Dynamic Graph Generation for Large Scale Operational Train Timetabling

F. Fischer C. Helmberg

Preprint 2011-10

Impressum:
Herausgeber:
Der Dekan der
Fakultät für Mathematik
an der Technischen Universität Chemnitz
Sitz:
Reichenhainer Straße 39
09126 Chemnitz
Postanschrift:
09107 Chemnitz
Telefon: (0371) 531-22000
Telefax: (0371) 531-22009
E-Mail: dekanat@mathematik.tu-chemnitz.de
Internet:
http://www.tu-chemnitz.de/mathematik/
ISSN 1614-8835 (Print)

Dynamic Graph Generation for Large Scale Operational Train Timetabling[*]

Frank Fischer[†] Christoph Helmberg[†]

July 6, 2011

The aim of operational train timetabling is to find a conflict free timetable for a set of passenger and freight trains with predefined stopping time windows along given routes in an infrastructure network so that station capacities and train dependent running and headway times are observed. Typical models for this problem are based on time-discretized networks for the train routes coupled by conflict constraints. They grow very fast for large scale instances and quickly lead to intractable models.

Motivated by the observation that relaxations mostly use a narrow corridor inside such networks, we develop a general dynamic graph generation framework in order to control this size even for infinite time horizons. It can be applied to time-discretized networks modelling the routing of objects through capacity restricted handling stations with the property that early paths in the network are preferred. Without sacrificing any information compared to the full model, it includes a few additional time steps on top of the latest arcs currently in use. This "frontier" of the graphs can be extended automatically as required by solution processes such as column-generation or Lagrangian relaxation. The corresponding algorithm is efficiently implementable and linear in the arcs of the non-time-expanded network with a factor depending on the basic time offsets of these arcs. We give some bounds on the required additional size in important special cases.

With this dynamic graph generation technique we are able to solve relaxations of large scale real-world train timetabling problems of the German railway network of *Deutsche Bahn*. By enhancing the informativeness of the relaxation by convex load-balancing functions that distribute the train load on single tracks, it forms the basis of a dynamic rolling horizon approach to finding integer solutions of good quality.

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

[†]Technical University of Chemnitz, Department of Mathematics, 09107 Chemnitz, {frank.fischer,helmberg}@mathematik.tu-chemnitz.de

Keywords: time-expanded networks, column generation, Lagrangian relaxation, large scale

MSC 2010: 90C90; 90B10, 90B06, 90C35, 90C06

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 a variant of the well known *train timetabling problem* (TTP), which tries to find conflict free timetables for a given set of trains in some railway network.

For the TTP there exist periodic and aperiodic models. In the periodic case most models are based on the *Periodic Event Scheduling Problem* (PESP) introduced in [22], which is well suited for the description of subway or fast-train networks, see [16] for a detailed survey on this topic.

The aperiodic TTP is usually modelled in one of two ways described next. The first approach is to use event-based models similar to PESP, see [19, 20]. 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 [5, 6, 7]. Nevertheless, the models require that most trains follow some periodic schedule.

The second approach is based on integer programming formulations using time discretized networks for the train routes, see [1, 3, 8, 9]. The main advantage of these formulations is the ability to deal not only with headway restrictions, but also other constraints like station capacities and prescribed timetables can be handled, *e. g.*, [4]. The solution methods include heuristic and exact branch-and-bound based methods using LP relaxation and Lagrangian relaxations [3, 10].

