Dynamic Graph Generation and Dynamic Rolling Horizon Techniques in Large Scale Train Timetabling*

Frank Fischer[1] **and Christoph Helmberg**[1]

1 Technical University of Chemnitz, Department of Mathematics, 09107 Chemnitz

Abstract

The aim of the train timetabling problem is to find a conflict free timetable for a set of passenger and freight trains along their routes in an infrastructure network. Several constraints like station capacities and train dependent running and headway times have to be satisfied.

In this work we deal with large scale instances of the aperiodic train timetabling problem for the German railway network. The problem is modelled in a classical way via time discretised networks, its Lagrange-dual is solved by a bundle method. In order to handle the enormous number of variables and constraints dynamic graph generation and dynamic rolling horizon techniques are employed.

1998 ACM Subject Classification G.1.6 [**Numerical Analysis**]: Optimization

Keywords and phrases combinatorial optimization, train-timetabling

Digital Object Identifier 10.4230/OASIcs.ATMOS.2010.45

1 Introduction

Railway planning problems have been in the focus of interest of applied mathematics for a long time. Many problems from this field have been tackled with methods from discrete optimization. In this work we deal with the well known *train timetabling problem* (TTP), which tries to find conflict free timetables for given set of trains in some railway network.

For the TTP there exist periodic and aperiodic variants. For the periodic case most well known models are based on the *Periodic Event Scheduling Problem* introduced in [19], which is well suited for the description of subway or fast-train networks, see [13] for a detailed survey on this topic.

The aperiodic TTP is usually modelled in one of two ways. The first approach is to use event-based models similar to PESP, see [16, 17]. Although quite successful on several instances, these models have the disadvantage that station capacities cannot easily be incorporated into the model. PESP models have also been adapted to non-periodic cases where periodic corridors are used by different trains [6, 5]. Nevertheless, this model requires that most trains follow some periodic schedule.

The second approach is based on Integer Programming formulations using time discretised networks for the train routes, see [8, 7, 1, 3]. The main advantage of these formulations is the ability to deal with headway restrictions, but also other constraints like station capacities and prescribed timetables can be handled, *e. g.*, [4]. The solution methods include

* This work was supported by the *Bundesministerium für Bildung und Forschung* under grant 03HEPAG4. Responsibility for the content rests with the authors.

heuristic and exact branch-and-bound based methods using LP relaxation and Lagrangian Relaxations [3, 9].

Building on [11] in this paper we describe the advances achieved by applying the new techniques of dynamic graph generation and load balancing in handling the very large scale TTP instances of the German railway company Deutsche Bahn (DB), stemming from a common project with "Verkehrsnetzentwicklung und Verkehrsmodelle (GSV)" of DB. The instances comprise about 10% of the whole German railway network with about 3000 passenger *and* freight trains in a time period of about six hours. Different train-type dependent running-times and headway-times as well as station capacities have to be considered.

The paper is structured as follows. In section 2.1 we introduce the TTP in a formal way and in section 2.2 we formulate the base model. Section 3 describes the solution methods applied and, finally, some numerical results are given in section 4.

2 The train timetabling problem

2.1 Problem description

The train timetabling problem can be described as follows. We are given an *infrastructure network* $G^I = (V^I, A^I)$ with V^I the set of nodes representing stations and track switches and A^I a set of directed arcs representing tracks. In a typical network there are two kinds of tracks, those that may be used in exactly one direction (*double line arcs*) and those that may be used in both directions (*single line arcs*). The set of double line arcs is denoted by $A^{I,2}$ and the single line arcs by $A^{I,1}$ respectively, and we have $A^I = A^{I,1} \dot\cup A^{I,2}$. Note, for $(u,v),(v,u) \in A^I$ we have $(u,v) \in A^{I,1} \Leftrightarrow (v,u) \in A^{I,1}$, both representing the same physical track. Each node $u \in V^I$ has an *absolute capacity* $c_u \in \mathbb{N} \cup \{\infty\}$ denoting the maximal number of trains to be at that node simultaneously and each arc $a = (u,v) \in A^I$ has a *directional capacity* $c_a \in \mathbb{N} \cup \{\infty\}$ denoting the maximal number of trains to be at node v approaching over a.

In the network a set of trains R has to be scheduled. For each train $r \in R$ its predefined route is given by the sequence of nodes $U(r) = (u_1^r, \dots, u_{n_r}^r)$ the train has to visit in order. Since the trains differ in size and speed, we assign each train a *train-type* $m(r) \in M = M_P \dot\cup M_F$, where M_P is the set of *passenger train types* and M_F the set of *freight train types*. For each arc we have *type and behaviour-dependent running-times* $t_R \colon A^I \times M \times B^2 \to \mathbb{R}_+, B = \{stop, run\}$, where $t_R((u,v), m, b_u, b_v)$ denotes the running time of a train of type m over arc (u,v) with stopping behaviours b_u, b_v on the incident nodes.

Important restrictions are the safety distances on tracks between successive trains. If two trains enter a track in the same direction or a single-line track in opposite directions there must be a minimal difference between the two entering times, the so called *headway-time*. Like running-times, headway-times depend on the types and stopping-behaviours of both trains. The mapping $t_H \colon A^I \times M \times B^2 \times M \times B^2 \to \mathbb{R}_+$ describes the headway times with $t_H((u,v), m_1, b_{1,u}, b_{1,v}, m_2, b_{2,u}, b_{2,v})$ the headway-time if a train with type m_1 and behaviours $b_{1,u}, b_{1,v}$ uses the track (u,v) followed by a train with type m_2 and behaviours $b_{2,u}, b_{2,v}$. Analogously, the mapping $t_{HS} \colon A^I \times M \times B^2 \times M \times B^2 \to \mathbb{R}_+$ describes the headway-time on a single line track if the second train follows in opposite direction.

A special requirement in our case is a *predefined timetable* for passenger trains. For each passenger train $r \in R, m(r) \in M_P$, and each of its stations $u = u_i^r$ we have a *stopping interval* $I_u^r = [t_u^{S,r}, t_u^{E,r}] \subseteq \mathbb{Z} \cup \{\pm\infty\}$ and a *minimal stopping time* $d_u^r \in \mathbb{Z}_+$. Train r has to arrive at station u before the end of its stopping interval $t_u^{E,r}$, must stop and wait at the station for at least d_u^r minutes and is not allowed to leave before $t_u^{S,r} + d_u^r$. For freight trains only the starting time of the train at its first station is given by $t^{S,r}_{u^r_1}$, for convenience we define $I^r_u = [0, \infty]$ and $d^r_u = 0$ for all $r \in R, i = 2, \ldots, n_r$ with $m(r) \in M_F$.

The aim is to find a feasible timetable for all trains, observe the predefined time windows for all passenger trains or violate them as little as possible, and let all (freight) trains reach their final station as early as possible.

2.2 Model

We model the TTP in a rather classical way via time discretised networks for each single train, see, *e. g.*, [11, 2, 3]. Let $T = \{1, \ldots, N\}$ be the discretised time steps, where N is large enough to guarantee the existence of a feasible solution, and $[t]$ denote the time-step to which some time t is rounded.

