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

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

Jan-Willem Goossens * Stan van Hoesel † Leo Kroon ‡

August 29, 2001

Abstract

An important strategic element 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 is included.

Keywords: Integer programming; Branch and cut; Combinatorial optimization; Railway transportation

1 Introduction

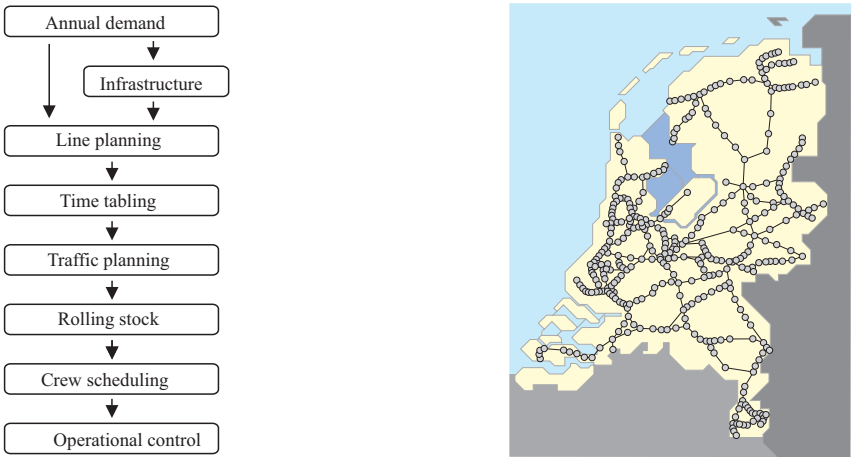
The planning problem faced by every railway operator consists of several consecutive stages, ranging from strategic decisions concerning e.g. infrastructure development, to real-time traffic control. Strategic problems are driven by estimates for the long-term demand. The first problem concerns the determination 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 [SS94, Odi97, Nac99], traffic planning (track assignment, platform assignment [Zwa97]), rolling stock planning [Sch93], and personnel planning.

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 (DLPP), i.e. the number of travelers that do not have to change trains during their journey, cf. [Bus98]. 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

*Dept of Quantitative Economics, University of Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands. E-mail: j.goossens@ke.unimaas.nl

†Dept of Quantitative Economics, University of Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands. E-mail: s.vanhoesel@ke.unimaas.nl

‡Rotterdam School of Management, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands. E-mail: l.kroon@fbk.eur.nl



(a) (b)

Figure 1: The different stages in the planning process (1(a)) and the Dutch railway network (1(b))

an inefficient use of resources. As an alternative objective, similar to [CDZ98] and [Bus98], this paper will focus on minimizing the operational costs of a line plan (CLPP).

There are several different approaches for formulating CLPP. In [CDZ98] it is proposed to formulate CLPP as a linear programming problem on binary variables (BLP). In [Bus98] an alternative formulation using fewer general integer variables (ILP) is presented. They compare the BLP and ILP formulation using a cutting plane algorithm, concluding that the much more compact ILP formulation is preferable for generating good feasible solutions, compared to the large BLP formulation. However, the lower bounds provided by the BLP are superior to the lower bounds of ILP, therefore the BLP formulation is preferable for proving optimality of a feasible solution ([Bus98]). By extending the reduction methods and classes of cutting planes presented in both articles, we show that the BLP formulation *can* be used to solve large instances of the problem using branch-and-cut.

In the next section the model formulation is presented. Section 3 explains the solution technology, and in section 4 we describe the computational study, based on instances of the Dutch railway operator NS (Nederlandse Spoorwegen).

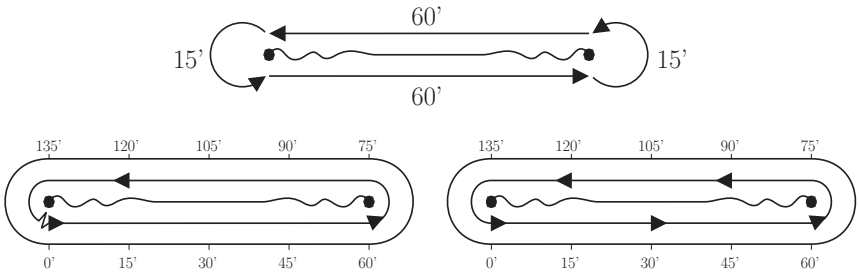
2 Model formulation

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 time table for the operated lines, though we assume that the time table will be cyclic with a cycle time of one hour, i.e. that the line plan is repeated every hour. The model described here will focus 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 cost 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. Together with the hourly frequency and the number of carriages per train allow us to determine the operational costs that can be attributed to operating a specific line.



(a) Once per hour (b) Twice per hour

Figure 2: Compositions vs. frequency.

Example 2.1
The required number of compositions 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 from its origin to its destination station of 60 minutes. 15 minutes are needed at either 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 (2(a)). However, operating our line at a frequency of two (2(b)), so 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 now a path in the graph. The lines and stations in the railway network are of exactly one of three types. 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 type of both 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 stations 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 route 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 [CDZ98].

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 [Olt94], this enables us to decompose the overall problem into separate line planning problems [Olt94]. Using this approach, we consider only line planning problems with exactly one type t of lines. We assume that for every type of line, there is exactly one type of carriage used to operate this line. Since 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 is given by the *origin/destination* (OD) matrix, specifying the number of travelers 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, given the OD matrix H with $h_{ij} = h_{ji}$ passengers from station i to j and vice versa, the number of passengers traversing edge e in the network can be given by $\underline{c}_e$. To provide maximum service to the passengers, we assume that each line is always operated in both directions. Thus, as in [CDZ98], we can regard lines as being undirected.