Building on [13, 14] we develop a new dynamic graph generation technique applicable in models with time-discretized networks having a cost structure preferring earlier solutions (see (C) on page 16 for the precise requirement). This is not only typical for TTP models but also for many other applications, *e. g.*, in the optimization of production lines. In those models, the time-expanded networks are used to model state-transitions of certain objects over time. The nodes of the network represent the state of the object at some time (*e. g.*, stations of trains, or certain modes of a machine like on, off, stand-by) and the arcs model the transitions between those states. The order of the states an object has to pass through may be fixed (fixed train-routes or steps of a production procedure) or may provide some alternatives (*e. g.*, trains may visit alternative routes, the same job may be performed by different machines or tools). A feasible work-flow of an object then corresponds to a path in the time-expanded network. The networks of all objects are coupled via constraints ensuring conflict free configurations of the whole system. The task is then to find feasible paths w.r.t. to some objective function which satisfy the coupling constraints. In most of these applications early paths are preferred to late ones because an early completion of the jobs is desired (*e. g.*, trains should arrive at their destination as early as possible, certain products to be manufactured should be finished as soon as possible). The overall number of time-steps required to complete all jobs, *i. e.*, the number of time-steps over which the networks have to be expanded, has usually to be estimated in advance to ensure the existence of feasible solutions. If the instances to be solved are of large scale or contain a large number of time-steps because of a fine discretization, models using time-expanded networks tend to grow very fast.

The dynamic graph generation approach is designed for situations described above when the models are solved using column-generation or Lagrangian-relaxation. The idea is to identify parts of the networks that are important for the current algorithmic state of the optimization process and to store only those together with a few additional time steps in a truncated model, so that the need of extending the truncated model even further can be detected within this model without loss of information with respect to the full model. By exploiting the special structure of the networks and the fact that early paths are preferred this leads to truncated models of adaptive finite size without requiring any a priori knowledge on a bound of the maximum time required for feasibility. We will show that dynamic graph generation can be implemented very efficiently (Algorithm 31, Theorem 34) depending only linearly on the number of arcs in the non time-expanded network together with a factor accounting for the maximal number of possible durations of single state-transitions (Observation 33). For the important special case of paths we will give tight a priori bounds on the size of the additional time steps that have to be kept in memory (Observation 35, Observation 36).

The TTP model described in this paper gives an excellent example for the use of this dynamic graph generation framework in that its three different types of networks serve to illustrate three different variants. Indeed, passenger trains, which should reach each intermediate station as early as possible (Observation 37), freight trains, which should reach their final destination as early as possible (Theorem 11, Remark 12), and configuration networks modelling feasible usages of infrastructure arcs (Observation 5, Remark 6) show not only how the size of the generated subnetworks can be bounded for various types of cost structures but also that the dynamic generation algorithm is very easy to implement and may be simplified even further in relevant special cases.

A second technique to improve the quality of TTP models is the introduction of load-balancing functions. These are convex functions penalizing high loads on single tracks in order to improve the distribution of trains over this track. This leads to relaxed solutions that give better suggestions to the rounding heuristics for finding integer solutions. Furthermore, load-balancing functions can be combined easily with the dynamic graph generation approach.

We will demonstrate the new techniques on some large scale real-world instances of the German railway company *Deutsche Bahn*(DB), stemming from a common project with "Verkehrsnetzentwicklung und Verkehrsmodelle (GSV)" of DB. These 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 we introduce the TTP in a formal way and formulate the base model. The main part of this work is Section 3 where we develop the framework of dynamic graph generation and apply it to our TTP model. Section 4 describes our rounding heuristics and its combination with a dynamic rolling horizon approach and Section 5 introduces load-balancing functions. Finally, in Section 6 we discuss numerical results for the real-world instances of DB.

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 visit that node simultaneously. 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 visit node v and entering 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 $U(r) = (u_1^r, \ldots, u_{n_r}^r)$ 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* $I_u^r = [t_u^{S,r}, t_u^{E,r}]$ with $t_u^{S,r}, t_u^{E,r} \in \mathbb{Z} \cup \{\pm\infty\}$ and a *minimal stopping time* $d_u^r \in \mathbb{Z}_+$. If the train need not stop at a station, these are $I_U^r = [-\infty, \infty]$ and $d_u^r = 0$. If $d_u^r > 0$, 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 $t_{u_1^r}^{S,r}$. For convenience we define $I_u^r = [-\infty, \infty]$ and $d_u^r = 0$ for all $r \in R, i = 2, \dots, n_r$ 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 discretized networks for each single train, see, *e. g.*, [2, 3, 14]. Let $T = \{1, \dots, \infty\}$ be the discretized time steps and let $[t]$ denote the time-step to which some time t is rounded (we do not fix the precision of the discretization so this may vary). Note that we allow an infinite number of time steps which is only possible because of dynamic graph generation, see Section 3.