For each train $r \in R$ let $G^r = (V^r, A^r)$ be the graph representing the predefined train-route with $V^r \subseteq (U(r) \cup \{\sigma^r = u^r_0, \tau^r = u^r_{n_r+1}\}) \times B$ with the interpretation $(u, b) \in V^r \Leftrightarrow$ train r visits station u with stopping behaviour b (note, a train may be forced to stop or is not allowed to stop at some stations) where σ^r is an artificial start node and τ^r is an artificial end node at which the train must stop. The set of arcs is $A^r = \{((u^r_i, b), (u^r_{i+1}, b')) \colon (u^r_i, b), (u^r_{i+1}, b') \in V^r\} \cup \{((u^r_i, \mathit{stop}), (u^r_i, \mathit{stop})) \colon (u^r_i, \mathit{stop}) \in V^r\}$. Each arc $((u, b), (u', b')) \in A^r$ is assigned a rounded running time

$$t^r_R(((u,b),(u',b')))$$
$$= \begin{cases} 1 & \text{if } u = u', \\ 0 & \text{if } u \neq u', \{u, u'\} \cap \{\sigma^r, \tau^r\} \neq \emptyset, \\ [t_R((u,u'), m(r), b, b')] & \text{if } b' = \mathit{run}, |\{u, u', \sigma^r, \tau^r\}| = 4, \\ [t_R((u,u'), m(r), b, b') + d^r_{u'}] & \text{if } b' = \mathit{stop}, |\{u, u', \sigma^r, \tau^r\}| = 4. \end{cases}$$

Note that each arc $(((u, b), t), ((u', b'), t') \in A^r, u \neq u'$ between two successive stations incorporates the minimal stopping time $d^r_{u'}$ at its destination. Now the time-expanded network $G^r_T = (V^r_T, A^r_T)$ is defined as $V^r_T = V^r \times T$ and $A^r_T = \{((u, t), (u', t')) \colon (u, u') \in A^r, t' = t + t^r_R((u, u'))\}$. As usual we introduce binary variables for each arc

$$x^r_e \in \{0, 1\}, r \in R, e \in A^r_T, \tag{1}$$

and the stopping-intervals are enforced by the simple constraints

$$x^r_{((u,t),(u',t'))} = 0, u \neq u', t < [t^{S,r}_u + d^r_u]. \tag{2}$$

There are two classes of constraints to be considered. First the capacity constraints in the nodes are modelled via coupling inequalities. Let $t \in T$ be a time step and $u \in V^I$ be an infrastructure node. Then

$$\begin{aligned} A(u,t) = &\{e = (((u', b'), t'), ((u, \mathit{stop}), \bar{t})) \colon e \in A^r_T, r \in R, u \neq u', t - d^r_u \leq \bar{t} \leq t\} \\ &\cup \{e = (((u', b'), t'), ((u, \mathit{run}), t)) \colon e \in A^r_T, r \in R\} \\ &\cup \{e = (((u, \mathit{stop}), t - 1), ((u, \mathit{stop}), t)) \colon e \in A^r_T, r \in R\}, \end{aligned}$$

denotes the set of all arcs that represent some train arriving (or waiting) at station u at time step t. Analogously, for $a = (u', u) \in A^I$ we define

$$\begin{aligned} A((u',u),t) = &\{e = (((u', b'), t'), ((u, \mathit{stop}), \bar{t})) \colon e \in A^r_T, r \in R, u \neq u', t - d^r_u \leq \bar{t} \leq t\} \\ &\cup \{e = (((u', b'), t'), ((u, \mathit{run}), t)) \colon e \in A^r_T, r \in R\} \\ &\cup \{e = (((u, \mathit{stop}), t - 1), ((u, \mathit{stop}), t)) \colon e \in A^r_T, r \in R, (u, u') \in A^r\}, \end{aligned}$$

the set of all arcs representing some train arriving (or waiting) at station u at time t coming over arc a. The absolute and directional capacities are then enforced by constraints

$$\sum_{e \in A(p,t)} x_e^r \leq c_p, \qquad p \in V^I \cup A^I, t \in T. \tag{3}$$

The second class of constraints are the headway constraints. Because headway times depend on train-types and stopping-behaviour which leads to complex conflict-graphs on the arcs, we used the idea of Borndoerfer und Schlechte [2] of *configuration networks*, which model feasible track allocations of an infrastructure arc instead of excluding conflicting arcs by cutting-planes. Let $a = (u, u') \in A^I$ be an infrastructure arc. For simplicity, we assume a is a double-line track. The configuration network $G^a = (V^a, A^a)$ of a is defined as follows. The set of nodes is $V^a = \{(e,p) \colon e = ((u,b),(u',b')) \in A^r, u \neq u', r \in R, p \in \{1,2\}\} \cup \{\sigma^a, \tau^a\}$ and the set of arcs is $A^a = \bigcup_{i=1}^4 A^{a,i}$ with

$$\begin{aligned}
A^{a,1} =& \{((e,1),(e,2)) \colon (e,1),(e,2) \in V^a\}, && \ldots \text{configuration arcs,}\\
A^{a,2} =& \{((e,2),(e',1)) \colon (e,2),(e',1) \in V^a, e \neq e'\}, && \ldots \text{headway arcs,}\\
A^{a,3} =& \{((e,1),(e,1)) \colon (e,1) \in V^a\} \cup \{(\sigma^a,\sigma^a),(\tau^a,\tau^a)\}, && \ldots \text{holdover arcs,}\\
A^{a,4} =& \{(\sigma^a,(e,1)) \colon (e,1) \in V^a\}\\
& \cup \{((e,2),\tau^a) \colon (e,2) \in V^a\} && \ldots \text{artificial start/stop-arcs.}
\end{aligned}$$

As for train-graphs, we time-expand this graphs w.r.t. headway times,

$$t_H^a(g) = \begin{cases} 1 & \text{if } g \in A^{a,3}, \\ 0 & \text{if } g \in A^{a,1} \cup A^{a,4}, \\ [t_H(a, m(r_1), b_1, b_1', m(r_2), b_2, b_2')] & \text{if } \begin{cases} g = ((e_1,2),(e_2,1)), \\ e_i = ((u,b_i),(u',b_i')) \in A^{r_i}, \\ i = 1,2, \end{cases} \end{cases}$$

and $G_T^a(V_T^a = V^a \times T, A_T^a)$ is the time-expanded *configuration graph* with

$$A_T^a = \{((u,t),(u',t')) \colon (u,u') \in A^a, t' = t + t_H^a((u,u'))\}.$$

Note, if $a = (u,u')$ is a single line arc the network is defined analogously but w.r.t. to t_H and t_{HS} and we have $G^{(u,u')} \equiv G^{(u',u)}$. A feasible configuration of infrastructure arc $a \in A^I$ corresponds to a path from $(\sigma^a, 1)$ to (τ^a, N) and in the graphs G^r an arc $e \in A_T^r, r \in R$ may be used only if its corresponding configuration arc $((e,1),t),((e,2),t))$ is contained in that path. Again, we introduce binary variables

$$x_e^a \in \{0,1\}, a \in A^I, e \in A^a, \tag{4}$$

and coupling *configuration constraints*

$$x_e^r = x_{e'}^a, e' = (((e,1),t),((e,2),t)),((e,1),(e,2)) \in A^{a,1}, a \in A^I. \tag{5}$$

For both graph types, a feasible solution corresponds to a path from the start to the end nodes. It will be convenient to collect the characteristic vectors of all feasible solutions in one of these graphs in the sets $\mathcal{X}^p, p \in R \cup A^I$,

$$\mathcal{X}^p = \{x^p \text{ is a feasible solution in } G_T^p\},$$

where $x^p = (x_e^p)_{e \in A_T^p}, p \in A^I \cup R$.

The objective function is designed so that delays of passenger trains are minimized and freight trains tend to run as fast as possible. Furthermore one has to take care of the different lengths of the train-routes. Let $t_{\min}^r(u) \in T$ be the earliest possible time when train r may leave from station u. For each time-step of delay the train is penalized by putting increasing costs on the outgoing run-arcs. We define the cost-function $w\colon \bigcup_{r \in R} A_T^r \to \mathbb{R}_+$ as follows. Let $e = (((u,b),t),((u',b'),t')) \in \bigcup_{r \in R} A_T^r$ be an arc then

$$w_e^r = \alpha^{m(r)} \cdot l_e \cdot \begin{cases} \sum_{\hat{t}=t_{\min}^r(u)}^{t} \hat{t}, & e = ((u,t),(u',t')), u \neq u', \\ 0, & \text{otherwise.} \end{cases}$$

where $\alpha^{m(r)}$ is a train-type-dependent scaling factor and l_e is the relative running-time over this arc w.r.t. the minimal running time of the train over its complete route.