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 [Här95].

The level of service that the proposed line plan offers to the passengers, is ensured using two classes of restrictions. First, the capacities of the lines should suffice to transport all passengers. Second, the line plan is enforced to guarantee a high number of connections between every two stations in the network. We will consider this issue in more detail in the next subsection.

2.1 Formulation

Our formulation for the Cost optimal Line Planning model is very similar to the formulation used by [CDZ98]. Given the undirected graph $G = (V, E)$ with vertex set V and 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 between its origin and destination station are given by the stations v for which $\exists w \in V : \{v, w\} \in l\}$. 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 $F \subset \mathbb{N}$, the possible number of carriages by $C \subset \mathbb{N}$. The smallest and largest elements of these two sets are denoted by $f_{\min} = \operatorname{argmin}_{f \in F}$, $f_{\max} = \operatorname{argmax}_{f \in F}$ and $c_{\min}$ and $c_{\max}$ respectively. For every edge e in the network we are given the required number of passing trains per hour, and the number of passengers to be transported.

For formulating the line planning problem as an integer linear programming problem, we introduce binary variable for every $(l, f, c) \in N$, with the set of triples as $N := L \times F \times C$. Every $i \in N$ can be associated with a combination $(l_i, f_i, c_i) \in N$. 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 can now be formulated as follows.

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

$$\text{subject to} \quad x \in S^{LOP} \tag{2}$$

where the set S^{LOP} is defined as

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

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

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

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

The first set of constraints, (3), enforce a minimum of $\underline{f}_e$ passing trains per hour on every edge $e \in E$. This constraint is part of the constraints imposed in the model to assure a high degree of passenger service. Restrictions (4) impose a lower bound $\underline{c}_e$ on the number of carriages crossing edge e in one hour. Typically $\underline{c}_e$ will reflect the number of passengers wanting to traverse edge e per hour divided by the number of passengers per carriage. 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. Note that, contrary to [CDZ98] and [Bus98], we have dropped the upper bound restrictions on the cumulative frequencies per track. The infrastructural restriction that are modeled this way are only rarely binding, but more, since the strategical focus of this study is not to find bottlenecks in the railway network, we have decided to drop them. Unfortunately, this does not alter the complexity of CLPP.

The objective function coefficients k_i, i.e. the cost of operating the line associated to variable i, is split into fixed and variable cost. Variable cost in this sense are hourly costs per kilometer associated to operating the line. They involve e.g. energy and maintenance cost. The fixed costs are costs that are incurred for the availability of the individual trains and carriages and involve e.g. depreciation cost. 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 incorporates three classes of resource costs:

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

along with a parameter d_l specifying the length of path l ([CDZ98, Bus98]).

Claessens et al. [CDZ98] prove that finding a cost optimal allocation of trains is NP-hard by a reduction of the vertex cover problem.

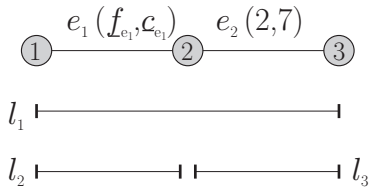


Figure 3: Example of a Cost optimal Line Planning instance.

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 pair of numbers in brackets behind the track names show the frequency and capacity lower bounds $\underline{f}_e$ and $\underline{c}_e$ respectively. The overall set of lines L for this network contains three lines $L := \{l_1, l_2, l_3\}$. We will 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.

3 Branch-and-Cut method for CLPP

The branch-and-cut techniques will be divided into three different sections: preprocessing (Section 3.1), Cutting planes (Section 3.2) and Tree search (Section 3.3). The preprocessing section focuses on reducing the initial size of the problem. This is done by removing superfluous variables and constraints and 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, subproblem selection and primal heuristics.

3.1 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 [CDZ98] and [Bus98] describe several preprocessing techniques, specific for the Cost optimal Line Planning problem. We will briefly recall these techniques and review them in a more general context.

3.1.1 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 [DE94].

The coefficient strengthening techniques will be described on the class of standard line planning problems (SLPP) with two types of constraints:

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

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$b$ 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 GUB 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 will derive several methods for strengthening problems of this form. Using notation similar to [DE94], the coefficient a_{kj} of row k and column j can in general be strengthened to

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

where $\Delta_{kj} = a_{kj} - b_k + Q_{kj} \geq 0$. Now, Q_{kj} can be interpreted as 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 $a_{kj} \geq b_k$ since 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 also $\hat{Q}_{kj} \leq Q_{kj}$ is valid. Alternatively, for any valid, though non-integer $\hat{Q}_{kj}$, we know that also $\lceil \hat{Q}_{kj} \rceil$ is valid. Both techniques will given lower bounds to the value of the following minimization problem.

Theorem 3.1
Consider a problem in standard form as defined in (7)-(10). Strengthening the coefficient a_{kj} of the constraint matrix A according to (11) with

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

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

Proof. We prove that $\forall x \in S : (b_k - Q_{kj}(h))x_j + \sum_{i \neq j} a_{ki} x_i \geq b_k$.
This is obviously valid for all $x \in S$ for which $x_j = 0$, since now the new coefficient of x_j is unimportant. For all $x \in S : x_j = 1$ 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 $x \in S : x_j = 1$ 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 will discuss two relaxations of this problem that give good bounds for Q for CLPP.

Example 3.1
Consider the set of integer points

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

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 $\{0, 0, \frac{1}{4}, 0, \frac{7}{8}\} \in \text{conv}\{S\}$ is cut off by the strengthened constraint.