For each train $r \in R$ let $G^r = (V^r, A^r)$ be the graph representing the predefined train-route with $V^r := \{(u, \mathit{run})\colon u \in U(r), r \text{ may pass through } u\} \cup \{(u, \mathit{stop})\colon u \in U(r), r \text{ may stop at } u\} \cup \{\sigma^r, \tau^r\}$ 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 $A^r = \{((u_i^r, b), (u_{i+1}^r, b'))\colon (u_i^r, b), (u_{i+1}^r, b') \in V^r\} \cup \{((u_i^r, \mathit{stop}), (u_i^r, \mathit{stop}))\colon (u_i^r, \mathit{stop}) \in V^r\}$. Each arc $((u, b), (u', b')) \in A^r$ is assigned a rounded running time

$$t_R^r(((u,b),(u',b'))) = \begin{cases} 1 & \text{if } u = u', \\ 0 & \text{if } u \neq u', \{u,u'\} \cap \{\sigma^r, \tau^r\} \neq \emptyset, \\ [t_R((u,u'), m(r), b, b')] & \text{if } b' = \mathit{run}, |\{u, u', \sigma^r, \tau^r\}| = 4, \\ [t_R((u,u'), m(r), b, b') + d_{u'}^r] & \text{if } b' = \mathit{stop}, |\{u, u', \sigma^r, \tau^r\}| = 4. \end{cases}$$

Note that the running time from a station u to a station from u' incorporates the minimal stopping time $d_{u'}^r$ at u'. Fig. 1 shows an example network for a train. The time-expanded

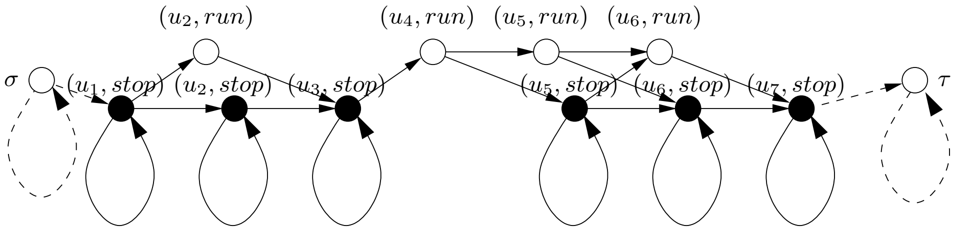


Figure 1: Example train graph G^r.

network $G_T^r = (V_T^r, A_T^r)$ is defined as $V_T^r = V^r \times T$ and $A_T^r = \{((u,t),(u',t'))^r\colon (u,u') \in A^r, t' = t + t_R^r((u,u'))\}$, see Fig. 2. As usual we introduce binary variables for each arc

