Optimising halting station of passenger railway lines

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

April 9, 2004

Abstract

In many real life passenger railway networks, the types of stations and lines characterise the halting stations of the train lines. Common types are Regional, Interregional or Intercity. This paper considers the problem of altering the halts of lines by both upgrading and downgrading stations, such that this results in less total travel time. We propose a combination of reduction methods, Lagrangian relaxation, and a problem-specific multiplier adjustment algorithm to solve the presented mixed integer linear programming formulation. A computational study of several real-life instances based on problem data of the Dutch passenger railway operator NS Reizigers is included.

Keywords: Mixed integer programming; Railway problems; Lagrangian relaxation

1 Introduction

The planning problem faced by railway operators is a complex multi-staged problem. The individual decision problems range from the strategic line planning problem (see Bussieck [2], Goossens et al. [6, 7], Zwaneveld [16]) via the construction of timetables (see Nachtigall [10], Odijk [12], Peeters [13], Schrijver and Steenbeek [15]), to traffic planning (see Zwaneveld [16]), rolling stock planning (see Schrijver [14]) and personnel (see Caprara et al. [3]), and shunting planning (see Gallo and Di Miele [4]).

The line planning problems in Bussieck [2], Goossens et al. [6], Zwaneveld [16] and Goossens et al. [7] describe the decision problem of finding routes in the railway network on which trains are to be operated. However, the stations at which these lines halt are dictated by the types of the stations and lines, as we will describe later. These types are part of the problem input. In contrast, this paper considers the line plan to be given, but concentrates on altering the stops of the lines along their route to decrease the total travel time of passengers through the network.

First, in §2, we introduce new concepts such as the line-event graph. Then, in §3 we show how to formulate the problem of optimising halting station of passenger railway lines as a multi-commodity network flow problem with additional constraints and variables. Using Lagrangian relaxation, we show in §4 how to find lower bounds for this problem. To effectively use these bounds in a branch-and-bound framework, as described in §5, we introduce a number of preprocessing and tree search techniques, together with a problem-specific multiplier adjustment algorithm. Finally, in §6, we describe a computational study based on instances of the Dutch passenger railway operator NSR.

*Dept of Quantitative Economics, University of Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands. E-mail: j.goossens@t75.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

Figure 1: The Dutch railway network.

2 Modelling

The railway infrastructure of the stations and tracks is modelled as an undirected graph $G = (V, E)$ of stations (vertices) and tracks (edges). This graph is also called the *network graph*. As an example, consider the network graph shown in Figure 1. By definition, every track $e = \{v, w\} \in E$ represents the track connecting stations v and w of V.

In addition, we are given a set of operated lines L, i.e., a line plan. A line $l \in L$ is described by a route in the railway network between an origin and a destination station along which trains are operated at a given hourly frequency $f(l) \in \mathbb{N}$. The route that a line l follows through the network G is a simple path of edges. For ease of notation, we define this route simply as $l \subseteq E$. We also define the subset of lines $L(v)$ as all the lines that pass some station $v \in V$.

Even though the subsequent stops of a line, also called the line's halts or dwells, could be any arbitrary set of the stations it passes along its route, we consider the halts to follow a strict pattern. The stations at which a line halts are defined by the *type* of the line. This type comes from a set of types $T = \{1, \ldots, T_{\max}\}$. Not only is every line l associated with a type $t(l)$, also every station $v \in V$ is of a type from T. The halting pattern of l, i.e., the stations at which l halts, are defined by the line type $t(l)$, and the types of the stations along the line's route. The types reflect the sizes of the stations: type 1 for stations in villages, up to type $T_{\max}$ for stations serving large metropolitan areas. Most real-life railway instances consider three types of stations and lines. These are often referred to as regional (R) for type 1, interregional (IR) for type 2, and intercity (IC) for type 3. Note that some authors refer to Regional lines and stations as *Aggloregional*. The types of the stations in the instance in Figure 1 are indicated by the sizes of the nodes in the network.

As is common in the Netherlands, but also in most other countries, the halting patterns of train lines follow a simple ordering. Train lines of type 1 halt at all stations they pass. Lines of type 2 skip the small stations of type 1, but halt at all stations of type 2 and higher, etc. Throughout this paper we assume that the halting stations of lines follow this strict hierarchical pattern in which a line of type t halts at all stations along its route that are of type t, or higher. In practise, however, exceptions are sometimes made.

For most planning problems, the types of lines and stations in the network are given as part of the input. In contrast, this paper considers the types of the lines to be given, and concentrates on finding a type for every station. Since the type of a station determines the halts of the passing lines, these types also influence the travel times through the network.

As an example, consider some station v. The type of this station is important not only for the

people that want to travel by train to and from v, but also for the people that pass v on their journey. Because some of the passing lines may not halt there, the travellers starting or ending their trip at v are restricted in their possibilities by the type of v. On the other hand, upgrading v to a higher station type can have a negative influence on the travel time of people on a train line passing v, if the new type of v causes their train to halt there.

The models we describe for the station type optimisation problem (STOP) focus on finding an assignment of types to stations that minimises the total travel time of all passengers through the network. We assume that all passengers choose their route through the network in such a way that they minimise their individual travel time. To allow the passengers this freedom, we also assume that the capacities of the train lines are sufficient. This defines a complete traffic assignment.

2.1 The line-event graph

Without knowing the halting stations of the lines, the travel times of train lines and, therefore, of the passengers are clearly not known. Note that the passengers choose their own shortest path through the network. To model this structure, we can construct the *line-event graph*. In this graph, every possible halt, train change, etc, is represented. For example, the line-event graph contains two nodes for every station that a line passes, with two pairs of opposing arcs between these nodes: a pair with a positive length representing a halting event (the halt arcs), and a pair with length zero (for not halting). It also contains arcs representing the trips between consecutive stations, as well as arcs modelling the changing between trains at a station (consider already Figure 3). In this line-event graph, depending on an assignment of types to stations, only a subset of the arcs is available. The length of an arc represents the amount of time needed for the event. This concept is formalised below.

Consider the network graph $G = (V, E)$ and a set of lines L. The line-event graph $G^L = (V \cup V^L, A^L)$ is a directed graph that is defined on two node sets V and V^L. We first discuss the set of *line-event nodes* V^L. This set is defined as

$$V^L = \{(l, e, v) | l \in L, e \in E, v \in V : v \in e, e \in l\}.$$

The set V^L contains for every train line two nodes for every station it passes. The event tuple $i \in V^L$ is defined as the triple $v = (l_i, e_i, v_i)$. The nodes in the set V are used in G^L as general source/sink nodes for every station. We refer to these nodes as *station nodes*.

The overall arc set A^L of *line-event arcs* is defined as the union of five sets:

$$\begin{aligned} A_d^L(e) &= \{(i,j) | i,j \in V^L : l_i = l_j, e = e_i = e_j, v_i \neq v_j\} \\ A_c^L(v) &= \{(i,j) | i,j \in V^L : l_i \neq l_j, v = v_i = v_j\} \\ A_h^L(v) &= \{(i,j) | i,j \in V^L : l_i = l_j, e_i \neq e_j, v = v_i = v_j\} \\ A_n^L(v) &= \{(i,j) | i,j \in V^L : l_i = l_j, e_i \neq e_j, v = v_i = v_j\} \\ A_p^L(v) &= \{(i,v),(v,i) | i \in V^L : v_i = v\} \end{aligned} \tag{1}$$

for all edges e and vertices v. The set $A_d^L(e)$ models the possible driving trips across track e. We refer to these arcs as driving arcs. The changing of trains at v is modelled by the change arcs in $A_c^L(v)$. The possible halting events and *non*-halting events at a station v are given by the arcs in the sets $A_h^L(v)$ and $A_n^L(v)$ respectively. Finally, the possibilities for passengers to enter and leave trains at v are modelled as the passenger arcs in $A_p^L(v)$. The total set of all change arcs, i.e., the union of the sets $A_c^L(v)$ for all v, is denoted A_c^L. Analogously, we define the sets A_d^L, A_h^L, A_n^L and A_p^L for the other line-event arcs.

Note that the set of non-halt arcs $A_n^L(v)$ is a copy of the set of halt arcs $A_h^L(v)$. Note also the existence of arcs implied by these definitions. For example, the existence of two change arcs (i, j) and (j, k), both at some station v and between lines $l_i \neq l_k$ implies that there also exists a change arc (i, k) at v.

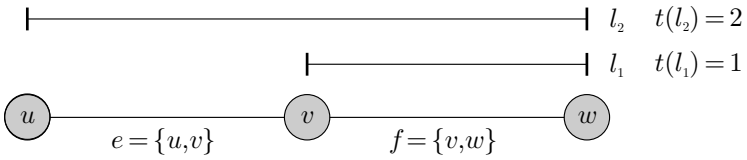


Figure 2: The network as a graph on three vertices and two edges, together with 2 lines.

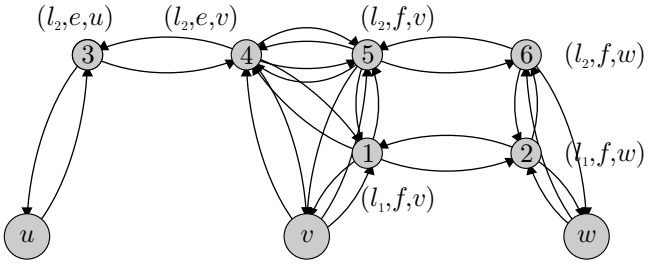


Figure 3: The line-event graph.

Example 2.1. *Consider the network graph $G = (V, E)$ with stations $V = \{u, v, w\}$ and tracks $E = \{e, f\}$ with $e = \{u, v\}$ and $f = \{v, w\}$ as shown in Figure 2. There are two lines l_1 and l_2 operated on this network. The first line uses only edge f and thus goes from v to w and vice versa, and is of type 1. The second line uses both tracks e and f and is of type 2.*

The line-event graph $G^L = (V \cup V^L, A^L)$ for this network is shown in Figure 3. The larger nodes are the nodes from $V = \{u, v, w\}$, the others are the line-event nodes from V^L.

For simplicity we have not only shown the triples (l_i, e_i, v_i) but we have also numbered these nodes. The arcs between nodes 1 and 2 represent the driving of line l_1 from station v to w and vice versa. There are two pairs of arcs between nodes 4 and 5. One pair represents the halting of line l_2 at v with a length of e.g. 2 minutes, either coming from u and going to w, or the other way around. The other two arcs between vertices 4 and 5 are the non-halt arcs of length 0. The arcs between vertices 1 and 4 represent the possibility for passengers to change from line l_1 to l_2 at v and vice versa. Finally, the arcs between node v and node 4 are there to model the possibilities for passengers to enter and to leave line l_2 at v.