The ILP formulation reads

$$\begin{aligned} \underset{\text{subject to}}{\text{maximize}} \sum_{r \in R} \sum_{e \in A^r} -w_e^r x_e^r & \\ x^p \in \mathcal{X}^p, & \qquad\qquad p \in R \cup A^I, \\ (3), (5), & \qquad\qquad \text{[coupling constraints]}. \end{aligned}$$

Note that we formulate this problem as a maximization problem so the dual becomes a minimization problem. Furthermore, since we have artificial arcs $((\sigma^r,t),(\sigma^r,t+1))$ which are not contained in any constraint, there is always a feasible solution of the model (each train can just start "late enough"). Because this may be unintentional, we usually assign high costs to those arcs.

3 Solution Methods

In this section we describe the methods used to solve the TTP. As the instances we regard have a very large number of stations, tracks and trains, standard solvers are not sufficient to handle those problems.

3.1 Bundle Method

The solution method is based on the *Lagrangian dual* of the model above obtained by relaxing the coupling constraints (3) and (5). Let $\bar{C}_1\bar{x} \leq \bar{c}_1$ denote the capacity constraints (3) and $\bar{C}_2\bar{x} = \bar{c}_2$ denote the configuration constraints (5). The Lagrangian dual problem reads

$$\min_{\substack{y_1 \geq 0 \\ y_2 \text{ free}}} \varphi(y_1, y_2)$$

where

$$\varphi(y_1,y_2) := \sum_{i=1}^{2} \bar{c}_i^T y_i + \sum_{p \in R \cup A^I} \varphi^p(y_1,y_2),$$

with

$$\begin{aligned} \varphi^r(y_1,y_2) &:= \max_{x^r \in \mathcal{X}^r} \sum_{e \in A^r} -w_e x_e^r - \Big(\sum_{i=1}^{2} y_i^T \bar{C}_i^r\Big) x^r, & r \in R, \\ \varphi^a(y_1,y_2) &:= \max_{x^a \in \mathcal{X}^a} -(y_2^T \bar{C}_2^a) x^a, & a \in A^I. \end{aligned}$$

Obviously, the φ^p are convex functions as maxima over affine functions. For each y_1, y_2 the evaluation of $\varphi(y_1, y_2)$ requires the solution of $|R \cup A^I|$ simple shortest path problems. Let $x(y_1, y_2)$ be an optimal solution of all subproblems for given y_1, y_2. Then

$$g(y_1, y_2) = \begin{pmatrix} \bar{c}_1 - \bar{C}_1 x(y_1, y_2) \\ \bar{c}_2 - \bar{C}_2 x(y_1, y_2) \end{pmatrix}$$

is a subgradient of φ at (y_1, y_2).

The ConicBundle [12] library implements a bundle method to solve problems of type

$$\min_{\substack{y_1 \geq 0 \\ y_2 \text{ free}}} f(y)$$

where $f(y)$ is a convex function given by a first-order oracle, *i. e.*, for given y the oracle returns $f(y)$ and a subgradient $g(y)$ of f at y. The method generates a sequence $(x_k)_{k \in \mathbb{N}}$ of *primal aggregates* that are convex combinations of the solutions returned by the oracle. For an appropriate subsequence $L \subseteq \mathbb{N}$, $(x_k)_{k \in L}$ converges to an optimal solution of the LP relaxation of the primal problem. Note that in general the x_k violate the coupling constraints $\bar{C}_1 x \leq \bar{c}_1$ and $\bar{C}_2 x = \bar{c}_2$ but nonetheless can be used as a good approximation to the optimal relaxed solution.

3.2 Dynamic Graph Generation

Because of the large number of arcs and nodes and the possibly large number of time steps N, it is not possible to keep the complete problem in memory. Therefore the concept of dynamic graph generation has been developed in order to reduce the memory requirements *without losing any information of the model.* Dynamic techniques for solving shortest-path problems on large networks attained significant attention in the last years, usually focused on road networks for route planning problems, see, *e. g.*, [18, 15, 14, 10]. In contrast to those problems, the cost functions in our case may change arbitrarily (*i. e.*, the weights may increase and decrease) at every iteration.

The key observation is that although a single train-graph may be huge due to time-expansion over many time steps, most trains only use a small portion of their graphs. Indeed, because the objective encourages trains to use "early" arcs, most paths tend to be near the first time-steps covered by the graphs. Therefore it seems worthwhile to keep only the necessary subgraphs in memory so that all shortest-path problems still can be solved correctly.

In this section we describe the concept of dynamic graph generation and how it fits into the bundle framework. Let $G = (V, A)$ be an acyclic graph, we allow loops, and let $\preceq$ denote the induced partial order (for generality in this section G is not restructured to the special structure of section 2). We assume that there are a unique minimal element $\underline{u} \in V$ and a unique maximal element $\overline{u} \in V$ (*i. e.*, each node is contained in some path from $\underline{u}$ to $\overline{u}$) and $(\underline{u}, \underline{u}), (\overline{u}, \overline{u}) \in A$.

Let $T = \{1, 2, \dots\}$ be the set of time steps.

▶ Definition 1. Let $d \colon A \to \{X \subseteq \mathbb{N}_0 \colon |X| < \infty\}$ be a function with $d((u, u)) = \{1\}$ for all $(u, u) \in A$. Then the graph $G_T = (V_T, A_T)$ with

$$\begin{aligned} V_T &:= V \times T, \\ A_T &:= \{((u, t_u), (v, t_v)) \in V_T \times V_T \colon (u, v) \in A, t_v - t_u \in d((u, v))\} \end{aligned}$$

is called *time-expansion of G*.

Let $c_0\colon A_T \to \mathbb{R}_+$ be a cost-function on the arcs of G_T. We partition the set A_T in two parts $A_T = A_1 \dot\cup A_2$ where A_1 is *closed* in the sense that $(u,t) \in V(A_1) \Rightarrow \forall t' \le t\colon (u,t') \in V(A_1)$, $A_1 = A_T(V(A_1))$ is induced and $(\underline{u},1) \in V(A_1)$. Let $c\colon A_T \to \mathbb{R}$ be another cost-function with $c_{|A_2} \ge c_{0|A_2}$. Now we define a subnetwork of G_T that is sufficiently large to solve the shortest-path problem on G_T w.r.t. c or detecting that this may not be possible. c_0 is a cost function with a well-known structure on the arcs A_T. The subset A_1 contains those arcs $a \in A_T$ whose actual costs $c(a)$ may differ from their original costs $c_0(a)$ due to the Lagrange multipliers, in fact, no information of the structure of c on A_1 is known. Because of this it is clear that the whole set A_1 must be kept in memory in order to solve the shortest-path problem w.r.t. c on G_T. In contrast, since we have some information about the structure of c on A_2 (see (**C1**) below for the precise requirement), not all arcs of A_2 must be kept in memory. The aim is now to characterize an appropriate subset $A_2' \subseteq A_2$ that is large enough to solve the shortest path problem on G_T or provides a certificate wherever it needs to be updated for this purpose.

We denote by $\partial A_1 := \{u \in V(A_1)\colon \exists\,(u,v) \in A_2\} \cup (\{\overline{u}\} \times T)$ the set of "boundary" nodes of A_1.

▶ Definition 2. Let G_T be a network and c be a cost-function as above. Then $G_T' = (V_T', A_T')$ with $A_T' = A_1 \dot\cup A_2'$, $A_2' \subseteq A_2$, is a *valid subnetwork* of G_T w.r.t. c if

(**S1**) $\forall\, w, w' \in \partial A_1, \forall\, w$-$w'$-paths $P \subseteq A_2$ there is a w-w'-path $P' \subseteq A_2'$ with $c_0(P') \le c_0(P)$.

▶ Observation 3. Let G_T' be a valid subnetwork of G_T, $u \in V(A_1)$, $v \in V(A_1) \cup \partial A_1$ and P be a shortest u-v-path in G_T' w.r.t. c', where $c'\colon A_T' \to \mathbb{R}$, $c'_{|A_1} = c_{|A_1}, c'_{|A_2'} = c_{0|A_2'}$. If $A(P) \cap A_2' = \emptyset$ then P is a shortest u-v-path in G_T w.r.t. c.