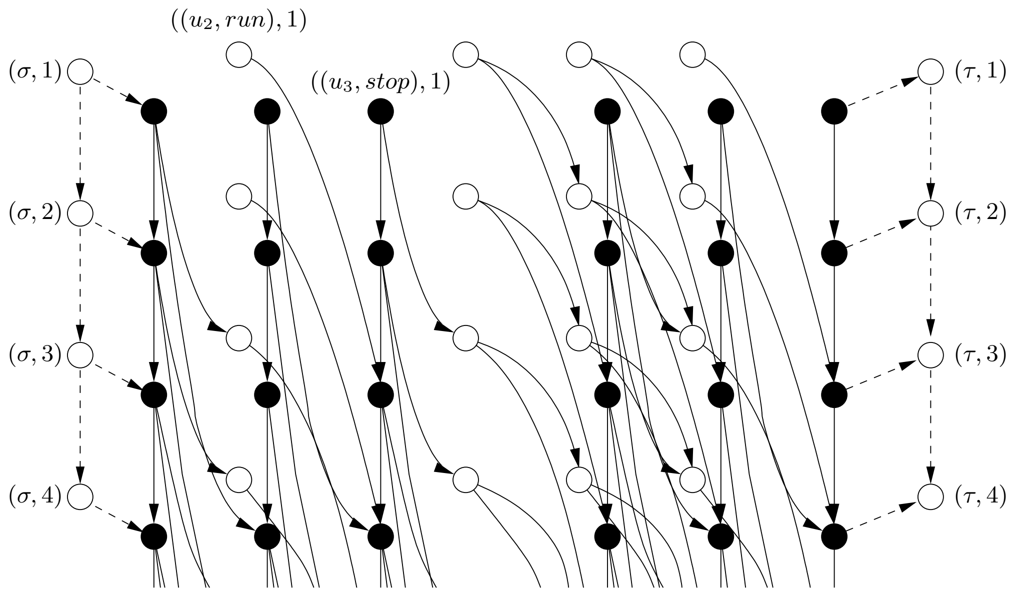


Figure 2: Example expanded train graph G_T^r.

$$x_e \in \{0,1\}, e \in A_T^r, r \in R, \tag{1}$$

and the stopping-intervals are enforced by the simple constraints

$$x_{((u,t),(u',t'))^r} = 0, u \neq u', t < [t_u^{S,r} + d_u^r], \tag{2}$$

which block all transfer arcs leaving u before $t_u^{S,r} + d_u^r$.

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 $(u', u) \in A^I$ be an infrastructure arc. Then

$$\begin{aligned} A((u',u),t) :=& \{e = (((u',b'),t'),((u,stop),\bar{t}))\colon e \in A_T^r, r \in R, \bar{t} - d_u^r \leq t \leq \bar{t}\} \\ & \cup \{e = (((u',b'),t'),((u,run),t))\colon e \in A_T^r, r \in R\} \\ & \cup \{e = (((u,stop),t-1),((u,stop),t))\colon e \in A_T^r, r \in R, (u',u) \in A^r\}, \end{aligned}$$

denotes the set of all arcs representing some train arriving (or waiting) at station u at time t coming over arc (u', u), and for some node $u \in V^I$

$$A(u,t) := \bigcup_{(u',u) \in A^T} A((u',u),t)$$

denotes the set of all arcs arriving from an arbitrary direction at u at time t. The absolute and directional capacities are then enforced by constraints

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

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 *configuration networks* of Borndoerfer and Schlechte [2]. These model admissible track allocations of an infrastructure arc $a = (u, u') \in A^I$ as follows. 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 $V^a = \{(e,p)\colon e = ((u,b),(u',b'))^r \in A^r, r \in R, p \in \{1,2\}\} \cup \{\sigma^a, \tau^a\}$ and the set of arcs is $A^a = \bigcup_{i=1}^4 A^{a,i}$ with

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

Fig. 3 shows an example graph for two trains. As for train-graphs, we time-expand these

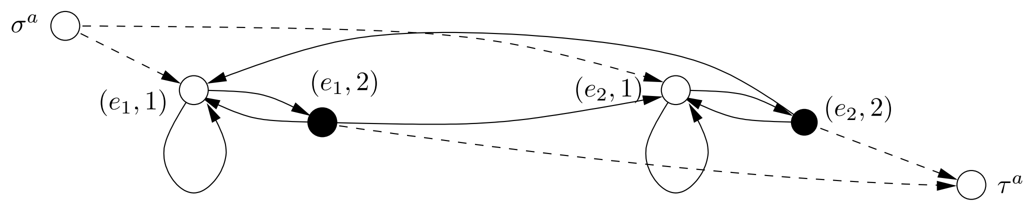


Figure 3: Example for a configuration network G^a for two trains.