All commodities use the line-event graph to find their (shortest) path through the network. However, as discussed earlier, the availability of the arc pairs in the line-event graph depends on the types of the stations.

Example 2.2. *Whether an event can occur, i.e., an arc can be used, depends on a given assignment of types. Consider the line-event graph of the previous example and the type-assignment vector $\bar{t}$ with $\bar{t}(u) = 2$, $\bar{t}(v) = 1$ and $\bar{t}(w) = 2$.*

Since line 2 is of type 2, this line does not halt at v when v is of type 1. Therefore, neither the pairs of change arcs, nor the two pairs of halt arcs are available. Thus, for station v, this

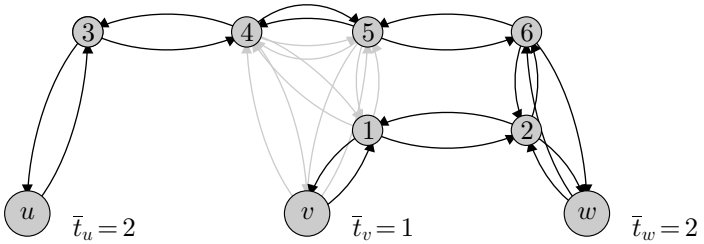


Figure 4: The line-event graph, given that station v is of type 1.

assignment allows only the arcs $(v, 1)$ *and* $(1, v)$ *and the non-halt arcs* $(4, 5)$ *and* $(5, 4)$ *to be used, as can be seen in Figure 4.*

To model the availability of arcs, we introduce the subgraph $G^L(\bar{t})$ of the line-event graph G^L for a given type-assignment vector $\bar{t}$ of types to the stations. The available arcs at station v, in case v is of type $t = \bar{t}_v$ are given by the sets

$$\begin{aligned} A_c^L(v,t) &= \{(i,j) \in A_c^L(v) : t(l_i) \leq t, t(l_j) \leq t\} \\ A_h^L(v,t) &= \{(i,j) \in A_h^L(v) : t(l_i) \leq t\} \\ A_n^L(v,t) &= \{(i,j) \in A_n^L(v) : t(l_i) > t\} \\ A_p^L(v,t) &= \{(i,v) \in A_p^L(v) : t(l_i) \leq t\} \end{aligned} \tag{2}$$

If station v is of type t, then the halt arcs, change arcs and passenger arcs of lines of type t and lower are available. On the other hand, only the non-halt arcs of lines of types strictly larger than t can be used. We also define the complementing sets, i.e., those arcs that are unavailable at a type strictly lower than t. For halt arcs, for example, these sets are given by $\bar{A}_h^L(v,t) = A_h^L(v) \setminus A_h^L(v,t)$.

Using these sets, we define the graph $G^L(\bar{t}) = (V \cup V^L, A^L(\bar{t}))$. This graph contains only the arcs of A^L that are available for a given assignment vector. In this definition $A^L(\bar{t})$ is the union of the above arc sets for $t = \bar{t}_v$ for all $v \in V$, together with the drive arcs of A_d^L. An assignment vector $\bar{t}$ is called *feasible* if between any pair of stations v and w, there exists a path from v to w in $G^L(\bar{t})$.

From these definitions we can make the following observations that are used throughout this paper.

Observation 2.3. *The sets of change arcs, halt arcs and passenger arcs are nested in* t *in the sense that for any station* v *it holds that*

$$A_r^L(v,t') \subseteq A_r^L(v,t) \qquad \text{for all } t' \leq t \text{ and } r \in \{c,h,p\}.$$

The opposite relation is true for the non-halt arcs:

$$A_n^L(v,t') \supseteq A_n^L(v,t) \qquad \text{for all } t' \leq t.$$

Observation 2.4. *The sets of halt and non-halt arcs at some station* v *are each other's complements:*

$$A_n^L(v,t) = A_h^L(v) \setminus A_h^L(v,t) \equiv \bar{A}_h^L(v,t) \qquad \text{for all } t \in T.$$

For a driving arc $(i,j) \in A_d^L(e)$, the length depends on the involved line l_j and edge $e = \{v_i, v_j\}$ of the network graph G. Thus, the lengths of pairs of driving arcs are symmetric. The driving times are part of the input.

A change arc $(i,j) \in A_c^L(v)$ has a length that only depends on the line l_j that is changed to. Given that line $l = l_j$ is operated at an hourly frequency of $f(l)$, we assume that these $f(l)$ trains per hour are evenly distributed across the hour, i.e., every $\frac{60}{f(l)}$ minutes. This leads to an expected change time of $\frac{60}{2f(l)}$.

The length of a halt arc $(i,j) \in A_h^L(v)$ of line l at station v is assumed to depend only on this line and station. The length of arc (i,j) is assumed to be less that $\frac{60}{2f(l)}$. All the arcs of A_n^L and A_p^L have length 0.

2.2 Valid paths in the line-event graph

Given a type assignment $\bar{t}$, a shortest path in $G^L(\bar{t})$ is not necessarily a realistic path for real life travellers. Using the graph of Figure 2, consider the path shown in Figure 5.

All the highlighted arcs are available for the assignment $\bar{t}$. Still, the path for travellers from u to w is invalid because it describes a path in which the passengers first change from line l_2 to l_1 using arc $(6,2)$, and then use the leaving arc $(2,w)$ to exit the system. We define valid paths as follows.

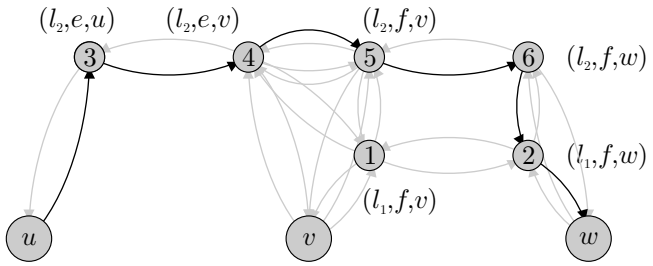


Figure 5: The highlighted path is not valid.

Algorithm 1 Find a shortest valid path.

Input: A graph $G^L(\bar{t})$, and two stations v and w.

1: Hide all passenger arcs (i, j) for which

$$i \in V \setminus \{v\} \qquad \text{or} \qquad j = v$$

2: Find the shortest path P from v to w that has the smallest number of arcs.

Definition 2.5. *A directed path $P = (a_1, \ldots, a_{2m+1}) \subseteq A^L$ from station $v \in V$ to station $w \in V$ is a* valid path *if and only if*

$$\begin{aligned} a_1, a_{2m+1} &\in A_p^L \\ a_{2i} &\in A_d^L && \text{for all } 1 \leq i \leq m \\ a_{2i+1} &\in A_c^L \cup A_h^L \cup A_n^L && \text{for all } 1 \leq i \leq m \end{aligned}$$

The arcs a_1 and a_{2m+1} are passenger arcs of A_p^L since, by construction, these are the only arcs incident to the station nodes $v \in V$.

Finding a shortest valid path is polynomially equivalent to finding a shortest path. Consider Algorithm 1. After Step 1, the only remaining passenger arcs are those that are outgoing arcs at v, and those that are incoming arcs at any station node other than v. The algorithm used in Step 2 to find the shortest path with the smallest number of arcs requires only a simple modification of the Dijkstra algorithm. Of all paths with minimal length this algorithm returns the one with the smallest number of arcs. We now show that the shortest path P is also valid.

Lemma 2.6. *The shortest path $P = \{a_1, \ldots, a_{2m+1}\}$ as calculated by Algorithm 1 is a shortest valid path.*

Proof. Hiding all outgoing passenger arcs at nodes other than v ensures the absence of internal passenger arcs in the shortest path from some station v to w. The assumption that our path P is a shortest path with the smallest number of arcs allows us to show that both arcs a_2 and a_{2m} can only be drive arcs of A_d^L. Since similar reasoning holds for arc a_2 and arc a_{2m}, we only discuss the latter case. Given that the arc $a_{2m+1} = (j, w)$ is a passenger arc, consider some arc $a_{2m} = (i, j)$, with $v_i = v_j$. Since, by construction, there also exists a passenger arc (i, w) of the same length as (j, w), we could replace the arcs a_{2m} and a_{2m+1} by (i, w) and thereby decrease the number of arcs in P.

The combination of a halt arc (i, j) and a change arc (j, k) in P can be ruled out by similar reasoning: by construction there exists a change arc (i, k) of the same length as (j, k). Replacing the sequence (i, j), (j, k) by (i, k) thus decreases the length of P by the (positive) length of (i, j).

For a sequence of two change arcs, the argumentation is very similar. Now, however, we use the property that the length of a change arc depends only on the line that is changed to, not on the feeder line. Thus, two change arcs (i, j) and (j, k) from lines l_i to l_j and from l_j to l_k can always be replaced by the arc (i, k).

To show that a pair of halt and non-halt arcs cannot occur consecutively in any path P in some $G^L(\bar{t})$, observe that for some line l at a station v both pairs of arcs cannot be present simultaneously. This is, by construction, also true for non-halt arcs and change arcs. □

Lemma 2.7. *For a line l and a station v, every valid shortest path contains at most one arc of l at station v, i.e., at most one arc $(i,j) \in A^L(v)$ for which $l = l_i$ or $l = l_j$.*

Proof. If a halt arc or a non-halt arc of l is used, then no other arcs of l can be used at v, since both line-event nodes have already been visited. In addition, the definition of a valid path tells us that at most one passenger arc at v can be used. Therefore, we are left with only two possibilities: a combination of a passenger arc and a change arc, or a combination of two change arcs. The first combination cannot be present in a shortest path, since also the passenger arc (of the same length) of the line that is changed from could have been used to exit the system. The usage of two change arcs can be ruled out by a similar exchange argument, since this valid path can be shortened by immediately changing from the first line to the last. □

3 Problem formulation

The number of travellers that want to travel from v to w, i.e., the demand for commodity $k = (v,w)$, is given by its entry H^{vw} in the origin-destination matrix H. By using the digraph $G^L(\bar{t})$ for some type assignment $\bar{t}$, we define the multi-commodity flow problem KSTOP($\bar{t}$) for routing the passengers (commodities) through the network as follows:

$$z_{\text{KSTOP}(\bar{t})} = \min \quad \sum_{k \in V \times V} dF^k \tag{3a}$$

$$\text{s.t.} \quad \mathcal{N}F^k = b^k \qquad \text{for all } k \in V \times V \tag{3b}$$

$$0 \leq F^k_{ij} \qquad \text{for all } k \in V \times V \text{ and } (i,j) \in A^L(\bar{t}) : i \notin V \setminus \{v^k\}, j \neq v^k \tag{3c}$$

where the variables F^k_{ij} represent the amount of flow of commodity k across arc (i,j) in the network. In this sense, F^k is for commodity k the vector of all flow variables for all arcs. The vector d gives the travel times for every line-event arc in $A^L(\bar{t})$. The matrix $\mathcal{N}$ is the node-arc incidence matrix of $A^L(\bar{t})$. Together with the column vector b^k of size $|V \cup V^L|$, it builds the flow balance constraints (3b). Note that for every commodity $k = (v,w)$, the vector b^k of size $|V|$ contains only two nonzero entries: a positive entry H^{vw} at v, and $-H^{vw}$ at w. The restrictions (3c) impose that the commodity $k = (v^k, w^k)$ can use arcs that are not outgoing from a station node, except v^k, and that are also not incoming arcs at v^k. The flow on arcs for which these conditions do not hold is set to 0. These constraints are identical to the restrictions in Algorithm 1.