Proof. Let P be a shortest u-v-path in G_T' w.r.t. c' with $P \subseteq A_1$ and assume there exists a u-v-path $\tilde{P}$ in G_T such that $c(\tilde{P}) < c(P)$. Then $\tilde{P} \not\subseteq A_1 \cup A_2'$ since otherwise $c'(\tilde{P}) \le c(\tilde{P}) < c(P) = c'(P)$.

Because $u \in V(A_1)$ and $v \in V(A_1) \cup \partial A_1$ there must be a first arc $(w_1, w_2) \in \tilde{P}$, $w_1 \in \partial A_1, w_2 \notin \partial A_1$ and a first reentering arc $(w_1', w_2') \in \tilde{P}$ with $w_1' \notin \partial A_1, w_2' \in \partial A_1$. By (**S1**) there is a w_1-w_2'-path $Q \subseteq A_2'$ with $c'(Q) = c_0(Q) \le c_0(w_1 \tilde{P} w_2') \le c'(w_1 \tilde{P} w_2')$, where $w_1 \tilde{P} w_2'$ denotes the subpath of $\tilde{P}$ connecting w_1 and w_2', and therefore $c'(u\tilde{P}w_1 Q \overline{w}_2 \tilde{P} v) \le c'(P)$. Continuing like this exchanging all subpaths of $\tilde{P}$ not part of A_T' we obtain a u-v-path $\hat{P} \subseteq A_T'$ with $c'(P) \le c'(\hat{P}) \le c'(\tilde{P}) \le c(\tilde{P}) < c(P) = c'(P)$, a contradiction. ◀

Observation 3 gives a sufficient condition on the size of the subnet to solve a shortest path problem in G_T. If the shortest-path in the subnet contains at least one arc in A_2', the subnet is not large enough and must be expanded by increasing A_1 and then a new set A_2' has to be computed.

In order to be efficient, the computation of the arc set A_2' must be possible without using the values of the cost functions on arcs currently not in memory. The construction below requires the computation of some data structures a priori and is independent of the concrete cost-function c.

For our construction we assume that there is a partition of the nodes V, the subset containing $u \in V$ being denoted by $[u]$, which has the following properties:

(**K1**) $u, v \in [w], u \ne v \Rightarrow (u,v) \notin A$,
(**K2**) $(u,v) \in A \Rightarrow [u] \times [v] \subseteq A$,
(**K3**) $[\underline{u}] = \{\underline{u}\}, [\overline{u}] = \{\overline{u}\}$.

The nodes in some subset $[u]$ may be interpreted as representing the same station for different modes in which the train uses this station, *e. g.*, the run-node and the stop-node at the same station of a train-graph may form such a subset. For any subset $W \subseteq V$ we denote by $[W] := \{[u] \colon u \in W\}$ the set of all subsets induced by W and, similarly, for $E \subseteq A$ we denote $[E] := \{([u], [v]) \colon (u, v) \in E\}$. If $P = u_1 \dots u_n$ is a path in G then $[P] = [u_{i_1}] \dots [u_{i_k}]$ denotes the induced path where $i_1 = 1, u_n \in [u_{i_k}]$, $[u_{i_{j-1}}] \neq [u_{i_j}], j = 2, \dots, k$ and $u_{l-1} \notin [u_l] \Rightarrow \exists\, j \in \{2, \dots, k\} \colon u_{l-1} \in [u_{i_{j-1}}], u_l \in [u_{i_j}]$ (*i. e.*, $[P]$ is the sequence of node subsets $[u_i]$ without multiple copies of the same subset).

Next we assume that the cost function c_0 satisfies the following condition:

(C1) Let $P = (u_1, t_1) \dots (u_n, t_n) \subseteq A_T$ and $P' = (u'_1, t'_1) \dots (u'_m, t'_m) \subseteq A_T$ be two paths with $[P] = [P']$ and so that for all $[u_i][u_{i+1}] = [u'_j][u'_{j+1}]$ with $[u_i] \neq [u_{i+1}]$ there holds $t_i \leq t'_j \wedge t_{i+1} \leq t'_{j+1}$. Then $c(P) \leq c(P')$.

This property means that the costs of two paths along the same route can be compared if one visits every moving arc earlier than the other.

In constructing the arc set A'_2 we want to ensure that any ∂A_1-path leaving A'_2 can be interrupted at some arc in A'_2 so that it can be extended to a compatible path in A'_2. For this, we define the following sets

▶ Definition 4. Let $u \in V$ be a node. Then

(i) for each $([u], [v]) \in [A]$, the *minimal traversal time* is

$$\underline{d}(([u], [v])) := \min\{d((u', v')) \colon u' \in [u], v' \in [v]\},$$

(ii) for all $u \in V, [v] \in [V], (u, v) \in A, u \neq v$, we choose

$$N(u, [v]) \in \operatorname{Argmin}\{\min d((u, v')) \colon v' \in [v]\},$$

the *canonical successor of* u *in* $[v]$,

(iii) for all $[v] \in [V]$,

$$N([v]) := \{v' \in [v] \colon \exists\, u \in V, N(u, [v]) = v'\},$$

(iv) starting with $V_t^{[\underline{u}]} := \{(\underline{u}, t)\}$ we recursively construct generic anchor sets $V_t^{[v]} \subset [v] \times T$ for all $t \in T$ large enough as follows: For all $([u], [v]) \in [A]$, denote $\underline{d} = \underline{d}(([u], [v]))$, and put

$$\begin{aligned}
\tilde{V}_t^{[v]} &:= [v] \times \{t\}, \\
\bar{t}_t^{[u][v]} &:= \max\{t' + \min d((u', [v])) \colon u' \in [u], (u', t') \in V_{t-\underline{d}}^{[u]}\}, \\
\overline{V}_t^{[u][v]} &:= N([v]) \times \left\{t, \dots, \bar{t}_t^{[u][v]}\right\}, \\
\underline{V}_t^{[u][v]} &:= \big\{(v', t_u + \min d((u', v'))) \in V_T \colon u' \in [u], v' = N(u', [v]), \exists\, t_u \leq t - \underline{d}, \\
&\qquad\qquad \exists\, ((u', t_u), (v'', t_v)) \in A_T, t_v > t\big\}, \\
V_t^{[u][v]} &:= \tilde{V}_t^{[v]} \cup \overline{V}_t^{[u][v]} \cup \underline{V}_t^{[u][v]}, \\
V_t^{[v]} &:= \bigcup \{V_t^{[u][v]} \colon (u, v) \in A, u \neq v\}.
\end{aligned}$$

Note that the sets $V_t^{[v]}, v \in V, t \in T$, can be computed a priori because the time-expanded graph G_T has the same structure for all $t \in T$. Now the aim is to select sets $V_t^{[v]} \subseteq V(A_2)$ for appropriate time steps $t \in T$ such that the subset $A'_2 \subseteq A_2$ with $V_t^{[v]} \subseteq V(A'_2)$ can be easily constructed. For this we define the following sequences of time steps.

▶ Definition 5. Let $t \in T$ and $[w] \in [V]$. Then $\mathcal{T}^{[w],t} = (\tau_{[u]}^{([w],t)})_{[u] \preceq [w]}$ is defined by

$$\begin{aligned} \tau_{[w]}^{([w],t)} &:= t, \\ \tau_{[u]}^{([w],t)} &:= \min\left\{\tau_{[v]}^{([w],t)} - \underline{d}(([u],[v]))\colon (u,v) \in A, u \prec v \preceq w\right\}. \end{aligned}$$

For each node $w \in V$ we define

$$\underline{t}_{[w]} := \min\left\{t\colon \forall\, v \preceq w, V^{[v]}_{\tau_{[v]}^{([w],t)}} \subseteq V(A_2) \setminus V(A_1)\right\}.$$

One can think of $\mathcal{T}^{([w],t)}$ as time steps such that the $V^{[u]}_{\tau_{[u]}^{([w],t)}}, u \preceq w$, contain the nodes of the fastest paths from some $v \preceq w$ to w ending in $V_t^{[w]}$ and $\underline{t}_{[w]}$ is the smallest possible time index such that all those paths are contained in A_2. Again, these sequences can be computed a priori because they do not depend on the concrete partition $A_1 \dot\cup A_2$ and are identically up to a shift of the last time-index t. The minimal possible time step $\underline{t}_{[w]}$ has to be computed for each concrete A_2 which can be done efficiently with the sequences known in advance. Note, the sequence $\mathcal{T}^{([\overline{u}],t)}$ would be sufficient to construct a set A_2' but this set can be quite large. In order to improve this, we combine several of the sequences above as follows.