The following corollary generalizes the idea of the previous example.

Corollary 3.1
Consider a problem in standard form as defined in (7). 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

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

where $r := \text{argmin}_{i|l_i \neq l_j, a_{hi} > 0} a_{ki}$, 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

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

■

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. The resulting continuous covering problem can, similar to the continuous knapsack problem, be solved to optimality using a greedy heuristic described later.

Example 3.2
Consider the feasible set S as given in Example 3.1. 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_{kj} with

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

is valid for S. Dropping the integrality restrictions on x_4 and x_5 allows us to use a greedy heuristic to solve this optimization problem. The optimal solution is $x_5 = 0.5$, $x_4 = 0$. This results in $Q_{13} = 3$, from which we know that

$$3x_1 + 4x_2 + 4x_3 + 4x_4 + 5x_5 \geq 5$$

is valid for S.

The following corollary generalizes this idea.

Corollary 3.2
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*

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

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

Proof. Clearly, (14) is a relaxation of the optimization problem in Theorem 3.1 and thus $Q''_{kj}(h) \leq Q_{kj}(h)$. ■

The greedy algorithm used to solve the continuous covering problem with non overlapping GUB constraints is very similar to the greedy heuristic for solving 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).

3.1.2 Variable reduction

We will 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 [CDZ98]. The variable reduction of [Bus98] uses similar techniques. Both parties however do not directly link 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 [Bus98], 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 3.1
Given an instance of SLPP *as defined in (7), 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*

$$c_i \leq c_j \qquad \text{and} \tag{15}$$

$$a_{ki} \geq a_{kj} \qquad \forall k \in \{1, \ldots, m\} \tag{16}$$

Proof. We can transform any feasible solution $\bar{x}$ in which $\bar{x}_j = 1$ into a feasible solution with $\bar{x}_j = 0$ and $\bar{x}_i = 1$. Feasibility is guaranteed by (16). Since the cost of i is less or equal to that of j, $c_i \leq c_j$, the value of the objective function in the new solution will not be higher than in the solution with $\bar{x}_j = 1$. ■

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 in the general upper bound constraints for the lines $l \in L$. The dominance technique described above is also used in [CDZ98], but without linking it to coefficient reduction techniques. Without coefficient strengthening, the model will not contain many dominated variables. The next example shows the contrary for a model whose coefficients have been strengthened.

Example 3.3
Let us consider the following instance of SLPP

$$\begin{array}{lrrrrrrrr} \min & 15x_1 & + 20x_2 & + 25x_3 & + 18x_4 & + 24x_5 & + 30x_6 & & \\ & 3x_1 & + 4x_2 & + 5x_3 & + 3x_4 & + 4x_5 & + 5x_6 & \geq 5 & \\ & x_1 & + x_2 & + x_3 & + x_4 & + x_5 & + x_6 & \geq 1 & \\ & x_1 & + x_2 & + x_3 & & & & \leq 1 & \\ & & & & x_4 & + x_5 & + x_6 & \leq 1 & \end{array} \tag{17}$$

with $x \in \{0,1\}^6$. *By applying the techniques from the previous section, we strengthen the coefficients in the first two constraints. Let us denote the coefficient of column* j *in row* i *in this problem by* a_{ij}. *We will use the order* $\{a_{11}, a_{21}, a_{12}, a_{22}, a_{13}, \ldots, a_{26}\}$ *for strengthening the coefficients, which is equivalent to variable by variable, first constraint 1, followed by constraint 2. For this coefficient order, the first two constraints can be replaced by*

$$\begin{array}{llllllll} 2x_1 & + 2x_2 & + 5x_3 & + 3x_4 & + 3x_5 & + 5x_6 & \geq 5 \\ 0x_1 & + 0x_2 & x_3 & + x_4 & + x_5 & + x_6 & \geq 1 \end{array} \tag{18}$$

Now it is clear that x_2 *is dominated by* x_1 *and* x_5 *by* x_4.

3.1.3 Constraint reduction

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

Theorem 3.2
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*

$$Q_{k0}(h) \equiv \min \sum_{i \in N} a_{ki} x_i \quad \text{subject to} \quad x \in P' \tag{19}$$

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

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 $x \in P' : \sum_{i \in N} a_{ki} x_i \geq Q_{k0}$. Since $S \subseteq P'$, this also holds for feasible solutions $x \in S$. ■

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

Example 3.4
Consider the feasible set S defined to contain all $x \in \{0,1\}^5$ satisfying

$$\begin{aligned} 3x_1 + 4x_2 + 7x_3 + 4x_4 + 7x_5 &\geq 7 \\ x_1 + x_2 + 2x_3 + x_4 + 1x_5 &\geq 2 \\ x_1 + x_2 + x_3 &\leq 1 \\ x_4 + x_5 &\leq 1 \end{aligned}$$

Defining the minimization problem (19) on the first two constraints, then $Q_{10}(2) = 7$ from the solution $(1,0,0,1,0)$. Thus, satisfying constraint 2 induces a left hand side value for the first constraint of 7 in the LP relaxation. Therefore, the first constraint is redundant.

Constraint dominance frequently occurs in CLPP instances, both between the capacity and frequency constraints of some track, as well as between service constraints on connecting tracks. Especially for dead-end tracks it often occurs that all lines covering this 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 former track are higher than that of the latter, then the latter is clearly redundant.

3.2 Cutting planes

Besides preprocessing, we will describe several classes of cutting planes that reduce X_{LP} as much to conv $\{X_{IP}\}$ as possible. The separation algorithms for these classes will be discussed in Section 4.2

3.2.1 Probing cuts

Let us describe this class of cutting planes on problems in standard form as defined in (7). We derive valid inequalities from information that is obtained from variable probing.