Note the absence of capacity restrictions on the amounts of flow in (3). As mentioned in §2, we assume that the capacities of the train lines are sufficient to meet the passenger demand.

Lemma 3.1. *For every feasible solution F^k_{ij} of* KSTOP($\bar{t}$) *there exists a solution with the same cost per commodity, but for which for every commodity the arcs with positive flow make up a valid path through the network.*

Proof. Since there are no capacity restrictions on the arcs of the flow problem KSTOP($\bar{t}$) it follows that it can be solved by considering a separate single-commodity flow problem for every commodity, and that each of these problems amounts to finding a shortest path from the origin vertex to the destination vertex. The arcs-usage restrictions in (3c) allow the same arcs to be used as in Step 1 of Algorithm 1. It follows from Lemma 2.6 that if there exists a shortest path between a pair of nodes in this restricted graph of Step 1, then there exists a shortest valid path of equal length. □

As the model formulation lacks restrictions that tie the different commodities together, the minimisation problems for the different commodities are independent. Every commodity chooses

the cheapest route(s) according to the arc lengths d through the network. Because there are no capacity restrictions present in the STOP problem, these paths form a feasible and optimal solution.

Lemma 3.1 shows that an optimal solution to the multi-commodity flow problem of KSTOP($\bar{t}$) can be found by solving a shortest path problem in $G^L(\bar{t})$ for every commodity $k \in V \times V$, considering only the available arcs in (3c). Thus, the set of arcs with positive flow of the commodities with source u form a shortest path tree rooted at u. This leads to the following equivalent reformulation of KSTOP($\bar{t}$) called STOP($\bar{t}$). This formulation uses the flow variables F^u_{ij} for $u \in V$ that represent the flow across arc (i,j) that originates at u.

$$
\begin{aligned}
z_{\mathrm{STOP}(\bar{t})} = \min \quad & \sum_{u \in V} dF^u & & \\
\text{s.t.} \quad & \mathcal{N}F^u = b^u & & \text{for all } u \in V \\
& 0 \le F^u_{ij} & & \text{for all } u \in V \text{ and} \\
& & & (i,j) \in A^L(\bar{t}) : i \notin V \setminus \{u\}, j \neq u
\end{aligned}
$$

where again the vector d contains the costs per arc in $A^L(\bar{t})$. We use this reformulation to describe a model for STOP, i.e., optimising STOP($\bar{t}$) over the type assignments $\bar{t}$.

The next section introduces a formulation for solving $\min_{\bar{t}} \mathrm{STOP}(\bar{t})$ as a mixed-integer programming problem.

3.1 Extending the STOP($\bar{t}$) formulation

To model the variable type t of a station v, we introduce additional binary variables x^v_t, where

$$
x^v_t = \begin{cases} 1 & \text{if station } v \text{ is of type } t, \text{ and} \\ 0 & \text{otherwise.} \end{cases}
$$

Two new classes of constraints are used to link the station type variables to the flow variables, allowing only a flow that is consistent with the value of the x^v_t variables. This formulation is called STOP.

$$
\begin{aligned}
\min \quad & \sum_{u \in V} dF^u & & & \text{(4a)} \\
\text{s.t.} \quad & \mathcal{N}F^u = b^u & & \text{for all } u \in V & \text{(4b)} \\
& \sum_{u \in V} \sum_{a \in \bar{A}^L_r(v,t)} F^u_a \le M \sum_{t' > t} x^v_{t'} & & \text{for all } v \in V, t \in T \text{ and } r \in \{h,c,p\} & \text{(4c)} \\
& \sum_{u \in V} \sum_{a \in \bar{A}^L_n(v,t)} F^u_a \le M \sum_{t' < t} x^v_{t'} & & \text{for all } v \in V, t \in T & \text{(4d)} \\
& \sum_{t \in T} x^v_t = 1 & & \text{for all } v \in V & \text{(4e)} \\
& 0 \le F^u_{ij} & & \text{for all } u \in V \text{ and } (i,j) \in A^L(\bar{t}) : i \notin V \setminus \{u\}, j \neq v & \text{(4f)} \\
& x^v_t \in \{0,1\} & & \text{for all } v \in V, t \in T & \text{(4g)}
\end{aligned}
$$

We refer to the optimal objective function value of (4) as z_{STOP}. Constraint (4c) for halt arcs enforces that if a station v is of some type t or lower, in which case $\sum_{t' > t} x^v_{t'} = 0$, then no flow is allowed across the arcs in $A^L_h(v) \setminus A^L_h(v,t) \equiv \bar{A}^L_h(v,t)$, i.e., those that are unavailable at type t or lower.

Theorem 3.2. *The problems* STOP *and* $\min_{\bar{t}} \mathrm{STOP}(\bar{t})$ *are equivalent.*

Proof. The additional restrictions of (4c)-(4d) are necessary restrictions to hold for any feasible solution F of STOP($\bar{t}$) for a vector $\bar{t}$. Thus, the flow F automatically satisfies the additional restrictions of STOP with $x_t^v = 1$ for $t = \bar{t}_v$ and zero otherwise.

Vice versa, consider some feasible solution (F, x) of STOP. We construct a vector $\bar{t}$ such that F is feasible for STOP($\bar{t}$). Because of (4e), we set $\bar{t}_v = t$ for those $x_t^v = 1$. It remains to be shown that only for arcs $a \in A^L(\bar{t})$ the flow F_a^u is strictly positive for any station u. By contradiction, assume that the solution (F, x) has some $F_a^u > 0$ for an arc a at some station v for which $x_t^v = 1$, but with $a \notin A^L(v,t)$. The following argument can similarly be applied to (non-) halt, change and passenger arcs. Consider the case where a is a non-halt arc. It follows that $a \in \bar{A}_n^L(v,t)$ since $a \notin A^L(v,t)$. However, since by assumption $x_t^v = 1$, this implies that the right-hand side of (4c) is equal to zero, and thus $F_a^u \leq 0$. Clearly, this contradicts that (F, x) is feasible in STOP. □

The problem formulation STOP allows us to model the Station Type Optimisation Problem. The difficulty, however, lies in the number of variables needed in the modelling. With the number of arcs in the line-event graph G^L of order $\mathcal{O}(mk + nk^2)$, the number of flow variables F_{ij}^u is of order $\mathcal{O}(n \times (mk + nk^2))$, where $n = |V|$, $m = |E|$ and $k = |L|$. Even though this is still polynomial in n, m and k, the number of variables needed to model practical instances is enormous (see §6). The next section considers Lagrangian relaxation on the complicating constraints (4c)-(4d). This allows us to solve the STOP formulation by considering only small independent subproblems.

4 Lagrangian relaxation

Consider dualising the complicating additional constraints (4c)-(4d). Since these are the only constraints linking the flow variables of different origin stations, this makes the remaining problem separable. Therefore, let us study the following relaxation of STOP, referred to as LR(λ).

$$
\begin{aligned}
z_{\mathrm{LR}}(\lambda) = \min \quad & \sum_{u \in V} dF^u + \lambda \left(\sum_{u \in V} A(u)F^u - \sum_{v \in V} B(v)x^v \right) \\
\text{s.t.} \quad & \mathcal{N}F^u = b^u && \text{for all } u \in V \\
& Cx^v = 1 && \text{for all } v \in V \\
& 0 \leq F_{ij}^u && \text{for all } u \in V \text{ and} \\
& && (i,j) \in A^L(\bar{t}) : i \notin V \setminus \{u\}, j \neq u \\
& x_t^v \in \{0,1\} && \text{for all } v \in V, t \in T
\end{aligned}
\tag{5}
$$

where the matrix A is an m by $|V| \cdot |A^L|$ matrix, and B is an $m \times (|V| \cdot T_{\max})$ matrix, with m equal to the number of additional constraints. We define $A(v)$ and $B(v)$ as the submatrices of A and B that contain only the columns for some station $v \in V$. Thus, for example, $B(v)$ is of size m by $T_{\max}$, and holds for station v the big-M values of the restrictions that contain the x_t^v variables. Note that the only nonzero elements that $B(v)$ contains are at the restrictions for station v. The class of constraints $Cx^v = 1$ represents the constraints for choosing exactly one type for every station as in (4e). The vector $\lambda \in \mathbb{R}_+^m$ is a multiplier vector with one element for every of the m additional constraints. Lagrangian theory tells us that for any nonnegative vector λ, the value $z_{\mathrm{LR}}(\lambda)$ is a lower bound for the objective value z_{STOP} of the original problem. In particular, recall the following well-known theorem in optimisation theory.

Theorem 4.1 (Kuhn and Tucker [8]). *For a given vector λ, if the optimal solution of* LR(λ) *is feasible in the original problem, and if λ and the dualised constraints satisfy the complementary slackness conditions, then the solution is also optimal for the original problem.*

We use this lower bound in a branch and bound scheme, and we are therefore interested in the vector λ that maximises $z_{\mathrm{LR}}(\lambda)$. For a review of Lagrangian relaxation and duality, we refer

to Nemhauser and Wolsey [11, Page 323]. First, consider solving LR(λ) for a given λ. We can rewrite and solve LR(λ) as

$$z_{\mathrm{LR}}(\lambda) = \min\ z_{\mathrm{LRF}}(\lambda) - z_{\mathrm{LRX}}(\lambda)$$

with the two subproblems LRF(λ) and LRX(λ) defined as

$$\begin{aligned} z_{\mathrm{LRF}}(\lambda) = \min \quad & \sum_{u \in V} (d + \lambda A(u)) F^u & & \\ \text{s.t.} \quad & \mathcal{N} F^u = b^u & & \text{for all } u \in V \\ & 0 \leq F^u_{ij} & & \text{for all } u \in V \text{ and} \\ & & & (i,j) \in A^L(\bar{t}) : i \notin V \setminus \{u\}, j \neq u \end{aligned} \tag{6}$$

and

$$\begin{aligned} z_{\mathrm{LRX}}(\lambda) = \max \quad & \sum_{v \in V} \lambda B(v) x^v & & \\ \text{s.t.} \quad & C x^v = 1 & & \text{for all } v \in V \\ & x^v_t \in \{0,1\} & & \text{for all } v \in V, t \in T \end{aligned} \tag{7}$$

First consider the maximisation problem (7). Since the constraints $Cx^v = 1$ impose that for every station v we choose exactly one station type, this problem can be solved to optimality by inspection:

$$x^v_t = \begin{cases} 1 & \text{if } t = \operatorname{argmax}_{t' \in T}\ \lambda B(v,t'), \\ 0 & \text{otherwise.} \end{cases} \qquad \text{for all } v \in V \tag{8}$$

where $\lambda B(v,t')$ is the objective function coefficient of variable $x^v_{t'}$.