graphs w.r.t. headway times via the function

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

where

$$\begin{aligned}
&t_H^f(a, m, b_1, b_1', b_2, b_2') :=\\
&\quad \min\{t_H(a, m, b_1, b_1', \bar m, \bar b, \bar b') + t_H(a, \bar m, \bar b, \bar b', m, b_2, b_2')\colon m \in M, \bar b, \bar b' \in B\}.
\end{aligned}$$

The function t_H^f denotes the minimal headway time between two run arcs of the same train. In principle, no train can use a certain infrastructure arc more than once, but when omitting the integrality constraints in the relaxation, a solution may contain more than one of the configuration arcs of some train. In order to strengthen the relaxation, the minimal headway time t_H^f ensures that in between two consecutive uses by the same train the time span is wide enough for at least one other train to run in between. By inserting this arc explicitly the relaxation is less likely to mess up the solution of some other trains, whose configuration arcs might otherwise be forced in between two consecutive uses of configuration arcs of the same train. This artificial decoupling also facilitates dynamic graph generation in Section 3.

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

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

Fig. 4 shows a time-expanded example graph corresponding to Fig. 3. If $a = (u,u') \in A^1$

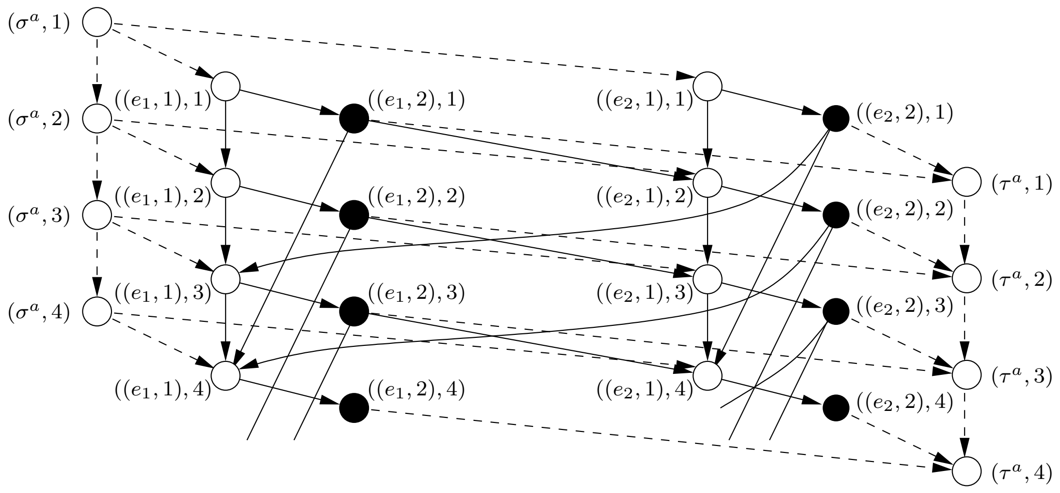


Figure 4: Example for a time-expanded configuration network G_T^a for two trains.

is a single line arc, the network is defined analogously but w.r.t. t_H and t_{HS} and we have only one network for both directions, *i. e.*, $G^{(u,u')} \equiv G^{(u',u)}$. An admissible 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 is contained in that path. Again, we introduce binary variables

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

and coupling *configuration constraints*, that link for $a = (u,u') \in A^I$ each running arc $e(a,r,b,b',t) := (((u,b),t),((u',b'),t'))^r \in A_T^r$, $r \in R$, to the corresponding configuration arc $\hat{e}(a,r,b,b',t) := (((e,1),t),((e,2),t)) \in A_T^a$ with $e = ((u,b),(u',b'))^r$,

$$x_{e(a,r,b,b',t)} = x_{\hat{e}(a,r,b,b',t)}, \quad a \in A^I, r \in R, e(a,r,b,b',t) \in A_T^r. \tag{5}$$