Example 3.5
Let us assume $\tilde{x}^*$ is the optimal solution to the LP relaxation 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*

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

Consider fixing x_1 to 1 in this constraint. We can now substitute x_1 out of the inequality, yielding

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

The left inequality is also valid in case $x_1 = 0$, since 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 Lemma 3.2.

Lemma 3.2
Consider an instance of SLPP *and some variable $j \in N$. Now, the inequality*

$$\sum_{i|l_i=l_j, a_{ki}>a_{kj}} a_{ki}x_i + \sum_{i|l_i\neq l_j} a_{ki}x_i \geq b_k - a_{kj} \tag{20}$$

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

Proof. If $\sum_{i|l_i=l_j} x_i = 0$, then (20) is obviously valid, since it is a relaxation of the original constraint k. If $\sum_{i|l_i=l_j} x_i = 1$, then

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

■

The way in which the cuts (20) are constructed preserves the problem specific structure of the initial inequality. Therefore, they can be used to generate new cuts of know classes of valid inequalities.

Corollary 3.3
Consider an instance of SLPP *and a subset of variables $F \subset N$. Now, the inequality*

$$\sum_{j\in F} \sum_{i|l_i=l_j, a_{ki}>a_{kj}} a_{ki}x_i + \sum_{i|\forall j\in F: l_i\neq l_j} a_{ki}x_i \geq b_k - \sum_{j\in F} a_{kj} \tag{21}$$

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

Proof. Consider applying Lemma 3.2 recursively for all $j \in F$. The resulting inequalities at every step are valid, thus after all $j \in F$, we arrive at (21). ■

3.2.2 2-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 3.6
Consider the following polytope S of a CLPP *instance, as also used in Example 3.1*

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

Since 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 3.3
For any instance of CLPP *the inequality*

$$\sum_{i \in C} x_i + \sum_{i \in \bar{C}} 2x_i \geq 2 \tag{22}$$

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

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

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

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 3.3 in which the variables are not divided into two but into several parts. In particular, the resulting multi covering (MC) cuts consider $n + 1$ subsets.

Corollary 3.4
Consider an arbitrary instance of CLPP *and some track* e. *Given a positive integer* n *for which* $2 \leq n + 1 < \underline{c}_e$, *then the inequality*

$$\sum_{s=1}^{n} \sum_{i \in C^s} s x_i + \sum_{i \in \bar{C}} (n+1) x_i \geq n + 1 \tag{23}$$

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

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, $s_i = \min\{s \in \mathbb{N} | b(s-1) \leq a_i n < bs\}$ is equal to $s_i = \min\{s \in \mathbb{N} | (b-\epsilon)(s-1) \leq a_i n \leq (b-\epsilon)s\}$ for suitably small $\epsilon > 0$. To prove that (23) is valid, we show that it is a Gomory cut with multiplication factor $n/(b-\epsilon)$. This is clear for the left hand side coefficients. It remains to show that $\lceil bn/(b-\epsilon) \rceil = n + 1$ for all applicable values of b and n. Clearly, $\lceil nb/(b-\epsilon) \rceil \geq n + 1$, yet, it is easy to show that $\lceil nb/(b-\epsilon) \rceil \leq \lceil nb/(b-1) \rceil \leq n + 1$: $\lceil nb/(b-1) \rceil - (n+1) = \lceil nb/(b-1) - (n+1) \rceil \leq \lceil n/(b-1) \rceil - 1 \leq 1 - 1 = 0$. ■

Example 3.7
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* $\bar{C} = \{5\}$. *Using this, the inequality*

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

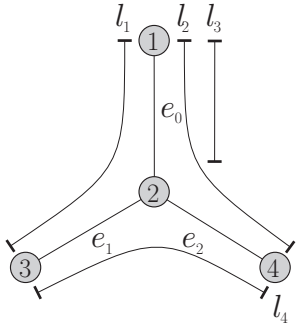
is valid for S.

3.2.3 Flow cover cuts

The flow cover cuts are described in Bussieck [Bus98]. These cuts use the line structure of the problem, in which lines typically cover more than one track.

Example 3.8
Consider the CLPP *instance shown in Figure 4. Assume $\tilde{x}^*$ is the optimal solution to the LP relaxation. Let us divide the three tracks into two sets, e.g. $\{e_0\}$ and $C := \{e_1, e_2\}$. Similarly, we can now divide all feasible solutions into two sets: those containing at least one line using only e_0, and those in which all lines passing e_0 also pass some tracks of C. Thus, either a valid line plan contains a line that uses only e_0, or validity implies that there will be at least $\underline{c}_{e_0}$ carriages per hour being pulled along the tracks of C.*

i	(l_i, f_i, c_i)	$\tilde{x}_i^*$
i_1	$(l_1, 1, 6)$	1.0
i_2	$(l_2, 1, 7)$	1.0
i_3	$(l_3, 1, 12)$	0.5
i_4	$(l_4, 1, 3)$	0.34



e	$(\underline{f}_e, \underline{c}_e)$
e_0	$(3, 19)$
e_1	$(1, 7)$
e_2	$(1, 8)$

$$(19 - 7 - 8)\tilde{x}_{i_3} + 1 \cdot 6\tilde{x}_{i_1} + 1 \cdot 7\tilde{x}_{i_2} + 2 \cdot 3\tilde{x}_{i_4} = 2 + 6 + 7 + 2 = 17 < 19$$

Figure 4: Flow cover example.