The subproblem (6) is also much easier to solve than the original problem. Given a vector λ, this problem is similar to STOP(t), but with all arcs in A^L. The objective function coefficients $d+\lambda A(u)$ make up a new vector of the arc costs per unit of flow originating from station v. However, the matrix $A(u)$ is based on the arc-presence in the sets $\bar{A}^L_r(v,t)$. Thus, for all pairs of stations u and u' the constraint matrices are equal: $A(u) = A(u')$. This is easy to see, since a flow variable F^u_a appears in the constraints based on the arc a for which it is defined, but irrespective of its origin station u.

If we define the new arc costs depending on λ as $d(\lambda) = d + \lambda A(\cdot)$, then the optimal routing in the multi-commodity flow problem can be found by solving *one* all-pairs shortest path problem using $d(\lambda)$ as arc costs. Given the shortest paths $P^{uw} \subseteq A^L$, the flows F can be reconstructed as $F^u_a = \sum_{w \in V : a \in P^{uw}} H^{uw}$.

In §5.2, we consider a standard subgradient based optimisation method, and a problem-specific method to find good multiplier vectors. First, however, consider the following well-known result by Geoffrion [5].

Theorem 4.2 (Geoffrion [5]). *If for all vectors $\lambda \geq 0$ the optimal value of $z_{\mathrm{LR}}(\lambda)$ is not altered by dropping the integrality conditions on its variables, then the value of the Lagrangian Dual problem* (LD)

$$z_{\mathrm{LD}} = \max_{\lambda \in \mathbb{R}^m_+} z_{\mathrm{LR}}(\lambda) \tag{9}$$

is equal to the optimal value of the linear programming relaxation of the original problem.

For the integrality restriction of (7), it is easy to see that all extreme points of $\{x \in \mathbb{R}^{|V| \cdot T_{\max}} : Cx = 1, 0 \leq x \leq 1\}$ are integer, since every variable is only part of exactly one of the constraints. Therefore, the following is true for the Lagrangian Dual LD.

Corollary 4.3. *The value of z_{LD} is equal to the value of the linear programming relaxation of STOP.*

This bound offers a way to test the quality of algorithms for finding a good multiplier vector λ.

5 The branch-and-bound algorithm

As mentioned in the previous section, STOP cannot be solved by only optimising the Lagrangian dual problem of (9). Therefore, we apply branch-and-bound on the x_t^v variables, using the Lagrangian relaxation LR(λ) for calculating lower bounds. The details of the branch-and-bound algorithm are described in three sections. First, we discuss preprocessing techniques. Then, the details of using the Lagrangian relaxation for deriving good dual bounds are presented. Finally, the section on tree search covers branching rules and primal heuristics.

5.1 Preprocessing

Strengthening the initial problem formulation is done on three levels. The next section discusses techniques for reducing the size of the line-event graph.

5.1.1 Line-event graph reduction

Identifying redundant lines in L can significantly reduce the size of the line-event graph.

Lemma 5.1. *Consider two distinct lines l and l' for which the path of line l' in the network graph G is contained in the path of l. If for every arc $a' = ((l', e_1, v_1), (l', e_2, v_2))$ of line l' the following holds*

$$d_a \leq d_{a'} \qquad \text{for all } a = ((l, e_1, v_1), (l, e_2, v_2)),$$

then line l' is dominated *by line l, and can be removed from the problem description.*

Proof. We show that every arc of line l' can be replaced by an arc of l that is not longer. This follows immediately from the fact that the path of l' is contained in the path of l, and the presence of arcs as implied by definition (1). Any solution (F', x) of STOP, that uses arcs of l', can thus be replaced by a solution (F, x), where the flow on arcs of l' is replaced by arcs of l. □

Due to the structure of the line-event graph, and the assumptions we have made concerning the lengths of its arcs, sufficient conditions for line dominance can easily be derived.

Corollary 5.2. *Consider two distinct lines l and l' for which the path of l' in the network graph G is contained in the path of l. If $t(l) = t(l')$ and $f(l) \geq f(l')$, then l' is dominated by l.*

5.1.2 Reducing the number of dualised constraints

The Lagrangian relaxation formulation presented in §4 is defined on the line-event graph, and the dualised constraints. The number of these constraints, i.e., the number of constraints that is relaxed, depends on the possible types $T(v)$ for the stations v in the network. These sets can be taken equal to the set of all available types T. However, the sizes of these sets can be reduced by using several intuitive methods. We discuss three such ideas.

Consider a station v and the lines $L(v) \subseteq L$ for which v is part of their route. Making v of a type that is higher than any of the lines is not useful. Similarly, a type of v that is lower than the lowest type of the lines in $L(v)$ would induce that no line halts at v.

The given set of operated lines implies that the start and end stations are available for these lines. In other words, the type of a station must be at least equal to the type of the lines that start or end there.

In case a station v is fixed to type $t \in T(v)$, then by definition the arcs of $A^L(v, t)$ are available at v. If, for some other type $t' \in T(v)$, the available set $A^L(v, t')$ is a subset of $A^L(v, t)$, then type t' can be removed from the set of available types of $T(v)$.

Fixing a station v at some type $t \in T(v)$ can make G^L unconnected. In this case, any assignment $\bar{t}$ with v at this type would be infeasible, and therefore t can be removed from $T(v)$. This holds in particular for the smallest type in $T(v)$.

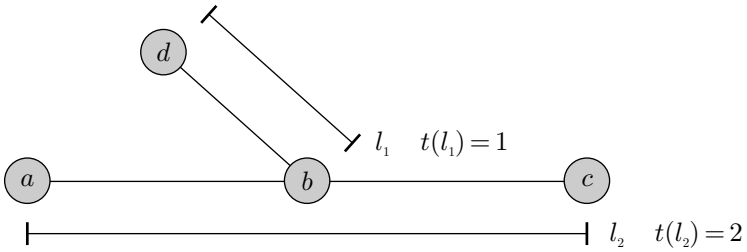


Figure 6: Setting station b to type 1 causes line l_2 not to halt at b.

Example 5.3. *Consider the following example with four stations $V=\{1,2,3,4\}$, three tracks $E=\{\{1,2\},\{2,3\},\{2,4\}\}$ and two lines, l_1 of type 1, operated between stations b and d, and l_2 of type 2 operated between a and c. Using the rules given above, we can set $T(2)=\{1,2\}$. However, if we consider the case where station b is of type 1, then this causes line l_2 not to halt at b. Thus, the travellers from b and d cannot reach a or c, and vice versa. Thus, type 1 is not a valid option for b.*

5.1.3 Coefficient reduction

We propose a formulation specific preprocessing rule to strengthen the *big-M* constraints of (4c)-(4d):

$$\sum_{u\in V}\sum_{a\in\bar{A}_h^L(v,t)} F_a^u \le M\sum_{t'>t} x_{t'}^v \qquad \text{for all } v\in V, t\in T(v) \tag{10}$$

$$\sum_{u\in V}\sum_{a\in\bar{A}_c^L(v,t)} F_a^u \le M\sum_{t'>t} x_{t'}^v \qquad \text{for all } v\in V, t\in T(v) \tag{11}$$

$$\sum_{u\in V}\sum_{a\in\bar{A}_p^L(v,t)} F_a^u \le M\sum_{t'>t} x_{t'}^v \qquad \text{for all } v\in V, t\in T(v) \tag{12}$$

$$\sum_{u\in V}\sum_{a\in\bar{A}_n^L(v,t)} F_a^u \le M\sum_{t'<t} x_{t'}^v \qquad \text{for all } v\in V, t\in T(v) \tag{13}$$

Given a station u and type t, note that the right-hand sides of the first three constraints are identical. In order to tighten the values of M, let us recall Lemma 2.6, i.e., the valid path lemma. From this we can show that non-halt arcs are the only arcs of which travellers originating from some station v can possibly use more than one arc at any station in the network. This is formalised in the following lemma.

Lemma 5.4. *A shortest valid path of a commodity contains at most one arc $a\in A^L(v)\setminus A_n^L(v)$ for every station $v\in V$.*

Proof. By contradiction, assume that two or more of these arcs are used. From Lemma 2.7 it follows that at least two lines are involved. Since at some station, v or other, the path changes from one line to another, this change could also have been made immediately at v, since the assumption of §2.2 tells that the time needed to change lines depends only on the line that is changed to. Thus, a valid path cannot be a shortest valid path if two or more arcs of $A^L(v)\setminus A_n^L(v)$ are used at $v\in V$. □

This lemma shows that any passenger can use at most one change or halt arc at any station. Therefore, we can merge (10) and (11) to

$$\sum_{u\in V}\sum_{a\in\bar{A}_h^L(v,t)} F_a^u + \sum_{u\in V}\sum_{a\in\bar{A}_c^L(v,t)} F_a^u \le M\sum_{t'>t} x_{t'}^v \qquad \text{for all } v\in V, t\in T(v) \tag{14}$$

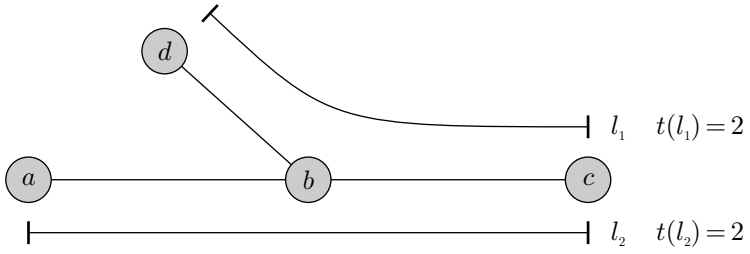


Figure 7: The shortest path from a to d can contain two non-halt arcs at b.

where, for a station v, for the value of M we can use the total number of travellers in the system except the people entering or leaving the system at u:

$$M \leftarrow \sum_{u,w \in V \setminus \{v\}} H^{uw} \qquad \text{for all } v \in V, t \in T(v) \tag{15}$$

Passengers entering or leaving at v use a passenger arc, and can thus not use any change or halt arcs according to Lemma 5.4.

For the passenger arcs at u, we already know from Lemma 5.4 that at most

$$M \leftarrow \sum_{u \in V} (H^{vu} + H^{uv}) \qquad \text{for all } v \in V, t \in T(v) \tag{16}$$

units of flow can pass on the arcs entering and leaving v in the line-event graph. Thus, the M for station v in constraint (12) can be replaced by this quantity.