For train networks as well as for configuration networks, a feasible solution corresponds to a path from the start $(\sigma^p, 1)$ to one of the end nodes $\{\tau^p\} \times T$, $p \in R \cup A$. It will be convenient to collect for each graph G_T^p with $p \in R \cup A^I$ the characteristic vectors of all feasible solutions in the set

$$\mathcal{X}^p = \left\{x^p \text{ is the incidence vector of a path from } (\sigma^p, 1) \text{ to some } (\tau^p, t) \text{ in } G_T^p\right\},$$

where $x^p = (x_e^p)_{e \in A_T^p}$.

Because no real objective function is available for the given data, we chose an objective function according to the general aim that every train should get as far as possible as early as possible. This should not only avoid delays of passenger trains but should also help to increase time buffers for compensating actual delays. Freight trains will also arrive as early as possible but because of this objective they may start earlier than required and block some resources in waiting at stations; this trade off effect might need further investigation. Still, any objective function of similar type will meet the requirements for the dynamic graph generation technique of Section 3.

The objectives are modelled by assigning costs to the individual arcs of the train graphs. In order to balance these costs one has to take care of the different lengths of the train-routes. For each node $u \in V^r$ let $t^r_{\min}(u) \in T$ be the earliest possible time when train r may leave from station u when using a fastest possible path, *i. e.*,

$$t^r_{\min}(u) = \min\{t \in T \colon \exists \text{ path } P = (\sigma^r, 1) \dots ((u,b),t)(v,t') \dots (\tau^r, t'') \text{ in } G_T^r, \\ v \in V^r \setminus (\{u\} \times B), b \in B\}.$$

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

$$w_e^r = \alpha^{m(r)} \cdot l_e \cdot \begin{cases} \sum_{\hat{t}=t^r_{\min}(u)}^{t} \hat{t}, & e = ((u,t),(u',t')), u \neq u', \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

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. $t^r_{\min}(u_{n_r})$, the minimal running time of the train over its complete route.

The ILP formulation reads

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

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.

2.3 Solution Approach

As the instances we regard involve a large number of stations, tracks and trains over a rather long time span with relatively small time steps, standard solvers are not sufficient to handle those problems. Even keeping the time expanded networks in memory is out of scope. In order to be able to handle the size of the networks we propose the technique of dynamic graph generation in Section 3. It is suitable for column generation as well as Lagrangian relaxation approaches. We see advantages in the algorithmic possibilities offered by Lagrangian-relaxation in conjunction with bundle-methods for quickly finding approximate primal solutions and lower bounds for the model. In the short outline below we highlight those aspects of the approach that are of relevance for dynamic graph generation.

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) with Lagrange multiplier vector λ_1 and $\bar{C}_2\bar{x} = \bar{c}_2$ denote the configuration constraints (5) with multiplier vector λ_2. The Lagrangian dual problem reads

$$\min_{\substack{\lambda_1 \geq 0 \\ \lambda_2 \text{ free}}} \varphi(\lambda_1, \lambda_2) \tag{7}$$

where

$$\varphi(\lambda_1, \lambda_2) := \sum_{i=1}^{2} \bar{c}_i^T \lambda_i + \sum_{p \in R \cup A^I} \varphi^p(\lambda_1, \lambda_2),$$

with

$$\varphi^r(\lambda_1, \lambda_2) := \max_{x^r \in \mathfrak{X}^r} \sum_{e \in A_T^r} -w_e x_e^r - \Big(\sum_{i=1}^{2} \lambda_i^T \bar{C}_i^r\Big) x^r, \qquad r \in R,$$

$$\varphi^a(\lambda_1, \lambda_2) := \max_{x^a \in \mathfrak{X}^a} - \left(\lambda_2^T \bar{C}_2^a\right) x^a, \qquad a \in A^I.$$