Conversely, if we extend all lines passing some tracks of C to also pass e_0, then lines that only pass e_0 merely fill the gap between $\underline{c}_{e_0}$ and $\sum_{e \in C} \underline{c}_e$. We can thus bound the capacity of variables using e_0, and not C to $f_i c_i \leq \underline{c}_{e_0} - \sum_{e \in C} \underline{c}_e$, or in this instance, to $(19 - 7 - 8) = 4$.

Lemma 3.4
Let $C \subset E$, $e_0 \in E \setminus C$ for which $\underline{c}_{e_0} > \sum_{e \in C} \underline{c}_e$. Then, the inequality

$$(\underline{c}_{e_0} - \sum_{e \in C} \underline{c}_e) \sum_{i \in N(e_0) | l_i \cap C = \emptyset} x_i + \sum_{e \in C} \sum_{i \in N(e)} f_i c_i x_i \geq \underline{c}_{e_0} \tag{24}$$

is valid.

Proof. First suppose $\sum_{i \in N(e_0) | l_i \cap C = \emptyset} x_i \geq 1$. Since 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. Now assume $\sum_{i \in N(e_0) | l_i \cap C = \emptyset} x_i = 0$. All lines that pass e_0 thus also use at least one track in E. This implies that

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

■

Corollary 3.5 identifies redundant choices for e_0 and E which is useful for the separation algorithm (Section 4.2).

Corollary 3.5
Consider a flow cover inequality defined by the pair (e_0, C) with $e_0 \in E$ and $C \subseteq E \setminus e_0$. If the capacity constraint for e_0 has been strengthened using the techniques described in Section 3.1, then flow cover inequalities with C such that no lines across e_0 cross any edge in C, i.e. $\{l | e_0 \in l, l \cap C \neq \emptyset\} = \emptyset$ cannot violate any feasible solution.

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, since $\{l \in L : e_0 \in l\} = \{l \in L : e_0 \in l, l \cap C = \emptyset\}$, this implies that $\sum_{i \in N(e_0), l_i \cap C = \emptyset} x_i \geq 1$. We can thus prove this corollary along the same lines as used in the proof of Lemma 3.4. ■

3.3 Tree search

As reported in [LS97], implementing problem specific branching can significantly reduce the size of the branching tree and thus speed up the solution process.

3.3.1 Branching

In general, the applied branching rule when considering binary problems is to branch on a single variable, thus creating two new sub problems. In this section we discuss several alternative rules.

Branching rules are used to split a problem into several, usually two, new sub problems. Similar to cutting planes in a class of cuts, every class of branching rules contains many possible branchings. Consider for example the class of variable branching, then out of all variables we have to select one variable to branch on. The quality of a branching rule 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. Determining the lower bounds for the possible new sub problems can be done exactly by explicitly solving the new LP relaxation (*strong branching*), or heuristically by determining estimates for the new objective values. In our implementation, we heuristically find a good candidate for every class of rules. We then use strong branching to compare the candidates of all classes of rules.

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 close 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

$$\sum_{i \in C} x_i \leq 1 \tag{25}$$

for $C \subseteq N$ with $x \in \{0,1\}^{|N|}$, is to branch on such a GUB constraint [LS97]. In any feasible solution, (25) ensures that at most one of the variables in C can be set to 1. Therefore, in general a valid branching scheme would be to create K branches in which subsequently for $k \in \{1, \ldots, K\}$ the variables in subsets $\bar{C}_k$ are forced to 0, or

$$x_i = 0 \text{ for all } i \in \bar{C}_k. \tag{26}$$

The sets $\bar{C}_k$ are subsets of C such that $\bigcup_{k \in K} C \setminus \bar{C}_k = C$. In most applications of GUB branching, only two new sub problems are created ($K = 2$). Now, instead of fixing a variable to 0 in all branches, we can use (25) to split the problem as follows

$$\sum_{i \in \bar{C}} x_i = 1 \qquad \text{versus} \qquad x_i = 0 \text{ for all } i \in \bar{C} \tag{27}$$

This splitting for $K = 2$ is stronger than using (26) and can be used for the set $\bar{C}$ and $C \setminus \bar{C}$. The problem is to determine the set $\bar{C}$. In special ordered set (SOS) branching an ordering of the variables in C is used to determine $\bar{C}$. In general, this ordering is given by weights a_i for all variables $i \in C$. Let us consider SOS branching for CLPP. 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 2 new problems as in (27) using $C := \{i | l_i = l\}$ and

$$\bar{C} := \{i \in C | f_i c_i \leq \delta\} \qquad \text{where } \delta := \sum_{i \in C} f_i c_i \tilde{x}_i. \tag{28}$$

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

Example 3.9
Consider a subproblem of the branching tree in which the optimal LP solution $\tilde{x}$ is fractional for some variables of line l. More specific, suppose $\tilde{x}_{(l,1,8)} = \tilde{x}_{(l,1,12)} = \frac{1}{2}$. The capacity of line l in this LP solution, is thus $\sum_{i|l_i=l} f_i c_i \tilde{x}_i = 1 \cdot 8 \cdot \frac{1}{2} + 1 \cdot 12 \cdot \frac{1}{2} = 10$ carriages per hour.

Line branching Similar to generalized upper bound branching rules, the line branching rule also splits a subproblem into two new sub problems 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

$$\sum_{i|l_i=l} x_i = 0 \qquad \text{versus} \qquad \sum_{i|l_i=l} x_i = 1 \tag{29}$$

The effect of line branching on the new sub problems 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 on branch allows us to obtain a higher 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

$$\sum_{i \in C} x_i \leq g \qquad \text{versus} \qquad \sum_{i \in C} x_i \geq g + 1 \tag{30}$$

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 (30) with $C := \{i \in N(e) | f_i c_i = b\}$. Since x is restricted to integer values, the two branches cover the complete solution space.