The number of non-halt arcs used at a station can be more than one. Consider the following example.

Example 5.5. *Let us study an instance with four stations and two lines as shown in Figure 7. In case the type of station b is less than the type of l_1 and of l_2, then travellers that want to travel from a to d have to use two non-halt arcs at b.*

From Lemma 2.7 it follows that at most one of the non-halt arcs of a line at a station can be used. Therefore, the total flow across the arcs in $A_n^L(v,t)$ is at most

$$M \leftarrow |L(v)| \left(\sum_{u \in V} (H^{vu} + H^{uv}) + \sum_{u,w \in V \setminus \{v\}} H^{uw} \right) \qquad \text{for all } v \in V, t \in T(v) \tag{17}$$

Note that the right-hand side sum of elements of H is equal to the total number of travellers in the system.

We now derive bounds for the maximum amount of flow via station v, i.e., for $\sum_{u,w \in V \setminus \{v\}} H^{uw}$. The technique derives guaranteed maximum travel times through the network, and uses these to determine if stations in the network can ever be passed by travellers of a commodity in any feasible solution. First, we derive lower bounds on the travel times between stations in $G^L(\bar{t})$ for any assignment $\bar{t}$.

Lemma 5.6. *The shortest valid path distance between two nodes v and w in G^L is a lower bound for the shortest valid path distance between v and w in $G^L(\bar{t})$ for any assignment $\bar{t}$.*

Proof. Trivial, since $G^L(\bar{t})$ is a subgraph of G^L. □

Next, consider the shortest path distance in the following graph that is derived from the line-event graph. The graph D is equal to $G^L(\bar{t})$ for $\bar{t}_v = \min\{t \in T(v)\}$. Thus, D contains the arcs that are available in case all stations are assigned to their lowest type. In contrast to the arcs in the line-event graph, the non-halt arcs in D are of a length that is equal to the length of their (unavailable) halt counterpart in $\bar{A}_h^L(v, \bar{t}_v)$.

Lemma 5.7. *Consider two nodes v and w for which a shortest valid path in D exists, with length s. The length of the shortest valid path between v and w in $G^L(\bar{t})$ for any feasible assignment $\bar{t}$ is at most s.*

Proof. We prove this lemma by contradiction. Define P to be the shortest valid path from v to w in D. If a shortest valid v-w path P' in $G^L(\bar{t})$ is longer than s, then either some arc lengths in $G^L(\bar{t})$ are higher than in D, or P contains arcs that are not present in $G^L(\bar{t})$. From Observation 2.3 it is clear that, by construction, the only arcs of D that can be unavailable in $G^L(\bar{t})$ are non-halt arcs. However, the complementarity relation of Observation 2.4 shows that for every unavailable non-halt arc there is an available halt arc. Since, in D, these arcs are of the same length it follows that P' cannot be longer than P. □

This lemma leads immediately to the following corollary.

Corollary 5.8. *Consider two nodes v and w for which a shortest valid path in D exists, with length s. Any v-w path in some $G^L(\bar{t})$ that is strictly longer than s is not a shortest valid path in this graph.*

Using the lower and upper bounds on the lengths of the shortest valid paths, we can prove the following theorem.

Theorem 5.9. *Consider two nodes v and w for which a shortest valid path in D exists, with length s. If the shortest valid path in G^L from v to u, combined with the shortest valid path from u to w is strictly longer than s, then u cannot be part of the shortest valid path from v to w in $G^L(\bar{t})$ for any assignment $\bar{t}$.*

Proof. From the assumption, and Lemma 5.6 it follows that a similar path from v to w via u in any $G^L(\bar{t})$ is at least as long as that in G^L, and thus also longer than s. The remainder follows from Corollary 5.8. □

Thus, we can bound the number of travellers passing some station in any type assignment. Only travellers for whom the conditions in Theorem 5.9 for some station v do *not* hold can possibly travel via v. The total number of these passengers for station v is used in (15) and (17) to replace the previous bound $\sum_{u,w \in V \setminus \{v\}} H^{uw}$ on the number of travellers via v.

5.2 Bounding methods

The Lagrangian relaxation formulation is used to find lower bounds for the optimal objective value in a branch-and-bound setting. Since the size of the enumeration tree strongly depends on the quality of these lower bounds, this section describes a problem-specific algorithm for finding good multiplier vectors for the Lagrangian relaxation. This method is based on a sensitivity analysis of the arc costs. We have also implemented the subgradient-based Revised Volume Algorithm (RVA) as described in Bahiense et al. [1]. The algorithm can be found in Algorithm 2 in the Appendix. At the end of this section we discuss several issues related to using Lagrangian relaxation for bounding in general.

5.2.1 Problem-specific multiplier adjustment algorithm

As we have seen previously, solving the Lagrangian relaxation problem for a given multiplier vector λ can be done by solving two polynomially solvable problems. This section describes a technique for finding good values for λ, that is based on a sensitivity analysis of the shortest paths resulting from optimising the flow problem LRF(λ) in (6). We refer to this multiplier adjustment algorithm as MAA.

First, recall the structure of the dualised constraints. In the original formulation of (4), each restriction relates the flow across a set of arcs to the variables modelling the station types. Following the dualisation, the multiplier λ_r of constraint $r = 1, \ldots, m$ thus contributes to the cost of a group of arcs in the problem (5). For simplicity, we refer to this set of arcs as $A^L(r)$. Conversely,

a specific arc can be contained in the arc set of more than one constraint. The relation between the multipliers λ of the constraints and the cost per unit per arc is given by the vector $d(\lambda)$. The set of constraints that are part of the restrictions of some station u are given by the set $R(u)$.

For suitably small changes in λ, and so for small changes in the arc costs $d(\lambda)_a$, the optimal flow F of LRF(λ) does not change. Therefore, we propose to do a sensitivity analysis on the arc costs $d(\lambda)_a$. This analysis results in a lower bound δ_a^-, and an upper bound δ_a^+ for every arc a. If we consider one arc a, then the flow solution F is optimal for any cost of a between these two bounds, given that all other arc costs remain unchanged.

A sensitivity analysis on the cost of an arc a with respect to an optimal flow F is equivalent to an arc cost analysis for shortest paths. We present a specialised algorithm for performing this analysis. The special structure of the underlying problem gives us the opportunity to derive better (wider) bounds on the allowed changes in arc costs. Of particular use is the fact that the costs of arcs are not changed individually, but are altered as a consequence of a change in a constraint multiplier that affects a set of arcs at the same time. A second important aspect is the valid path property that must hold for the paths of travellers through the network. The proposed sensitivity analysis algorithm is outlined in Algorithm 3 in the appendix.

We assume that the optimal flow F does not change if the implied changes in all the arc costs remain within the sensitivity bounds. This allows us to model the effects of changes of the multiplier vector to changes in the value of the Lagrangian dual. The following linear program determines a good update $\lambda \leftarrow \lambda + \Delta$ by building the vector Δ station by station.

Given a flow F, a change Δ_r in the multiplier of some constraint r has a two-sided effect on the objective function value of (5). Firstly, through (6), it results in a change of

$$\Delta_r (AF)_r = \Delta_r \sum_{a \in A^L(r)} F(a) = \Delta_r c_r \qquad \text{with } c_r \equiv \sum_{a \in A^L(r)} F(a)$$

where $F(a)$ is the total flow of all origin stations on arc a.

Secondly, via (7) there can be a change in the optimal solution x. The new vector Δ appears in the objective function of (7). As shown in (8), the optimal solution to this problem is found by station-wise setting the x_t^u to one for which the objective function coefficient is highest.

The overall effect of a variable Δ_r on the two subproblems can be expressed using linear restrictions. For the constraints of a station v, the problem of finding the vector elements Δ that maximise the objective value of the Lagrangian relaxation is modelled as follows

$$\max \quad \sum_{r \in R(u)} c_r \Delta_r - Z \tag{18a}$$

$$\text{s.t.} \quad \delta_a^- \leq \sum_{r | a \in A^L(r)} \Delta_r \leq \delta_a \qquad \text{for all } a \in A^L(u) \tag{18b}$$

$$(\lambda + \Delta) B(u,t) \leq Z \qquad \text{for all } t \in T \tag{18c}$$

$$\Delta_r \in \mathbb{R}_+ \qquad \text{for all } r \in R(u) \tag{18d}$$

$$Z \in \mathbb{R}_+ \tag{18e}$$

The variable Z is used to model the optimal objective function value $z_{\text{LRX}}(\lambda)$ of (7). The constraints (18b) link the change in the multiplier of every constraint r to the upper and lower bounds δ_a^+ and δ_a^- for arcs on which r is defined. According to (8), the optimal solution for station u takes the type t that maximises $(\lambda + \Delta)B(u,t)$. The restrictions (18c) model this by ensuring that Z is at least as large as $(\lambda + \Delta)B(u,t)$ for any type t of station u. Since Z has a negative coefficient in the objective function of the maximisation problem here, this ensures that Z actually attains the correct value.

The bounds derived by the sensitivity analysis are only valid for changes in the costs of individual arcs. Changing the costs of many arcs, as we propose here, can therefore not guarantee an improvement. However, consider only the arcs $R(u)$ at station u. Lemma 5.4 shows that every valid shortest path can contain at most one of them, except for the non-halt arcs. Therefore, we

could propose to iteratively solve LR(λ) for the new vector $\lambda \leftarrow \lambda + \Delta$, for changes made to the arcs of only one station. However, extensive initial experiments have shown that solving for all changes at the same time gives better results. This procedure is shown in Line 30 and Line 34 of Algorithm 3.

5.2.2 Using Lagrangian relaxation for branch-and-bound

A solution (F, x) of LR(λ) for some vector λ is used in the primal heuristic of §5.3. Alternatively, in case the solution (F, x) is primal feasible, then such a solution is only optimal for the subproblem, if (F, x) is an optimal solution of LR(λ) and if the complementary slackness conditions hold. These are the well-known conditions under which a solution of a Lagrangian relaxation is also optimal for the original problem (see Geoffrion [5]).

The optimality condition is satisfied automatically by the algorithms used to solve LR(λ). In case the complementary slackness conditions are not satisfied, then this means that the Lagrangian relaxation lower bound resulted in a feasible solution, but there is no guarantee that this solution is the best feasible solution for this subproblem.

When some station u is branched on, then its x_t^u variables and the constraints $R(u)$ can be removed from the problem description. The remaining question is what happens to the lower bound in the child node. Let us compare the value of the lower bound for a vector λ^* in the parent node and in the child node.