The φ^p are convex functions because they are maxima over affine functions. For each λ_1, λ_2 the evaluation of $\varphi(\lambda_1, \lambda_2)$ requires the solution of $|R \cup A^I|$ simple shortest path problems. Let $x(\lambda_1, \lambda_2)$ denote the optimal solution computed by solving all subproblems for given λ_1, λ_2 (it will have finite support). Then

$$g(\lambda_1, \lambda_2) = \begin{pmatrix} \bar{c}_1 - \bar{C}_1 x(\lambda_1, \lambda_2) \\ \bar{c}_2 - \bar{C}_2 x(\lambda_1, \lambda_2) \end{pmatrix}$$

is a subgradient of φ at (λ_1, λ_2). We solve this model using a bundle approach, see, *e. g.*, [12]. Starting from $(\lambda_1^{(0)}, \lambda_2^{(0)}) = (0, 0)$ and $x^{(0)} = x(\lambda_1^{(0)}, \lambda_2^{(0)})$ it determines, in iteration $k \in \mathbb{N}$, multipliers $(\lambda_1^{(k)}, \lambda_2^{(k)}) \in \text{span}\{g(\lambda_1^{(j)}, \lambda_2^{(j)}) : 0 \leq j < k\}$ together with a primal aggregate $\bar{x}^{(k)} \in \text{conv}\{x^{(j)} : 0 \leq j < k\}$ and a primal optimizer $x^{(k)} = x(\lambda_1^{(k)}, \lambda_2^{(k)})$ such that an appropriate subsequence $(\bar{x}^{(k)})_{k \in L}$, $L \subseteq \mathbb{N}$, converges to an optimal (primal) solution of the LP-relaxation. In general those $\bar{x}^{(k)}$ violate the coupling constraints $\bar{C}_1 x \leq \bar{c}_1$ and $\bar{C}_2 = \bar{c}_2$, nonetheless they can be used as a good approximation to the optimal relaxed solutions.

The *actual cost-function* of each of the subproblems φ^p, $p \in R \cup A^I$, has several important properties for the dynamic graph generation framework below. First, with respect to the *basic cost-function* (for $\lambda_1 = 0, \lambda_2 = 0$) later paths are never better than earlier paths, so that trains may run as fast as possible (see Section 3.3 for a more formal statement of this property). Second, in each iteration the multipliers only change in coordinates whose corresponding constraints were violated in one of the previous iterations, and these changes induce localized changes of specific types in the *actual cost-function.* Indeed, all constraints concern certain time windows and can be violated if and only if a primal evaluation returned a primal solution with nonzero support in this time window. Furthermore, due to $C_1 \geq 0$ and $\lambda_1 \geq 0$, the violated inequality constraints only lead to increased costs in the shortest path problems. In contrast, equality constraints may make certain arcs more attractive by reducing their cost. This difference is of relevance for Section 3 and requires close inspection.

For equality constraints (5) a constraint can be violated if and only if the corresponding train-arc e or its associated configuration arc $\hat{e}$ has been used. The algorithm evaluating φ^a may safely be implemented so that it never uses a configuration arc whose associated train-arc has not yet been used. In consequence, the actual costs of the train-arc and the configuration arc may decrease in comparison to the basic cost function if and only if the arc has already been used in a previous optimal solution returned for φ^r.

Because an arbitrary subset of trains may lead to the violation of the inequality constraints (3) modelling station capacities, such a constraint may have a positive Lagrange-multiplier and may therefore change the actual cost-function for a specific train even if none of the returned solutions of this train contributed to the constraint. Due to the non-negativity of multipliers and constraint coefficients this can only increase the cost of the corresponding arcs in comparison to the basic cost function, so there is no need to consider such arcs if they are not even of interest for the basic costs.

These properties are essential for the dynamic graph generation framework and we summarize them for later reference.