3.3.2 Sub problem selection

For the selection of the next open subproblem to be processed we use the well known best-first rule. Selecting the best subproblem from the list of open sub problems is done by reviewing the estimates for the objective function value of the LP relaxation. In the simplest case, these estimates are the value of the LP relaxation in the parent problem. These estimates will however postpone the fathoming of sub problems that will turn out to have an LP lower bound that is above the value of the current best known solution. To prevent this, we use the strong branching bounds that where calculated in the previous subsection for finding the best branching rule.

3.3.3 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 $L' := \{l \in L | \sum_{i|l_i=l} \tilde{x}_i > 0\}$. This significantly smaller problem is given to CPLEX, while bounding the available wall clock 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 5.

4 Implementation issues

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

4.1 Preprocessing implementation

All preprocessing techniques of Section 3.1 are applied at every subproblem of the enumeration tree. The implementation of the coefficient reduction techniques of Section 3.1.1 is not described here, but in Section 4.2.

4.1.1 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. Since 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.

4.1.2 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 3.2 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 sub problems of the enumeration tree, we only apply this technique on constraints h and k belonging to the same track.

4.2 Cutting planes implementation

Next, we describe the separation algorithms to find violated inequalities of the proposed classes of cutting planes. In every round, 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.

4.2.1 Coefficient strengthening cuts

The results of the coefficient strengthening techniques of Section 3.1.1 depend on the order of strengthening. The later a coefficient is strengthened, the small the effect of strengthening will be. We will 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 sub problems a new cut built up out of 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 valid $Q_{kj}(h)$, 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 determined by

$$\bar{a}_{kj} \leftarrow \min\{a_{kj}, (b_k - Q_{kj})^+\} \qquad \text{where } Q_{kj} := \max\{Q'_{kj}(k), Q'_{kj}(h), Q''_{kj}(h)\}. \tag{31}$$

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

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. In the second phase, we reuse this ordering of the lines, but now strengthen all coefficients of variables i with integer $\tilde{x}_i$.

The resulting cutting planes are added to the LP of the current subproblem. Since 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.

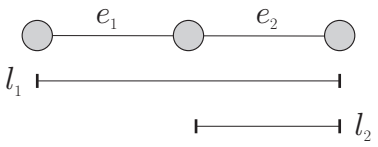
4.2.2 2-Cover cuts

The separation algorithm for the class of 2-cover cuts is straightforward, since there is only one 2-cover inequality for every track. However, substituting fixed variables out of the service

constraints, gives rise to new 2-cover (2C) inequalities. We also use the probing techniques from Section 3.2.1 for finding new violated 2-cover inequalities.

Example 4.1
Consider the instance with two tracks shown in Figure 5. 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*

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 5: Two cover: virtual setting

$x_j + 2 \cdot x_r + \ldots \geq 2$*. Clearly,* $\tilde{x}$ *satisfies this 2C cut. Now consider probing on* x_j*. This shows that substituting* x_j *out of the capacity constraint results in a valid inequality with* $\underline{c}_{e_2} = 10 - 5 = 5$*. The corresponding 2C cut* $2 \cdot x_r + \ldots \geq 2$ *is violated 50% by* $\tilde{x}$*.*

For every track e, the set of probed variables F as in Corollary 3.3 is build 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 $0 < \tilde{x}_i < 1$.

4.2.3 Multi cover cuts

The probing set $F = \{i | e \in l_e, \tilde{x}_i = 1\}$ is used for finding violated multi cover 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 multi cover constraint for n for which the violation is maximal.

4.2.4 Flow cover cuts

Solving the separation problem for flow cover cuts is done heuristically. As mentioned before, simply enumerating over all possible choices for e_0 and C is not an option. Corollary 3.5 shows us that e_0 and C should not be too far apart in the graph G. From the proof of Lemma 3.4 it is clear, given $\tilde{x}^*$ the solution to the LP relaxation, that violated flow cover cuts can only be found for tracks e_0 and C for which $0 < \sum_{i \in N(e_0) | l_i \cap C = \emptyset} x_i < 1$. The separation algorithm considers only all $e_0 \cup C \subseteq \delta(v)$ for all stations $v \in V$, where we define $\delta(v) := \{\{v, w\} : w \in V, \{v, w\} \in E\}$ for $v \in V$. Given e_0 and C, we can check violation in linear time.

4.3 Branching rules implementation

4.3.1 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 [LS97], 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 (28) and $C \setminus \bar{C}$. Note the possibility of a branching cycle, i.e. the current solution $\tilde{x}$ remains feasible in one of the new sub problems, implying that for this new problem, the previously chosen branching rule will again be best. This is prevented by ensuring that $\emptyset \neq \bar{C} \neq \{i \in C | \tilde{x}_i > 0\}$.

4.3.2 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 $\tilde{x}$:

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

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 $\min\{D^-, D^+\}$. This rule represents the multiple variable variant to the standard objective function based degradation estimate.