Lemma 5.10. *For a given vector $\lambda \geq 0$, removing any redundant dualised restriction r from* LR(λ) *increases the value of the Lagrangian bound from $z_{\text{LR}}(\lambda)$ to $z_{\text{LR}}(\lambda')$ for λ' equal to λ with element $\lambda'_r = 0$.*

Proof. The objective function of LR(λ') is

$$\begin{aligned} z_{\text{LR}}(\lambda') &= \min_{F,x} \sum_{v \in V} dF^v + \lambda' (AF - Bx) \\ &= \min_{F,x} \sum_{v \in V} dF^v + \lambda (AF - Bx) - \lambda_r (AF - Bx)_r . \end{aligned}$$

Since r is redundant, it follows that $(AF)_r \leq (Bx)_r$ for any solution (F, x), and thus $z_{\text{LR}}(\lambda) \leq z_{\text{LR}}(\lambda')$ for any $\lambda \geq 0$. □

This lemma shows that removing any obsolete restrictions from the dualised constraints by fixing their multiplier to zero does not lower the optimal solution value of the minimisation problem for the Lagrangian relaxation.

Corollary 5.11. *Consider a parent and child node in the branch-and-bound tree for which a vector λ results in a lower bound of $z_p \equiv z_{\text{LR}}(\lambda)$ in the parent node. In the child node, after fixing the variables and removing the redundant constraints by setting the elements of λ to zero, the Lagrangian relaxation lower bound is at least z_p.*

5.3 Tree search

Next, we discuss the construction of the enumeration tree and the search process through it.

5.3.1 Branching

Branching rules are used to split a problem into several new subproblems that are easier to solve than the original problem. Normal variable branching for binary variables creates two new subproblems, one for every possible value of the binary variable. Alternatively, we propose to create at most $T_{\max}$ new subproblems, where in every new node a chosen station u is assigned to be of type t with $t \in T(u)$. Clearly, the station to use for branching can be chosen in an number of ways.

Branching rules influence the number of nodes of the branch-and-bound tree. Because the size of the enumeration tree strongly depends on how quickly the upper and lower bounds converge, we prefer a branching station that improves these bounds (see also Linderoth and Savelsbergh [9]). A difficulty of the STOP problem is that many of the variables—in particular, all x variables, and the flows on the non-halt arcs and passenger arcs—do not appear in the objective function (4a). We consider three branching rules, given a solution (F, x) of the Lagrangian relaxation problem for the multiplier vector λ.

Maximum unavailable flow branching For every station u, we consider the current amount of flow of F that would become unavailable if u where to be of type t in $T(u)$. The quality of branching on u is given by the smallest amount of unavailable flow over the types in $T(u)$. Thus, we branch on the station for which

$$\min_{t \in T(u)} \sum_{a \in \bar{A}^L(u,t)} F(a)$$

is maximal.

Constraint based branching Consider the vector $AF - Bx$ that represents the dualised constraints. A positive entry shows a violated constraint $AF \leq Bx$ of (5). Every constraint can be contributed to exactly one station. This branching rule proposes to branch on the station with the maximally violated constraint, i.e., the station u for which

$$(AF - Bx)_r$$

is maximal.

Estimated degradation branching Throughout the branch-and-bound tree, it is likely that a station is branched on more than once. To estimate the increase in the lower bound of the new child problems when branching on some station u, this branching rule uses the average increases over previous nodes where u was used for branching (see also Linderoth and Savelsbergh [9]). For every station u, we store the sum of the differences between the lower bound at the parent node, and the lower bound at the current node after fixing u to a certain type. This total improvement for u is reset after a fixed number of times u was branched on. Then, this estimate P_u is used to weigh the amount of unavailable flow, as in the first branching rule. We choose the station u for which

$$\min_{t \in T(u)} P_u \sum_{a \in \bar{A}^L(u,t)} F(a)$$

is highest. To initialise the estimate P_u, we iteratively fix every station u to the types $t \in T(u)$.

5.3.2 Primal heuristics

A primal feasible solution (F, x) is completely described by an assignment vector x of types to stations. We propose two algorithms to construct good assignments according to a solution (F, x).

The first primal heuristic is based on the flow solution F of LRF(λ). Every station is assigned the type t for which the amount of unavailable flow of F is smallest: $\bar{t}_u = \operatorname{argmin}_t \sum_{a \in \bar{A}^L(u,t)} F(a)$. This assignment is then used with STOP($\bar{t}$) to find a primal feasible flow and the total travel time needed.

Second, consider the assignment variables x_t^u that are produced by LRX(λ). From the construction of this solution in (8) it is clear that the solution x_t^u can also be represented by an assignment vector $\bar{t}$. Again, a primal feasible flow can easily be constructed from this assignment by solving STOP($\bar{t}$).

These fast primal heuristics are used at every node in the branch-and-bound tree to find good primal solutions.

	NS3600	NSNH	NSRandstad
# Stations	28	36	122
# Tracks	27	37	138
# Types	3	3	3
# Lines	14	14	64

Table 1: Basic statistics for the used instances.

	NS3600	NSNH	NSRandstad
# LR cons	120	176	432
# Lines	14	14	64
# Nodes	168	288	1308
# Arcs	1296	3074	19812
$z_{\mathrm{LR}}(\mathbf{0})$	4326	3221	3404

Table 2: Initial statistics before preprocessing. The numbers of nodes and arcs refer to the line-event graph. The value $z_{\mathrm{LR}}(\mathbf{0})$ is the Lagrangian bound for the zero multiplier vector.

6 Computational results

The proposed Lagrangian relaxation based branch-and-bound algorithm of §5 is tested using three real life instance of NSR. Table 1 shows some basic statistics for these instances, such as the number of stations, tracks and available types, as well as the number of lines in the instance. Figure 8 shows the network graphs for these instances. All computations were obtained on an AMD Athlon XP 2700+ with 1 GB internal memory running Linux, kernel 2.4.18, using CPLEX 8.1.

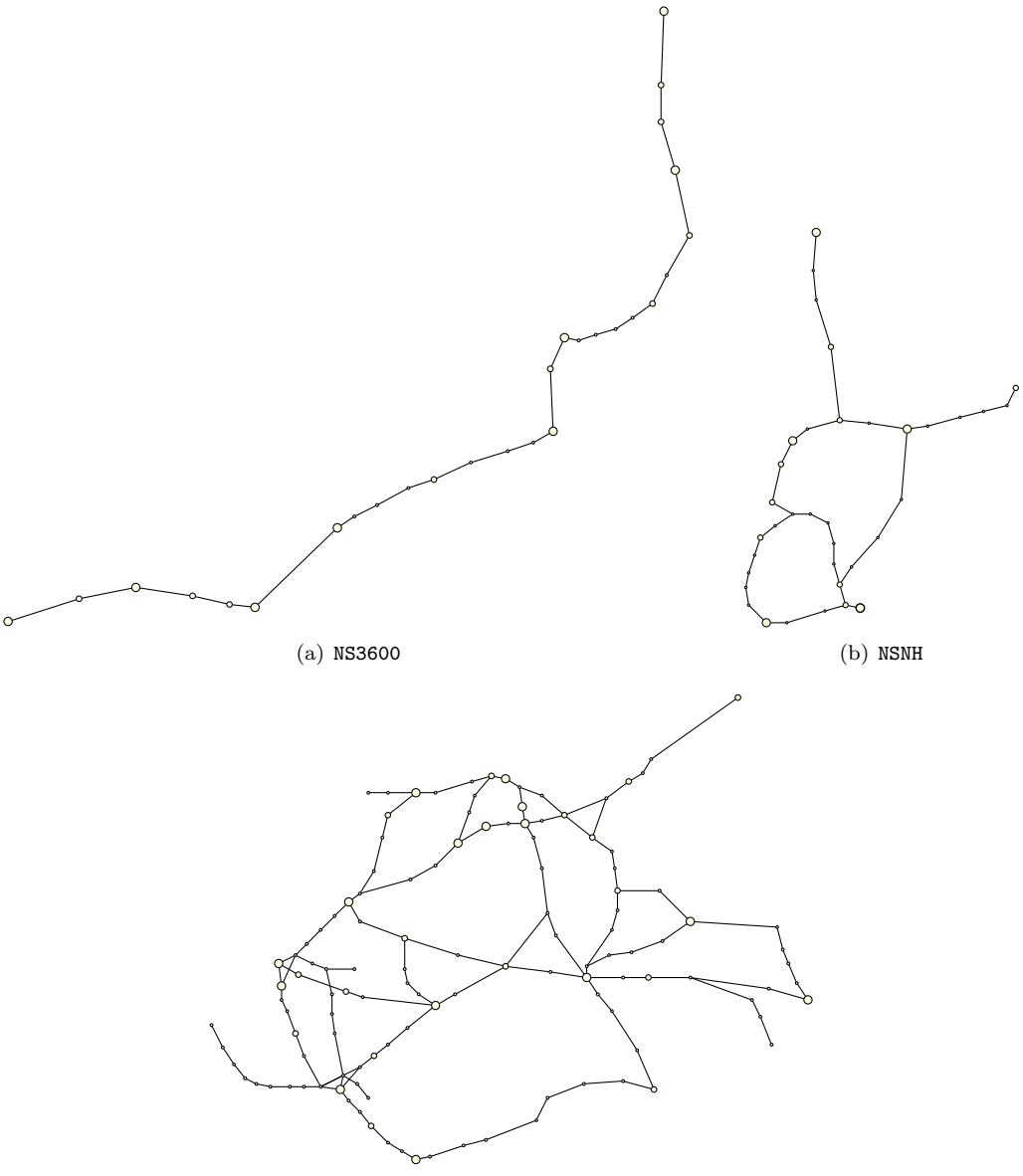
A short list of characteristics for the line-event graphs of these instances before preprocessing is shown in Table 2. The sizes of the line sets ('# Lines'), node sets ('# Nodes') and arc sets ('# Arcs') are presented. In addition, the table also gives the number of dualised constraints in the Lagrangian problem ('# LR cons'), and the values of the Lagrangian relaxation $z_{\mathrm{LR}}(\lambda)$ for $\lambda = \mathbf{0}$.

Using the preprocessing techniques described in §5.1, the sizes of the line-event graphs can be reduced considerably, as can be seen in Table 3. By eliminating redundant lines from the problem description, the number of nodes and arcs is reduced on average by around 27% and 50% respectively. The strength of the reduction techniques for the available types of stations is shown by the number of dualised constraints for the Lagrangian relaxation. For some of the stations only one available type remains. These stations can thus be fixed to this type a-priori ('# Fixed'). Note also the improvements in the values for $z_{\mathrm{LR}}(\lambda)$.

The line-event graphs for these instances are too large to have a useful visual representation.

