rail passenger lines. *Eur. J. Oper. Res.* **110** 474–489.

Dietrich, B. L., L. F. Escudero. 1994. Obtaining clique, cover and coefficient reduction inequalities as Chvatal-Gomory inequalities and Gomory fractional cuts. *Eur. J. Oper. Res.* **73** 539–546.

Gallo, G., F. Di Miele. 2001. Dispatching busses in parking depots. *Transportation Sci.* **79** 322–330.

Goossens, J. H. M., C. P. M. van Hoesel, L. G. Kroon. 2002. On solving multi-type line planning problems. METEOR Research Memorandum RM/02/009, University of Maastricht, Maastricht, The Netherlands.

Härte, F. L. 1995. Het verbeteren van dienstregelingen en het opstellen van lijnvoeringen. Master's thesis, Free University Amsterdam, Amsterdam, The Netherlands (in Dutch).

Linderoth, J. T., M. W. P. Savelsbergh. 1999. A computational study of search strategies for mixed integer programming. *INFORMS J. Comput.* **11**(2) 173–187.

Lindner, T. 2000. Train schedule optimization in public rail transport. Ph.D. thesis, Braunschweig University of Technology, Braunschweig, Germany.

Martello, S., P. Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley and Sons, New York.

Nachtigall, K. 1999. Periodic network optimization and fixed interval timetables. Ph.D. thesis, Deutsches Zentrum für Luft- und Raumfahrt, Braunschweig, Germany.

Odijk, M. A. 1997. Railway timetable generation. Ph.D. thesis, Delft University of Technology, Delft, The Netherlands.

Oltrogge, C. 1994. Liniënplanung für mehrstufige Bedienungssysteme im öffentlichen Personenverkehr. Ph.D. thesis, Technical University of Braunschweig, Braunschweig, Germany (in German).

Schrijver, A. 1993. Minimum circulation of railway stock. *CWI Quart.* **3** 205–217.

Schrijver, A., A. Steenbeek. 1994. Dienstregelingontwikkeling voor Railned (timetable construction for Railned). Technical report, CWI, Center for Mathematics and Computer Science, Amsterdam, The Netherlands (in Dutch).

Thienel, S. 1995. ABACUS, A Branch-And-CUt System. Ph.D. thesis, Universität zu Köln, Köln, Germany.

Zwaneveld, P. J. 1997. Railway planning, routing of trains and allocation of passenger lines. Ph.D. thesis, Erasmus Universiteit Rotterdam, Rotterdam, The Netherlands.