4.3.3 Capacity branching

Given an LP solution $\tilde{x}$ in a subproblem, we choose the track e with the largest number of fractional variables $|\{i | e \in l_i, 0 < \tilde{x}_i < 1\}|$ to be used for capacity branching. We only consider capacity branching for tracks with at least 5 fractional variables. Note that, given a track e and a branching capacity b, the parameter g is given, since it must satisfy $g < \sum_{i \in C} \tilde{x}_i < g + 1$. The estimates for the increase in the objective function value for the left and right branch are now calculated for every capacity b as

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

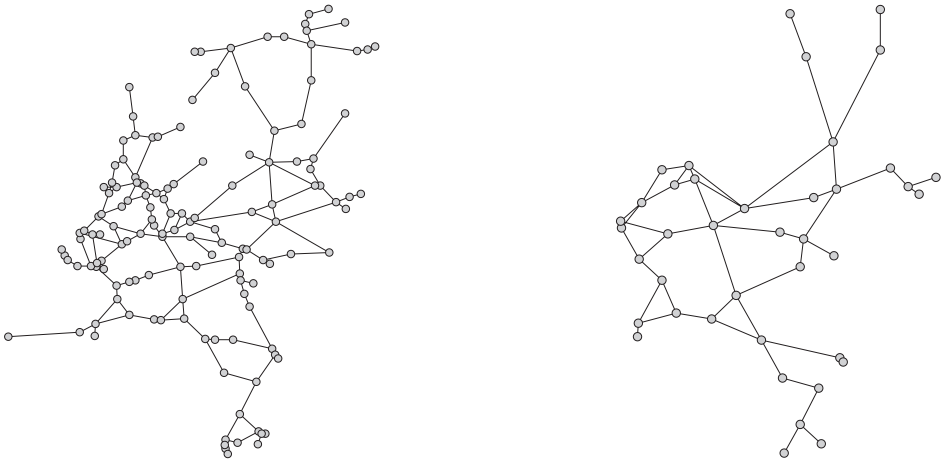
for both branches respectively. The branching capacity b is selected to be such that $\min\{D^-, D^+\}$ is highest.

5 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 last four instances are also described by [Bus98]. However, the instances and models are not identical. Therefore, comparing results is not useful. The last two characters in the names of the instances give the train type that is considered. All techniques have been implemented in the branch-and-cut framework ABACUS [Thi95, ABA98]. The linear programming relaxations arrising in the sub problems of the branch-and-cut tree are solved by CPLEX 6.6.1. The network reduction methods as described in [CDZ98] 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. Besides the number of variables, the table also shows the number of service constraints,

Instance:	SP97AR	SP97IC	SP98AR	SP98IR	SP98IC
#stations	141	40	118	44	41
#tracks	177	52	134	44	46
#lines	1212	831	913	420	627
Set F	$\{1,2,3,4\}$	$\{1,2\}$	$\{1,2,3,4\}$	$\{1,2\}$	$\{1,2\}$
Set C	$\{1,\ldots,5\}$	$\{3,\ldots,15\}$	$\{2,\ldots,10\}$	$\{3,\ldots,12\}$	$\{3,\ldots,15\}$

Table 1: Instance description.



(a) (b)

Figure 6: The graphs for the instances SP97AR (6(a)) and SP97IC (6(b)).

so initially, two constraints for every track. The effectiveness of the developed cutting planes is measured in two steps. First 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 gap closed compared to the individual results.

Before presenting the branch-and-cut results, let us review the three proposed branching rules of Section 4.3 together with the standard variable branching. To analyze the effect of a specific branching rule, we kept 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 [LS97]. Table 4 illustrates this. The capacity branching rule of Section 4.3.3 has by far the highest average increase. However, since the standard deviation

Instance:	SP97AR	SP97IC	SP98AR	SP98IR	SP98IC
#var. initially	24240	21606	32868	8400	16320
#var. after (4.1.1)	14101	12497	15065	3651	10894
#con. initially	354	104	268	88	92
#con. after (4.1.2)	181	60	191	65	63
size reduction	70%	67%	67%	68%	54%

Table 2: Preprocessing results for variable and constraint reduction.

	Best UB	No cuts	All cuts	(4.2.1)	(4.2.4)	(4.2.2)	(4.2.3)
SP97AR	6728	6461	6526 (24%)	6499 (14%)	6486 (9%)	6475 (5%)	6472 (4%)
SP97IC	4302	4169	4221 (39%)	4204 (26%)	4183 (11%)	4173 (3%)	4191 (17%)
SP98AR	5307	5154	5244 (59%)	5219 (42%)	5190 (24%)	5193 (25%)	5193 (25%)
SP98IR	2182	2117	2159 (65%)	2145 (43%)	2140 (35%)	2128 (17%)	2141 (37%)
SP98IC	4495	4367	4443 (59%)	4413 (36%)	4376 (17%)	4381 (11%)	4385 (14%)

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.

of the increase for this rule is also high, the tree will be far from balanced.As predicted, the line branching rule of Section 4.3.2 combines both a high mean increase with a relatively low standard deviation and thus causing the enumeration three 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 sub problems, due to the simplicity of this branching rule.

The branch-and-cut algorithm is tested on the five instances using both different classes of cutting planes, and different branching rules. The cutting planes are tested using all classes, only the coefficient strengthening cuts and the coefficient strengthening cuts combined with either the flow cover cuts, the 2-cover cuts or the multi cover cuts. All combinations of cuts are tested using only variable branching, and with using variable branching combined with line branching. In addition, when applying all classes of cuts we have also tested using all four branching rules. The results for all 55 problems are show in Table 5.