We start our analysis of the reduced problems at the root node of the branch-and-bound tree. To test the effectiveness of the two techniques mentioned in §5.2, several combinations of them are tried at the first node. Apart from the two algorithms, we also test the zero multiplier vector, as an obvious trivial 'algorithm'. The statistics for the revised volume algorithm, the multiplier adjustment algorithm and the zero-vector ('Zero') are shown in Table 4. The column 'MAA, RVA' gives the results for first applying the MAA, and then using the resulting vector as input for the RVA. The column 'RVA, MAA' is similar, but for the reverse order. The last column shows the value of the linear programming lower bound, obtained from constructing and solving the LP relaxation of the original STOP model of (4). As was shown in Corollary 4.3, these LP bounds are tight upper bounds on the best possible values of the Lagrangian relaxation. The LP relaxation for the last problem, NSRandstad, turned out too large to construct. More information on the number of variables and constraints of STOP can be found in Table 8 in the Appendix.

For the first instance, the improvement of the lower bound ('z_{LR}'), compared to using just the zero-vector, is considerable. Here RVA takes much more time, but also slightly outperforms MAA.



(a) NS3600 (b) NSNH

(c) NSRandstad

Figure 8: The networks for the instances NS3600 (8(a)), NSNH (8(b)) and NSRandstad (8(c)).

	NS3600	NSNH	NSRandstad
# LR cons	60 (−50%)	126 (−28%)	321 (−26%)
# Fixed	10 (+36%)	6 (+17%)	57 (+47%)
# Lines	7 (−50%)	9 (−36%)	40 (−37%)
# Nodes	130 (−23%)	230 (−20%)	922 (−30%)
# Arcs	660 (−49%)	1773 (−42%)	9502 (−52%)
$z_{\text{LR}}(\mathbf{0})$	4939 (+14%)	3353 (4%)	3717 (+9%)

Table 3: Statistics after preprocessing. The first row shows the number of stations for which the type could be fixed a-priori. The numbers of node and arcs refers to the line-event graph. The value $z_{\text{LR}}(\mathbf{0})$ is the Lagrangian bound for the zero multiplier vector.

Instance		Zero	MAA	RVA	MAA, RVA	RVA, MAA	LP
NS3600	z_{LR}	4939	5019	5024	5019	5025	5026
	z_{STOP}	5497	5497	5497	5497	5497	
	Gap	11.30%	9.52%	9.41%	9.52%	9.39%	
	Time	0.24	0.47	4.14	1.59	4.32	32.95
NSNH	z_{LR}	3353	3368	3353	3368	3368	3370
	z_{STOP}	4252	4252	4252	4252	4252	
	Gap	26.81%	26.25%	26.81%	26.25%	26.25%	
	Time	0.31	0.87	2.37	3.01	3.09	165.95
NSRandstad	z_{LR}	3717	3717	3717	3717	3717	-
	z_{STOP}	4200	4200	4200	4200	4200	
	Gap	12.99%	12.99%	12.99%	12.99%	12.99%	
	Time	2.4	5.95	127.73	131.41	129.81	-

Table 4: Statistics for the root node of the branch-and-bound tree.

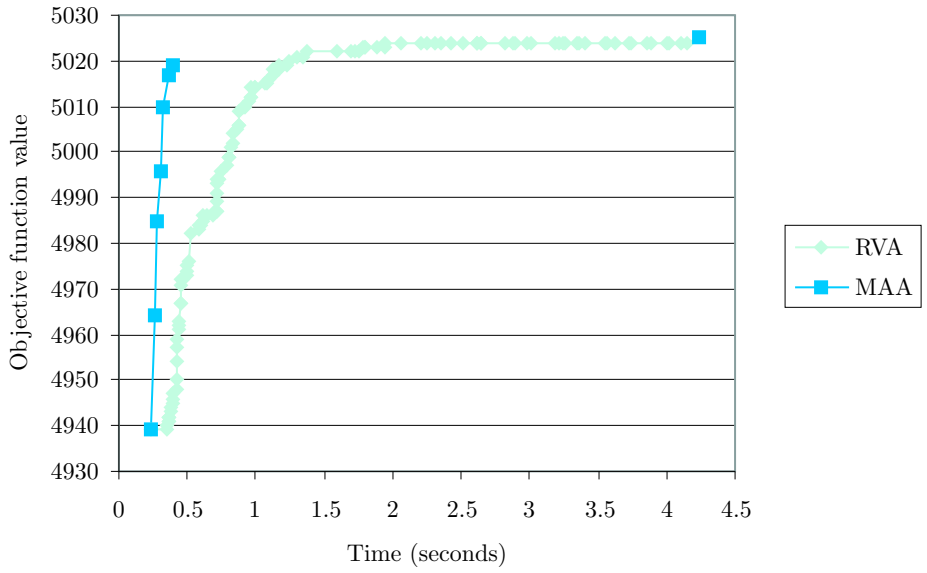


Figure 9: The improvement of the lower bound $z_{\text{LR}}(\lambda)$ over time (seconds), for NS3600 at the root node, for combinations of the improvement techniques.

Instance		Flow	Constraint	Degradation
NS3600	z_{LR}	5489	5489	5489
	z_{STOP}	5489	5489	5489
	Gap	0.00%	0.00%	0.00%
	# Nodes	389	377	331
	Time	6.8	6.68	6.51
NSNH	z_{LR}	4088	4139	4139
	z_{STOP}	4203	4139	4139
	Gap	2.81%	0.00%	0.00%
	# Nodes	240489	28333	18500
	Time	*	507.15	320.14
NSRandstad	z_{LR}	3972	3949	3979
	z_{STOP}	4135	4135	4138
	Gap	4.10%	4.71%	4.00%
	# Nodes	12900	12978	13752
	Time	*	*	*

Table 5: Branch-and-bound statistics for using the different branching rules on the three instances. An asterisk (*) indicates that the time limit of 3600 seconds was exceeded.

Combining RVA and MAA, in this order, further enhances the quality of the lower bound. This is shown graphically in Figure 9. The graph shows the value of the so far best lower bound as a function of time. The different nodes in the graph indicate the iterations of the two algorithms. For the RVA, the nodes represent only the iterations at which an improvement was made. The final bound is very close to the theoretical limit imposed by the linear programming relaxation. This is also true for the second instance, NSNH. However, for this instance RVA cannot improve upon the trivial lower bound, nor on the vector of MAA. Based on these findings, we use the combination of RVA, followed by MAA to find the multiplier vector at the root node.

Table 4 also shows the results of applying the primal heuristics of §5.3 ('z_{STOP}'). The remaining gaps between the lower bounds and these upper bounds ('Gap') are still considerable. However, as we shall see in Table 6, these initial solutions are within a few percent of the optimal solutions.

Next, we investigate the different branching rules presented in §5.3. Each of the three instances was tested using one of the branching rules. For a maximum computation time of one hour, the results of these nine tests are reported in Table 5.

An important difference in the results is the fact that, using the maximum unavailable flow rule ('Flow'), the instance NSNH could not be solved within the given time bound, while the two other rules solve within a few hundred seconds. Based on these results, we propose to use the estimated degradation rule ('Degradation') as the default branching rule.

Using the Degradation based branching rule, and strengthening the root node as described earlier, we have tested all three instances against three different scenarios for applying the bound improvement techniques. To investigate the computational tradeoff between better bounds and complex improvement techniques, we first tried using only the zero-vector at every node. This considers the zero-vector, and the multiplier vector of the parent, and chooses the one that gives the best bound. Through its simplicity, this technique is very fast at every node. Alternatively, we tried the MAA and RVA individually, but we have not tested a combination of these two.

The findings of applying different improvement techniques are shown in Table 6. A maximum computation time of two CPU hours is used. The first two instances could be solved to optimality, regardless of the configuration. However, the necessary amount of time and the number of enumeration nodes differs significantly. For these instances, the number of nodes is smallest when using the MAA at every node, even though the difference is only modest. The RVA does not outperform the trivial zero-vector method. For NSNH, the number of nodes needed to solve the instance is much higher. With respect to the required amount of time, the zero-vector method

Instance		Zero	MAA	RVA	STOP
NS3600	z_{LR}	5489	5489	5489	5489
	z_{STOP}	5489	5489	5489	5489
	Gap	0.00%	0.00%	0.00%	0.00%
	# Nodes	303	292	303	200
	Time	5.35	12.71	94.94	68.47
NSNH	z_{LR}	4139	4139	4139	4139
	z_{STOP}	4139	4139	4139	4139
	Gap	0.00%	0.00%	0.00%	0.00%
	# Nodes	18500	17068	25221	8700
	Time	319.17	1265.59	6284.49	4600
NSRandstad	z_{LR}	3996	3954	3835	-
	z_{STOP}	4138	4138	4138	-
	Gap	3.55%	4.65%	7.90%	-
	# Nodes	29733	4497	157	-
	Time	*	*	*	-

Table 6: Computational results of applying the MAA and RVA techniques at every node of tree. The results of the first column are obtained using only the zero-vector at every node. An asterisk (*) indicates that the time limit of 7200 seconds was exceeded. The NSRandstad could not be constructed using the original STOP formulation.

Instance	Current	New	Difference
NS3600	5607	5489	−118 (−2.1%)
NSNH	4331	4139	−192 (−4.4%)
NSRandstad	4281	4138	−143 (−3.3%)

Table 7: Comparing the current and new total travel times.

excels due to its simplicity. The time spent at a node is roughly a factor of four lower than for MAA. Even though the quality of the individual lower bounds is less, the overall computation times are lower.

The last column in Table 6 shows the findings for solving the complete STOP formulation of (4) as a mixed integer programming problem using CPLEX 8.1. As mentioned before, the NSRandstad instance is too large to construct in this way. The remaining two instances can be solved. However, the computation times are far worse than using, e.g., the zero-vector method, or MAA. This shows that the techniques proposed in this paper perform much better on practical instances than the off-the-shelf solver CPLEX.

In Table 7 we compare the total travel time using the currently operated station types ('Current'), with the best solution of Table 6 ('New'). The improvements of between 2.1% and 4.4% are indications that a reassignment of station types can indeed shorten the total travel time of the passengers. From the complete traffic assignment used by STOP it is easy to compute estimates of, for example, the number of direct travellers (see Table 9 in the Appendix). The effect of the new type assignment on the average number of necessary train changes is estimated to be small.

7 Summary and conclusions

This paper outlined an approach for solving the problem of determining the halting patterns for passenger trains through a network of stations and tracks. We first derived a mixed integer programming formulation of this problem, based on a multi-commodity flow formulation with

additional decision variables and restrictions (§3). For practical applications this model is too large. To make the problem more tractable, we introduced a Lagrangian relaxation (§4). This relaxation can be solved by considering a number of smaller easy-to-solve problems. The provided lower bounds are used in a branch-and-bound algorithm. This algorithm is described by introducing several classes of preprocessing rules (§5.1), lower bound improvement methods (§5.2) and branching rules (§5.3).