▶ Definition 6.

$$\begin{aligned} \Delta([u],[v]) := \Bigg\{ & \delta \in \mathbb{N}\colon \delta \geq \underline{d}(([u],[v])), \forall\,(u,t_u) \in V_t^{[u]}, \exists\,((u,t_u),(v,t_v)) \in A_T, \\ & \left[\left((v,t_v) \in V_{t+\delta}^{[u][v]}\right) \vee ((v,v) \in A, t_v \leq t+\delta)\right], \forall\, t \in T \text{ large enough}\Bigg\}. \end{aligned}$$

Note, by construction we have $\underline{d}(([u],[v])) = \min \Delta([u],[v])$. Each $\delta \in \Delta([u],[v])$ is a shift such that any path ending in $V_t^{[u]}$ for some $t \in T$ can be extended to some node in $V_{t+\delta}^{[u]}$ and can also be computed in advance.

1. $T^{[\overline{u}]} := \{\underline{t}_{[\overline{u}]}\}$.
2. $$\begin{aligned} T^{[u]} := \Bigg\{ & \min\Big\{t_w - \delta\colon t_w - \delta \geq \underline{t}_{[u]}, \delta \in \Delta([u],[w]), \forall\,(u,t_u) \in V^{[u]}_{t_w-\delta}, \exists\,(v,t_v) \in V^{[w]}_{t_w}, \\ & \exists\,(u,t_u)(v,t_v')\dots(v,t_v) \subset A_2\Big\}\colon t_w \in T^{[w]}, (u,w) \in A, u \prec w\Bigg\}. \end{aligned}$$

In particular, $\delta = \underline{d}(([u],[v])) \in \Delta([u],[v])$ is feasible. Using these sets we define

$$\tilde{V} := \bigcup_{u \in V} \bigcup_{t \in T^{[u]}} V_t^{[u]} \qquad \text{and} \qquad \tilde{A} := \{((u,t_u),(v,t_v)) \in A_2\colon (u,t_u),(v,t_v) \in \tilde{V}\}.$$

Once $\Delta([u],[v])$ is available, it is not hard to compute the sets $T^{[u]}, u \in V$, and therefore $\tilde{V}$ and $\tilde{A}$. The set $\tilde{A}$ is already large enough so that for any sequence $[u_1]\dots[u_n], ([u_i],[u_{i+1}]) \in [A]), i = 1,\dots,n-1$, there is a path $P = v_1 \dots v_m$ such that $V(P) \cap [u_i] \neq \emptyset$. In a last step we need to enlarge $\tilde{A}$ by some further arcs.

▶ Definition 7.

$$\hat{V}^{[\overline{u}]} := V^{[\overline{u}]}_{\underline{t}_{[\overline{u}]}}$$

and, for u $\prec \overline{u}$,

$$\begin{aligned}
\hat{V}^{[u]} :=& \Big\{(u,t) \in V(A_2) \colon \exists\, (u',t') \in \bigcup_{u'' \in [u]} B_{u''} \cup \bigcup_{\bar{t} \in T^{[u]}} V_{\bar{t}}^{[u]}, t \le t' \Big\} \quad \text{with} \\
B_u :=& \{(u,t_u) \colon\ \exists\, ((u,t_u),(v,t_v)) \in A_2, (v,t_v) \in \hat{V}^{[v]}, ((v,v) \in A \vee (v,t_v) \in \partial A_1), \\
& \qquad \exists\, t \in T^{[u]}, \exists\, (u',t'_u) \in V_t^{[u]}, \\
& \qquad \nexists\, (u',t'_u)(v,t'_v)\dots(v,t'_v+k) \subset A_2, t'_v + k = t_v\} \\
\cup& \{(u,t_u) \colon\ \exists\, ((u,t_u),(v,t_v)) \in A_2, (v,t_v) \in \hat{V}^{[v]}, \\
& \qquad \exists\, ((v,t_v),(w,t_w)) \in A_2, v \neq w, \exists\, t \in T^{[u]}, \exists\, (u',t'_u) \in V_t^{[u]}, \\
& \qquad \nexists\, (u',t'_u)(v',t'_v)\dots(v',t'_v+k)(w,t_w) \subseteq A_2, v' \in [v], t'_v + k \le t_v\},
\end{aligned}$$

and finally collecting all nodes together

$$\begin{aligned}
V' &:= \bigcup_{u \in V} \hat{V}^{[u]}, \\
A'_2 &:= \{((u,t_u),(v,t_v)) \in A_2 \colon (u,t_u),(v,t_v) \in V'\}.
\end{aligned}$$

The sets $\hat{V}^{[u]}, u \in V$, and $B_u, u \in V$, can be computed efficiently, starting from $\overline{u}$ and then going back along the partial ordering. The sets B_u ensure that we can reroute all paths reentering into V'.

▶ **Theorem 8.** *The graph $G'_T = (V'_T, A'_T)$ with $A'_T = A_1 \dot{\cup} A'_2$ and $V'_T = V(A'_2)$ is a valid subnetwork of G_T w.r.t. c_0 satisfying (**C1**).*

The proof of the theorem is technically involved and deferred to the appendix.

The train-graphs $G^r, t \in R$, have the structure required for the construction above. The configuration graphs $G^a, a \in A^I$, however, have not, but since their cost function is zero, it is trivial to extend valid subnetworks.

3.3 Rounding Heuristic

In order to obtain integer solutions, we use rounding heuristics based on the approximated relaxed solution generated by the bundle method. The main goal of the heuristic is to preserve the main characteristics of the relaxed solution while generating integral train paths. Therefore an incremental rounding is applied to generate integer flows for some trains as long as these flows do not deviate too much from the fractional flow. If further rounding produces solutions too far off the relaxation, we resolve the relaxation with partially fixed train paths. In particular, we used the following general framework for our rounding heuristics.

First we identify some conflicts in the relaxed solution. Two trains $r_1, r_2 \in R$ are in conflict on some infrastructure arc $a \in A^I$ if their average flow-times are too close to each other, *i. e.*, $|\bar{t}(r_2,a) - \bar{t}(r_1,a)| \le C^H(r_1,r_2,a)$ where $C^H(r_1,r_2,a)$ is some constant, which usually depends on the headway-times between r_1, r_2 on a. Let $\mathcal{C} = \{C = (r_1,r_2,a,t) \colon r_1, r_2$ are in conflict on a at time $t\}$ be the set of conflicts. The heuristic chooses some conflict $C = (r_1,r_2,a,t) \in \mathcal{C}$ and tries some variants of runs for r_1, r_2 up to a (keeping the trains as close to the relaxed solution as possible) while no other train moves. The best variant w.r.t. some value-function is then fixed. When a variant is fixed, all arcs that are blocked by the fixed arcs due to some constraints are removed from the fractional solution. This may lead to the case in which some train does not have any flow left in the relaxed solution over some infrastructure arc. In this situation we mark this train as “killed” from this arc on. The algorithm is outlined as follows.

1. Solve the relaxation, determine all conflicts $\mathcal{C}$.
2. While $\mathcal{C} \neq \emptyset$
   - **a.** get $C = (r_1, r_2, a, t) \in \mathcal{C}$ with t minimal, set $\mathcal{C} := \mathcal{C} \setminus \{C\}$,
   - **b.** if r_1 or r_2 is killed before a, kill both trains and go to 2,
   - **c.** try several possible variants of running r_1, r_2 until C,
   - **d.** rate the variants and select the best one,
   - **e.** fix the variant, remove all blocked arcs from the relaxed solution, kill trains if necessary,
   - **f.** go to 2,
3. if not all trains are finished go to 1.