We enforced a maximum calculation 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 overall 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, for

	Mean	St.dev	Init. LB	After 1h	Increase
Variable	3.8 (7%)	4.8 (9%)	4443	4465	22 (42%)
Sos (4.3.1)	2.0 (4%)	4.0 (8%)	4443	4458	15 (29%)
Line (4.3.2)	3.6 (7%)	3.4 (6%)	4443	4464	21 (40%)
Cap (4.3.3)	16 (32%)	27 (52%)	4443	4451	8 (15%)

Table 4: The increase in the objective function due to the branching rule

		All cuts			(4.2.1)		(4.2.1) + (4.2.4)		(4.2.1) + (4.2.2)		(4.2.1) + (4.2.3)		
		All	Var.	V + L	Var.	V + L	Var.	V + L	Var.	V + L	Var.	V + L	Best
SP97AR	UB	6924	6841	6924	6919	6901	**6728**	6867	6949	7056	6929	6894	6728
	LB	6547	**6551**	6547	6534	6531	6550	6545	6537	6532	6536	6533	6551
	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%
	#Nodes	633	923	665	1485	1285	1293	1027	1005	803	1339	1039	
	#Done	321	461	332	742	643	646	514	502	402	669	519	
SP97IC	UB	4308	4304	4337	4314	**4302**	4325	4307	4319	4309	4316	4312	4302
	LB	**4244**	**4244**	**4244**	4228	4230	4236	4234	4232	4232	4236	4235	4244
	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%
	#Nodes	1176	1519	1029	2503	2473	2291	2425	1633	1271	2155	1609	
	#Done	616	760	514	1252	1236	1145	1213	817	636	1078	804	
SP98AR	UB	5438	5380	5311	5308	**5307**	5312	5317	5439	5459	5314	5313	5307
	LB	5262	**5263**	5261	5243	5244	5252	5255	5252	5254	5249	5247	5263
	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%
	#Nodes	635	929	707	2074	1655	1819	1479	1019	757	1375	1097	
	#Done	340	464	354	1039	828	910	739	510	378	688	549	
SP98IR	UB	**2182**	**2182**	**2182**	**2182**	2183	**2182**	**2182**	**2182**	**2182**	**2182**	**2182**	**2182**
	LB	**2182**	**2182**	**2182**	**2182**	2178	**2182**	**2182**	**2182**	**2182**	**2182**	**2182**	**2182**
	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%
	#Nodes	236	293	239	5821	9033	665	689	1601	2827	2925	3283	
	#Done	236	293	239	5821	6503	665	689	1601	2827	2925	3283	
	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	4501	4499	**4495**	4518	4511	4506	4508	4510	4503	4509	4512	4495
	LB	4469	**4470**	4469	4441	4438	4465	4465	4446	4445	4446	4441	4470
	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%
	#Nodes	1488	2567	1685	2833	2923	2827	2921	2493	1941	2835	2835	
	#Done	785	1300	844	1418	1461	1428	1467	1249	970	1419	1161	

Table 5: Computational results.

which an optimal solution was also found by [Bus98].

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 sub problems 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 balancedness of the enumeration tree does not outweigh the larger number of processable sub problems 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 Section 3.3.3. Since the LP solution is input for the heuristic, results can vary when different classes of cutting planes are used.

6 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 [CDZ98] and [Bus98]. Where these previous works 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 (Section 3.1), cutting planes (Section 3.2) and branching rules (Section 3.3). 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. While the preprocessing techniques significantly reduce the size of the initial problem, the mentioned classes of cutting planes effectively strengthen the LP lower bounds. Of the five test instances, we can prove to have the optimal solution of one instance. Of the remaining 4 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.

References

[ABA98] Universität zu Köln, Universität Heidelberg. *ABACUS, A Branch-And-CUt System*, October 1998.

[Bus98] Michael Bussieck. *Optimal lines in public rail transportation.* PhD thesis, University of Braunschweig, 1998.

[CDZ98] M.T. Claessens, N.M. van Dijk, and P.J. Zwaneveld. Cost optimal allocation of rail passenger lines. *European Journal of Operational Research*, 110:474–489, 1998.

[DE94] B.L. Dietrich and L.F. Escudero. Obtaining clueq, cover and coefficient reduction inequalities as chvatal-gomory inequalities and gomory fractional cuts. *European Journal of Operational Research*, 73:539–546, 1994.

[Här95] F.L. Härte. Het verbeteren van dienstregelingen en het opstellen van lijnvoeringen. Master's thesis, Free University Amsterdam, 1995.

[LS97] J.T. Linderoth and M.W.P. Savelsbergh. A computational study of search strategies for mixed integer programming. Technical report, Georgia Institute of Technology, 1997.

[Nac99] K. Nachtigall. *Periodic Network Optimization and Fixed Interval Timetables.* PhD thesis, Deutsches Zentrum für Luft- und Raumfahrt, Braunschweig, Germany, 1999.

[Odi97] M.A. Odijk. *Railway Timetable Generation.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1997.

[Olt94] C. Oltrogge. *Linienplannung für mehrstufige Bedienungssyteme im öffentlichen Personenverkehr.* PhD thesis, Technical University of Braunschweig, 1994.

[Sch93] A. Schrijver. Minimum circulation of railway stock. *CWI Quarterly*, 3:205–217, 1993.

[SS94] A. Schrijver and A. Steenbeek. Dienstregelingontwikkeling voor Railned (Timetable construction for Railned). Technical report, CWI, Center for Mathematics and Computer Science, Amsterdam, The Netherlands, 1994. In Dutch.

[Thi95] S. Thienel. *ABACUS, A Branch-And-CUt System.* PhD thesis, Universität zu Köln, 1995.

[Zwa97] P.J. Zwaneveld. *Railway Planning, Routing of Trains and Allocation of Passenger Lines.* PhD thesis, Erasmus Universiteit Rotterdam, 1997.