The described techniques were tested on three real-life instances of NSR. From these tests we conclude that the techniques perform well on practical instances, and significantly better than using the MIP formulation with CPLEX 8.1. The preprocessing techniques considerably reduce the size of the initial problem, and the improvement methods effectively strengthen the lower bounds. Two of the three test cases were solved to optimality within a time limit of two hours. The remaining case was provably solved to less than 4% of optimality. Compared to the current situation, the proposed solutions offer a reduction in the total travel time of between 2.1% and 4.4%.

Acknowledgments

This research is partly sponsored by the Human Potential Program of the European Union under contract no. HPRN-CT-1999-00104 (AMORE).

References

[1] L. Bahiense, N. Maculan, and C. Sagastizábal. The volume algorithm revisited: relaxation with bundle methods. *Mathematical Programming*, 94:41–69, 2002.

[2] M.R. Bussieck. *Optimal lines in public rail transport.* PhD thesis, Technical University Braunschweig, Braunschweig, Germany, 1998.

[3] A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P.L. Guida. Algorithms for railway crew management. *Mathematical Programming*, 79:125–141, 1997.

[4] G. Gallo and F. Di Miele. Dispatching busses in parking depots. *Transportation Science*, 79: 322–330, 2001.

[5] A.M. Geoffrion. Lagrangean Relaxation for Integer Programming. *Mathematical Programming Study*, 2:82–114, 1974.

[6] J.H.M. Goossens, C.P.M. van Hoesel, and L.G. Kroon. A branch-and-cut approach for solving line planning problems. METEOR Research Memorandum RM/01/016, University of Maastricht, Maastricht, The Netherlands, 2001. Forthcoming in Transportation Science.

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

[8] H.W. Kuhn and A.W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, 1951.

[9] J.T. Linderoth and M.W.P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.

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

[11] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*. John Wiley & Sons, Inc., New York, 1988.

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

[13] L.W.P. Peeters. *Cyclic railway timetable optimization*. PhD thesis, Erasmus Research Institute of Management (ERIM), Erasmus University Rotterdam, Rotterdam, The Netherlands, 2003.

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

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

[16] P.J. Zwaneveld. *Railway planning, routing of trains and allocation of passenger lines*. PhD thesis, Erasmus Universiteit Rotterdam, Rotterdam, The Netherlands, 1997.

Appendix

7.1 Bounding algorithms

Algorithm 2 Revised Volume Algorithm

Input: A vector of Lagrangian multipliers $\lambda_0 \geq 0$, and a primal bound z^*.
A tolerance $m_1 \in (0,1)$, and a relaxation factor $\mu \in (0,2)$.
A maximum # of iterations T, and a # of consecutive null steps N.
Two threshold values δ_ω and δ_ϵ.

Solve $\mathrm{LR}(\lambda_0)$, and let (F, x) be the optimal solution.
Initialise $\nu_0 = (AF - Bx)$, $z_1 = F$, $\hat{\lambda}_1 = p_1 = \lambda_0$, $\omega_0 = \omega_1 = \nu_0$, and $\epsilon_1 = 0$.
Calculate the new stepsize

$$s_{t+1} = \mu \frac{z^* - z_{\mathrm{LR}}(\lambda_t)}{\|\omega_t\|}. \tag{19}$$

Set $k = t = 1$.
while $t \leq T$ and $z_{\mathrm{LR}}(\hat{\lambda}_k) \leq z^*$ **do**
  Make the move $\lambda_t = \hat{\lambda}_k + s_t \omega_t$.
  Compute the ascent measure $\delta_t = s_t \|\omega_t\|^2 + |\langle \omega_t, \hat{\lambda}_k - p_t \rangle| + \epsilon_t$.
  if $\|\omega_t\|^2 \leq \delta_\omega^2$ or $|\langle \omega_t, \hat{\lambda}_k - p_t \rangle| + \epsilon_t \leq \delta_\epsilon$ **then**
    stop
  end if
  Solve $\mathrm{LR}(\lambda_t)$ as described in §4, and let (F, x) be the optimal solution.
  Set $\nu_t = (AF - Bx)$.
  if $z_{\mathrm{LR}}(\lambda_t) \geq z_{\mathrm{LR}}(\hat{\lambda}_k) + m_1 \delta_t$ **then** {*serious step*}
    Reset the counter $n = 0$, and set $k \leftarrow k + 1$
    Set $\hat{\lambda}_k = \lambda_t$
    if $\mu < 1$ **then** $\mu \leftarrow 1.1 \cdot \mu$ **end if**
  else {*null step*}
    $n \leftarrow n + 1$
    if $n \geq N$ **then** {*too many null steps*} Set $n \leftarrow 0$ and $\mu \leftarrow 0.5\mu$ **end if**
  end if
  Calculate the new stepsize s_{t+1} as in (19) above.
  Set $E_t = \nu_t(\hat{\lambda}_k - \lambda_t)$, and $\hat{E}_t = \omega_t(\hat{\lambda}_k - p_t)$.
  Set $\alpha_t \in [0,1]$ to be the value closest, or equal to

$$\frac{\frac{\hat{E}_t - E_t}{s_{t+1}} - \langle \nu_t, \omega_t \rangle + \|\omega_t\|^2}{\|\nu_t\|^2 - 2\langle \nu_t, \omega_t \rangle + \|\omega_t\|^2} \tag{20}$$

  Compute

$$\begin{aligned} z_{t+1} &= \alpha_t F + (1 - \alpha_t) z_t \\ \omega_{t+1} &= \alpha_t \nu_t + (1 - \alpha_t) \omega_t \\ p_{t+1} &= \alpha_t \lambda_t + (1 - \alpha_t) p_t \\ \epsilon_{t+1} &= \alpha_t \sigma_t + (1 - \alpha_t) \epsilon_t \end{aligned}$$

  where $\sigma_t = (1 - \alpha_t)\langle \nu_t - \omega_t, p_t - \lambda_t \rangle$
  Set $t \leftarrow t + 1$.
end while

Output: the proposed multiplier vector $\hat{\lambda}_k$

Algorithm 3 Multiplier Adjustment Algorithm

Input: A vector of Lagrangian multipliers λ, and a tolerance m_1.
A distance matrix with $D_{vi} \in \mathbb{R}_+$ for $v \in V$ and $i \in V \cup V^L$.
A predecessor matrix with $P_{vi} \in A^L$ for $v \in V$ and $i \in V \cup V^L$.
The set P_v contains the arcs of the shortest path tree rooted at v.
A maximum # of iterations T, and a # of consecutive null steps N.
A factor $c > 0$ for widening the derived sensitivity bounds.

Set $\hat{\lambda} = \lambda$ {*the best vector is* $\hat{\lambda}$}
while $t \leq T$ and $z_{\mathrm{LR}}(\hat{\lambda}) \leq z^*$ **do**
  for all stations v **do**
    for all $a = (i, j) \in A^L : a \notin P_v$ **do** {*loop all non-tree arcs*}
      if $P_{vi} \notin A^L_d$ **then break end if**
      Set $d = D_{vj} - (D_{vi} + d(\lambda)_a)$ {*d is non-positive*}
      Set $a' = (i', j') = P_{vj}$
      if a' is not in the same constraints as a **then** {*consider decreasing the cost of a*}
        $\delta^-_a = \max\{\delta^-_a, d\}$
      end if
      while $i \neq j$ **do**
        if $D_{vi} > D_{vj}$ or $t = v$ **then** {*consider decreasing the cost of arcs to i*}
          $a' = (i', j') = P_{vi}$ {a' *is on the path to the tail of a, but not to head.*}
          if $(a' \in A^L_d)$ or (a at a different station than a') **then**
            Update $\delta^-_{a'} = \max\{\delta^-_{a'}, d\}$
          end if
          $i \leftarrow i'$
        else {*consider increasing the cost of arcs to j*}
          $a' = (i', j') = P_{vj}$ {a' *is on the path to the head of a, but not to tail.*}
          if a' is not in the same constraints as a **then**
            Update $\delta^+_{a'} = \min\{\delta^+_{a'}, -d\}$
          end if
          $j \leftarrow i'$
        end if
      end while
    end for
  end for
  Stretch the bounds: $\delta^- \leftarrow c \cdot \delta^-$, and $\delta^+ \leftarrow c \cdot \delta^+$.
  if $\|\delta^+ - \delta^-\| \leq m_1$ **then stop end if**
  for all stations v that are not fixed **do**
    Solve the improvement problem (18) for v, giving the vector Δ
    Set $\lambda \leftarrow \lambda + \Delta$.
  end for
  Solve $\mathrm{LR}(\lambda_t)$ as described in §4, and let (F, x) be the optimal solution.
  if $z_{\mathrm{LR}}(\lambda)/z_{\mathrm{LR}}(\hat{\lambda}) \geq 1 + m_1$ **then** {*serious step*}
    Reset the counter $n = 0$.
    Set $\hat{\lambda} = \lambda$.
  else {*null step*}
    $n \leftarrow n + 1$
    if $n \geq N$ **then** {*too many null steps*} **stop end if**
  end if
end while

Output: the proposed multiplier vector $\hat{\lambda}$

7.2 Additional tables

	NS3600	NSNH	NSRandstad
#Var. initially	31642	99536	2257130
#Var. after	15348	55790	1055054
#Con. initially	148	212	554
#Con. after	88	162	443
Size reduction	71%	57%	63%

Table 8: Statistics for the original STOP formulation, before and after preprocessing.

		NS3600		NSNH		NSRandstad	
Distribution	Direct	93.5%	(−0.1%)	91.6%	(−0.3%)	80.0%	(−1.0%)
	1 Change	6.3%	(0.0%)	7.8%	(0.7%)	18.3%	(0.8%)
	2 Changes	0.2%	(0.1%)	0.5%	(−0.4%)	1.7%	(0.2%)
Speed (kph)	Direct	72.0	(3.6%)	62.9	(5.6%)	66.0	(4.5%)
	1 Change	47.2	(−7.3%)	43.0	(−0.6%)	50.7	(4.5%)
	2 Changes	0.0	(0.0%)	39.1	(−4.3%)	46.6	(1.7%)
Solution	Upgraded	0	(0.0%)	2	(5.6%)	0	(0.0%)
	Downgraded	3	(10.7%)	3	(8.3%)	11	(9.0%)
	Tr. time	5489	(−2.1%)	4139	(−4.4%)	4138	(−3.3%)
	Tr. distance	6274	(0.1%)	4057	(−0.1%)	4100	(0.4%)

Table 9: Solution statistics for travellers in the proposed solution. The differences with using the current station types are shown between brackets. The distribution of the travellers and their average speeds are shown for the direct travellers, and for those travellers that have to change trains once, etc. The last rows of the table show the percentages of all stations that were upgraded or downgraded relative to the currently operated station types, and the new total travel time and total travelled distance.