Usually the relaxation is not solved on the complete interval but only until some time horizon. The horizon is moved further in time as more and more time steps are fixed by the heuristic. Because the step sizes by which the horizon is moved are not fixed a priori but determined during the solution process, we call this approach *dynamic rolling horizon*.

3.4 Load Balancing Functions

A main flaw of approximated solutions obtained via Lagrange-relaxation is that they tend to use too much capacity on arcs and nodes. Especially the configuration constraints $x^r_e = x^a_{e'}$ are often violated by the approximate solution since a single constraint with small fractional flow on x^r_e, say $x^r_e < 0.1$, is regarded as "almost feasible". These contribute only a small value to the subgradient. Therefore the relaxed solution often splits the train-flows into many small fractional paths violating configuration constraints by a tiny amount. High precision solutions are required to counter this effect, these entail rather long computation times for bundle methods.

This motivates the introduction of load balancing functions on arcs, which can be seen as a soft variant of headway constraints. For each train arc $e = ((u, b), (u', b')) \in A^r, u \neq u'$, corresponding to some infrastructure arc $a = (u, u') \in A^I$ we assign a value $\beta_e > 0$ representing the amount of capacity in time steps the usage of e would block. *E. g.*, if all headway-times of e would be two time-steps, then $\beta_e = 3$ since the usage of e would not only block the arc a at the time step of e but also one time step before and one after.

The usage-level of an arc $a = (u, u') \in A^I$ in the interval $[t_0, t_0 + \Delta]$ may be represented by a weighted sum of the form

$$\sum_{r \in R} \sum_{e=((u,b),(u',b')) \in \bigcup A^r} \sum_{t=t_0}^{t+\Delta} \beta_e x^r_{(((u,b),t),((u',b'),t'))}.$$

The purpose of the load-balancing function is to distribute capacity consumption on an arc equally over time. Therefore we introduce a convex function $f^{a,t}_b \colon \mathbb{R}_+ \to \mathbb{R}_+$ with

$$0 \in \operatorname{Argmin}\{f^{a,t}_b(z) \colon z \in \mathbb{R}_+\}, \tag{6}$$

add $-\sum_{a \in A^I} \sum_{t \in T} f^{a,t}_b(z^{a,t})$ to the objective function and add coupling constraints

$$\sum_{r \in R} \sum_{e=((u,b),(u',b')) \in \bigcup A^r} \sum_{t=t_0}^{t+\Delta} \beta_e x^r_{(((u,b),t),((u',b'),t'))} \leq z^{a,t}, a = (u, u') \in A^I, u \neq u'. \tag{7}$$

Note, condition (6) allows to formulate the constraint (7) as an inequality. With this the global cost function satisfies $c_{|A^2_d} \geq c_{0|A^2_d}$ during the execution of the bundle method which is required by the dynamic graph generation. In our implementation we choose piecewise linear convex functions $f_b^{a,t}$ that balance free minutes in a certain time-interval against train-minutes.

4 Numerical Results

For our tests we considered real-world instances of the German railway network, one which consists of the main long-distance and freight route along the river Rhine (a.k.a. Rhein-Main-Schiene) and another one which comprises roughly Baden-Wuerttemberg, for a time-period of about 6 hours with a discretisation of one minute. Table 1 shows the sizes of those instances.

Instance	nodes	arcs	ld[1)] trains	sd[1)] trains	freight trains
1	445	744	25	30	82
2	1776	3852	116	2640	632

1) ld = long distance trains, sd = short distance trains.

Table 1 Instances

We tested the algorithm with and without load-balancing and different horizon step sizes on both instances, the different configurations are listed in Table 2. The results can be seen in Table 3, all computations have been done on an Intel Core i7 CPU with 2.67 GHz and 12GB RAM.

run	1	2	3	4	5	6	7	8	9	10
load-distribution	yes	yes	yes	yes	no	no	no	no	yes	no
horizon-size[1)]	30	30	60	30	30	30	60	30	60	60
look-ahead size[2)]	30	60	30	60	30	60	30	60	120	120

1) maximal number of minutes to be fixed in one iteration,
2) additional number of minutes after horizon-size, in which the relaxation and heuristic solutions will be computed but not fixed before the next iteration.

Table 2 Test parameters

Table 3 indicates that the generated solutions have few delays and exhibit large savings in time compared with the original timetable of the trains. The main short-coming of the solutions is the violation of a few capacity constraints. The main motivation for introducing load-balancing functions was to get a better distribution of the single train runs in the relaxed solution. Because the rounding heuristic is guided by the relaxation, a good distribution leaves more room for the rounding heuristic to find feasible integer routes. As Table 3 shows, the introduction of load-balancing functions reduced the number of conflicts as well as the number and size of delayed passenger trains whereas the saved time for freight trains decreases, as expected.

In order to demonstrate the benefits of dynamic graph generation, Table 4 compares the number of all arcs in the train-networks with and without dynamic graph generation (for the dynamic case, the numbers are taken at the end of the relaxation).

Note that this table does only count the arcs of the train-graphs not the arcs of the configuration networks. This is because configuration networks grow very fast: A configuration

instance	run	late ld[1]	avg. delay[2] ld	max. delay[3] ld	late sd	avg. delay sd	max. delay sd	avg. sav. freight[4]	#conf.[5]	solution time
1	1	0	0	0	3	334	552	3904	8	784s
1	2	0	0	78	4	723	1482	3992	13	1242s
1	3	2	882	1386	6	571	1212	3950	13	841s
1	4	2	342	486	4	768	864	4150	4	1076s
1	5	0	0	78	5	828	2262	4594	12	516s
1	6	1	378	378	6	513	1386	4524	19	867s
1	7	3	578	1038	4	595	798	4297	20	569s
1	8	2	312	366	5	420	852	4232	14	778s
2	9	4	358	798	309	532	2700	972	44	7.5h
2	10	6	526	618	334	503	5196	1045	85	3h
2	4	3	353	912	307	476	1644	942	70	4.3h
2	8	8	692	1254	336	492	2400	1056	75	1.5h

1) number of trains with ≥ 3 minutes delay w.r.t. predefined timetable,
2) average maximal delay in seconds of trains with ≥ 3 minutes delay w.r.t. predefined timetable,
3) maximal delay in seconds of late trains w.r.t. predefined timetable, original timetable,
4) average savings of freight trains in seconds compared with the original timetable,
5) number of unresolved capacity conflicts.

Table 3 Solution quality

inst.	maximal number of time-steps	static	dynamic	inst.	maximal number of time-steps	static	dynamic
1	3600s	876162	275265	2	3600s	3654905	579152
1	7200s	1777667	312326	2	7200s	9476644	830430
1	10800s	3008239	476651	2	10800s	17573262	1195572

Table 4 Arc count for static and dynamic graphs

network on an arc with five trains where each train has all 4 possible running behaviours contains for a period of 60 time steps about

$$60 \cdot (\underbrace{4 \cdot 5 \cdot 4 \cdot (5-1)}_{\text{headway arcs}} + \underbrace{2 \cdot 4 \cdot 5}_{\text{holdover and configuration arcs}}) \geq 20000$$

arcs. Because each infrastructure arc has a corresponding configuration network and there are several hundred arcs in the infrastructure network, this leads to a huge number of variables which cannot be handled without dynamic generation, especially since most arcs carry more than only five trains per hour.

A Proofs

▶ Observation 9. Assume $t \in T$ and $(v, t_v) \in V_t^{[u][v]}$. Then $t_v \leq \bar{t}_t^{[u][v]}$.

Proof. For $(v, t_v) \in \tilde{V}_t^{[v]}$ or $(v, t_v) \in \overline{V}_t^{[u][v]}$ the assertion is clear. If $(v, t_v) \in \underline{V}_t^{[u][v]}$ there must be a $(u, t_u) \in V_T$ with $t_u = t_v - \min d((u, v)) \leq t - \underline{d}$ and $v = N(u, [v])$. Because

$(u, t - \underline{d}) \in V_{t-\underline{d}}^{[u]}$ we have $\bar{t}_t^{[u][v]} \geq t - \underline{d} + \min d((u,v)) \geq t_u + \min d((u,v)) = t_v$. ◀

▶ Observation 10. Let $((u,t_u),(v,t_v)) \in A_T$ be an arc, $u \prec v$ and let $t \in T$ with $t_v \geq t > \underline{d}$ where $\underline{d} := \underline{d}(([u],[v]))$.

(i) If $t_u \leq t - \underline{d}$ then there is an arc $((u,t_u),(v',t'_v)) \in A_T$ with $t'_v \leq t_v$ and $(v',t'_v) \in V_t^{[u][v]}$.
(ii) If $\exists\, \tilde{v} \in [v]$, $(\tilde{v}, t_v) \in V_t^{[u][v]}$, then there is an arc $((u,t_u),(v',t'_v)) \in A_T$ with $t'_v \leq t_v$ and $(v',t'_v) \in V_t^{[u][v]} \subseteq V_t^{[v]}$.

Proof.

(i) If $t_v = t$ we know $(v,t_v) \in \tilde{V}_t^{[v]} \subseteq V_t^{[u][v]}$. So assume $t_v > t$. Then by definition we have $(N(u,[v]), t_u + \min d((u,[v]))) \in \underline{V}_t^{[u][v]} \subseteq V_t^{[u][v]}$.
(ii) If $t_u \leq t - \underline{d}$ we are in case (i), so assume $t_u > t - \underline{d}$. We set $v' := N(u,[v])$ and $t'_v := t_u + \min d((u,v))$. On the one hand we have $t'_v > t - \underline{d} + \min d((u,v)) \geq t$ and on the other hand we know $t'_v \leq t_v \leq \bar{t}_t^{[u][v]}$ by assumption. It follows $(v',t'_v) \in \overline{V}_t^{[u][v]} \subseteq V_t^{[u][v]}$.

◀

▶ Proposition 11. The arc set $\tilde{A}$ has the following properties.

(i) Let $[u_1]\ldots[u_n] \subseteq [A]$ be a path. Then there are time-steps $t_i \in T^{[u_i]}, i = 1,\ldots,n$, such that $t_i + \delta_i = t_{i+1}, i = 1,\ldots,n-1$, for some $\delta_i \in \Delta([u_i],[u_{i+1}])$.
(ii) Let $[u_1]\ldots[u_m] \subseteq [A]$ be a path with $t_i \in T^{[u_i]}, i = 1,\ldots,m$, and $\delta_i \in \Delta([u_i],[u_{i+1}])$, $t_i + \delta_i = t_{i+1}, i = 1,\ldots,m-1$. For any $(v_1,t_{v_1}) \in V_{t_1}^{[v_1]}$ there is a path

$$P := (v_1,t_{v_1})(v_2,t_{v_2}^1)\ldots(v_2,t_{v_2}^{n_2})(v_3,t_{v_3}^1)\ldots(v_m,t_{v_m}^{n_m})$$

with $[P] = [u_1]\ldots[u_m]$ and $(v_i, t_{v_i}^{n_i}) \in V_{t_i}^{[u_{i-1}][u_i]}, i = 2,\ldots,m$.
(iii) Let $P = (u_1,t_{u_1})\ldots(u_n,t_{u_n}) \subseteq A_2$ be a path and let $t_i \in T^{[u_i]}, i = 1,\ldots,m$, and $\delta_i \in \Delta([u_i],[u_{i+1}]), t_i + \delta_i = t_{i+1}, i = 1,\ldots,m-1$, be defined as in (i) for $[u_1]\ldots[u_n]$. If $t_{u_i} \leq t_i$ for some $i = 1,\ldots,n$ and $t_{u_{i+1}} > t_{i+1}$ then there is a $(v,t) \in V_{t_{i+1}}^{[u_i][u_{i+1}]}$ such that $t \leq t_{u_{i+1}}$ and $((u_i,t_{u_i}),(v,t)) \in A_T$.

Proof.

(i) For $n = 1$ we know by definition there exists a $t_1 \in T^{[u_1]}$.
For $n > 1$ we know by induction there are time steps $t_i \in T^{[u_i]}, i = 2,\ldots,n$, and $\delta_i \in \Delta([u_i],[u_{i+1}]), i = 2,\ldots,n-1$, with $t_i + \delta_i = t_{i+1}, i = 2,\ldots,n-1$. By definition of $T^{[u_1]}$ we have a $\delta_1 \in \Delta([u_1],[u_2])$ such that $t_1 := t_2 - \delta_1 \in T^{[v_1]}$.
(ii) For $m = 1$ the statement is clear. So let $m > 1$ and assume we have already constructed the path $(v_1,t_{v_1}^1)\ldots(v_{m-1},t_{v_{m-1}}^{n_{m-1}})$. By definition of $\Delta([u_{m-1}],[u_m])$ we know there are a $v_m \in [u_m]$ and a $t_{v_m}^1 \in T$ with $((v_{m-1},t_{v_{m-1}}^{n_{m-1}}),(v_m,t_{v_m}^1)) \in A_T$ and $(v_m,t_{v_m}^1) \in V_{t_m}^{[u_{m-1}][u_m]}$ or $((v_m,v_m) \in A \wedge t_{v_m}^1 \leq t_m)$. In the first case we set $n_m := 1$ and the path $(v_1,t_{v_1}^1)\ldots(v_{m-1},t_{v_{m-1}}^{n_{m-1}})(v_m,t_{v_m}^1)$ has the desired property. Otherwise we may insert wait-arcs $(v_m,t_{v_m}^1)(v_m,t_{v_m}^1+1)\ldots(v_m,t_m)$ and appending those arcs is sufficient because $(v_m,t_m) \in \tilde{V}_{t_m}^{[u_m]} \subseteq V_{t_m}^{[u_{m-1}][u_m]}$.
(iii) Because $\delta_i \geq \underline{d} := \underline{d}(([u_i][u_{i+1}]))$ we know $t_{u_{i+1}} > t_{i+1} = t_i + \delta_i \geq t_i + \underline{d}$ and therefore $t_{u_i} \leq t_i \leq t_{i+1} - \underline{d}$. By Observation 10, (i) there is a $(v,t) \in V_{t_{i+1}}^{[u_i][u_{i+1}]}$ such that $t \leq t_{u_{i+1}}$ and $((u_i,t_{u_i}),(v,t)) \in A_T$.

◀

▶ **Proposition 12.** Let $P := (u, t_u) \dots (v, t_v) \subseteq A_2$ be a path with $t_u \le \min T^{[u]}$.

(i) If $(v, t_v) \notin V'$, then there is a path $P' := (u, t_u) \dots (v', t'_v)$ with $(v', t'_v) \in V_t^{[v]}, t \in T^{[v]}$, $c(P') \le c(P)$ and $|A(P') \setminus A'_2| < |A(P) \setminus A'_2|$.

(ii) If $(v, t_v) \in \partial A_1$ and $|A(P) \setminus A'_2| > 0$ then there is a path $P' := (u, t_u) \dots (v, t_v)$ with $c(P') \le c(P)$ and $|A(P') \setminus A'_2| < |A(P) \setminus A'_2|$.

Proof.

(i) Assume $[P] = [u_1] \dots [u_n]$ and let $t_i \in T^{[u_i]}, i = 1, \dots, n$, be the time-steps as defined in Proposition 11. Because $(v, t_v) \notin V' \supseteq V_{t_n}^{[v]}$ we have $t_u \le t_1$ and $t_v > t_n$ there must be some arc $((x, t_x), (y, t_y)) \in P$ with $x \in [u_i]$ and $y \in [u_j]$ for some $i, j \in \{1, \dots, n\}$ so that $t_x \le t_i$ and $t_y > t_j$. By Proposition 11, (iii) there is some arc $((x, t_x), (y', t'_y)) \in A_T$ with $(y', t'_y) \in V_{t_j}^{[u_j]} = V_{t_j}^{[y]}$ and $t'_y \le t_y$. So we may choose a latest arc $((\hat{x}, t_{\hat{x}}), (\hat{y}, t_{\hat{y}})) \in P, \hat{x} \in [u_k]$, such that there is an arc $((\hat{x}, t_{\hat{x}}), (\hat{y}', t'_{\hat{y}})) \in A_T$, $(\hat{y}', t'_{\hat{y}}) \in V_{t_l}^{[\hat{y}]}$, $t_l \in T^{[y]}$, and $t'_{\hat{y}} \le t_{\hat{y}}$. By Proposition 11, (ii) we find arcs

$$P_1 := (\hat{y}', t'_{\hat{y}})(v_{l+1}, t^1_{v_{l+1}}) \dots (v_{l+1}, t^{n_{l+1}}_{v_{l+1}}) \dots (v_n, t^{n_n}_{v_n})$$

with $(v_i, t^{n_i}_{v_i}) \in V_{t_i}^{[u_{i-1}][u_i]}, i = l+1, \dots, n$, and by Observation 9 we get $t^{n_i}_{v_i} \le \bar{t}_{t_j}^{[u_{i-1}][u_i]}$. Furthermore we know by Observation 10, (ii) for each arc $((p, t_p), (q, t_q)) \in P, q \in [u_i], i = l+1, \dots, n$, that $t_q > \bar{t}_{t_i}^{[u_{i-1}][u_i]}$ since otherwise we had a contradiction to the choice of $(\hat{x}, t_{\hat{x}})$. By **(C1)** the path

$$P' := (u, t_u) \dots (\hat{x}, t_{\hat{x}})(\hat{y}', t'_{\hat{y}})(v_{l+1}, t^1_{v_{l+1}}) \dots (v_n, t^{n_n}_{v_n})$$

fulfills $c(P') \le c(P)$ and $|A(P') \setminus A'_2| < |A(P) \setminus A'_2|$.

(ii) Let $(w, t_w) \in V(P) \setminus V'$ be a node. By (i) there is a path $P_1 = (u, t_u) \dots (w', t'_w)$ with $(w', t'_w) \in V_{t_j}^{[w]}, t_j \in T^{[w]}$ and $c(P_1) \le c((u, t_u) \dots (w, t_w))$ and $|A(P_1) \setminus A'_2| < |A((u, t_u) \dots (w, t_w)) \setminus A'_2|$. Choose the first $((x, t_x), (y, t_y)) \in P$ with $(x, t_x) \notin V'$, $(y, t_y) \in V'$, $(w, t_w) \prec (y, t_y)$. Using Proposition 11, (ii) we may extend P_1 by a path $P_2 = (w', t'_w) \dots (x', t'_x), x' \in [x]$, which satisfies by choice of $(x, t_x) : c(P_2) \le c((w, t_w) \dots (x, t_x))$. Because $(x, t_x) \notin B_x \subseteq V'$ we are in one of two cases:

1. If $(y, y) \in A$ or $(y, t_y) \in \partial A_1$ then by definition of B_x there is a path $P_3 = (x', t'_x) \dots (y, t_y) \subset A'_2$ or

2. if $(y, y) \notin A$ and $(y, t_y) \notin \partial A_1$, then by definition of B_x there is a $((y, t_y), (z, t_z)) \in P$ and a path $P_3 = (x', t'_x) \dots (z, t_z) \subset A_2$ that is also in A'_2 except possibly the last arc if $(z, t_z) \notin V'$.

Then the path $P' = P_1 P_2 P_3 \dots (v, t_v)$ satisfies $c(P') \le c(P)$ and $|A(P') \setminus A'_2| < |A(P) \setminus A'_2|$.

◀

Proof. (of Theorem 8): The path P fulfills the conditions of Proposition 12 and by applying this proposition repeatedly we get a path $P' \subseteq A_2$ with $c_0(P') \le c_0(P)$. ◀

References

1 Ralf Borndörfer and Thomas Schlechte. Models for railway track allocation. In Christian Liebchen, Ravindra K. Ahuja, and Juan A. Mesa, editors, *ATMOS 2007*, Dagstuhl, Germany, 2007. IBFI, Schloss Dagstuhl, Germany.

2 Ralf Borndörfer and Thomas Schlechte. A suitable model for a bi-criteria optimization approach to railway track allocation. ZIB-Report 08-22, ZIB, 2008.

3 U. Brännlund, P. O. Lindberg, A. Nou, and J. E. Nilsson. Railway timetabling using lagrangian relaxation. *Transportation Science*, 32(4):358–369, 1998.

4 Valentina Cacchiani, Alberto Caprara, and Paolo Toth. A column generation approach to train timetabling on a corridor. *4OR: A Quarterly Journal of Operations Research*, 6(2):125–142, June 2008.

5 Gabrio Curzio Caimi, Martin Fuchsberger, Marco Laumanns, and Kaspar Schüpbach. 09. periodic railway timetabling with event flexibility. In Christian Liebchen, Ravindra K. Ahuja, and Juan A. Mesa, editors, *ATMOS 2007*, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

6 Gabrio Curzio Caimi, Martin Fuchsberger, Marco Laumanns, Kaspar Schüpbach, and Stefan Wörner. The periodic service intention as a conceptual framework for generating timetables with partial periodicity. In *ISROR Proceedings, 2009*, 2009.

7 A. Caprara, M. Fischetti, P. Guida, M. Monaci, G. Sacco, and P. Toth. Solution of real-world train timetabling problems. In *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3*, page 3030, Washington, DC, USA, 2001. IEEE Computer Society.

8 Alberto Caprara, Matteo Fischetti, and Paolo Toth. Modeling and solving the train timetabling problem. *Oper. Res.*, 50(5):851–861, 2002.

9 Alberto Caprara, Michele Monaci, Paolo Toth, and Pier Luigi Guida. A lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Appl. Math.*, 154(5):738–753, 2006.

10 Daniel Delling and Giacomo Nannicini. Core routing on dynamic time-dependent road-networks. Technical report, Ecole Polytechnique, 2008. `http://www.optimization-online.org/DB_HTML/2008/12/2164.html`.

11 Frank Fischer, Christoph Helmberg, Jürgen Janßen, and Boris Krostitz. Towards solving very large scale train timetabling problems by lagrangian relaxation. In Matteo Fischetti and Peter Widmayer, editors, *ATMOS 2008*, Dagstuhl, Germany, 2008. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

12 Christoph Helmberg. *ConicBundle 0.3.6*. Fakultät für Mathematik, Technische Universität Chemnitz, 2010. `http://www.tu-chemnitz.de/~helmberg/ConicBundle`.

13 Christian Liebchen. *Periodic Timetable Optimization in Public Transport*. PhD thesis, Technical University Berlin, 2006.

14 Giacomo Nannicini, Philippe Baptiste, Gilles Barbier, Daniel Krob, and Leo Liberti. Fast paths in large-scale dynamic road networks. *Comput. Optim. Appl.*, 45(1):143–158, 2010.

15 Peter Sanders and Dominik Schultes. Engineering highway hierarchies. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 804–816, London, UK, 2006. Springer-Verlag.

16 Michael Schachtebeck and Anita Schöbel. IP-based techniques for delay management with priority decisions. In Matteo Fischetti and Peter Widmayer, editors, *ATMOS*, volume 08002 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.

17 Anita Schöbel. Integer programming approaches for solving the delay management problem. In Frank Geraets, Leo G. Kroon, Anita Schöbel, Dorothea Wagner, and Christos D. Zaroliagis, editors, *ATMOS*, volume 4359 of *Lecture Notes in Computer Science*, pages 145–170. Springer, 2004.

18 Dominik Schultes and Peter Sanders. Dynamic highway-node routing. In *WEA'07: Proceedings of the 6th international conference on Experimental algorithms*, pages 66–79, Berlin, Heidelberg, 2007. Springer-Verlag.

19 P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM J. Discret. Math.*, 2(4):550–581, 1989.