Observation 1 *For $k \in \mathbb{N}_0$ and $p \in R \cup A^I$, let $(\lambda_1^{(k)}, \lambda_2^{(k)})$ and $x^{(k)}$ be the points generated by the algorithm above, put*

$$A^p_{(k)} = \{e \in A^p_T : (x^{(j)})^p_e = 1 \text{ for some } 0 \leq j \leq k\},$$

and denote by $\bar{w}^p_{(k)} \in \mathbb{R}^{\mathcal{X}^r}$ the linear cost function of subproblem $\varphi^p(\lambda_1^{(k)}, \lambda_2^{(k)})$. Then

$$(\bar{w}^p_{(k)})_e \geq (\bar{w}^p_{(0)})_e \text{ for all } e \in A^p_T \setminus A^p_{(k)}$$

and the set $\{e \in A^p_T : (\bar{w}^p_{(k)})_e \neq (\bar{w}^p_{(0)})_e\}$ is finite.

3 Dynamic Graph Generation

Time-expanded graphs in large scale real world instances usually contain a large number of arcs and nodes over a large number of time-steps so that it is impossible to keep the complete model in memory. We propose a dynamic graph generation technique 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.*, [11, 17, 18, 21]. In contrast to those problems, in our approach the cost functions may change arbitrarily (*i. e.*, the weights may increase and decrease) at every iteration as long as the initial cost structure satisfies a reasonable technical condition. While we concentrate on Lagrangian relaxation in our application, the same technique can be applied in column generation approaches.

The key observation is that although a single train-graph may be huge due to time-expansion over many time steps, most trains use only 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.

The framework is more general than required by the model in Section 2 in the following two aspects. First, we do not require the train-routes to be fixed. Instead, routing of the trains is allowed, *e. g.*, for modelling some alternative routes to bypass some construction site. Second, for each train the running times for two successive stations with given stopping behaviour does not need to be fixed, but several possible running times may be chosen. This allows to model different speeds at which the train may run from one station to the next (this can be combined with corresponding changes in headway times).

Throughout this chapter we will use the following notation for paths. Let $G = (V, A)$ be a directed graph, then $P = u_1 \dots u_m$ denotes the path or walk over the sequence of nodes $u_1, \dots, u_m \in V$ where $(u_i, u_{i+1}) \in A, i = 1, \dots, m-1$. For some subset $A' \subseteq A$ we will write $P \subseteq A'$ when each arc of P is contained in A'. For two nodes $x, y \in V$ which are contained in the path P, say $x = u_i$ and $y = u_j$ with $i \leq j$, we denote by xPy the subpath of P starting at node x and ending at node y, *i. e.*, $xPy = u_i u_{i+1} \dots u_j$. Similarly, we write $xP = u_i \dots u_m$ and $Py = u_1 \dots u_j$ for the subpaths of P starting at x or ending at y, respectively. Likewise, for two paths P and Q that visit the same node x somewhere along their path, the path PxQ first follows P to x and from there continues along Q. The set of nodes visited by P is denoted by $V(P) := \{u_1, \dots, u_m\}$ and the set of arcs by $A(P) := \{(u_i, u_{i+1}) \colon i \in \{1, \dots, k-1\}\}$.

We start by introducing the basic graphs. Let $G = (V, A)$ be an acyclic directed graph, except for loops. The induced partial order on the nodes is the *topological order.* We assume that there are a unique minimal element $\underline{u} \in V$ and a unique maximal element $\overline{u} \in V$ (*i. e.*, for each node there is path from $\underline{u}$ to $\overline{u}$ containing it) and $(\underline{u}, \underline{u}), (\overline{u}, \overline{u}) \in A$, see Fig. 5 for an example of a base graph with alternative routes.

The node groups indicated in this figure represent the same station for different modes in which the train uses this station. Except for loops these nodes are independent (not linked by arcs) and have the same predecessors and successors. In describing the route of the train it is irrelevant which of the nodes of such a group is visited. Therefore it will often be convenient to consider a partition of the nodes V into classes of *independent clones*, the subset containing $u \in V$ being denoted by $[u]$, which has the following

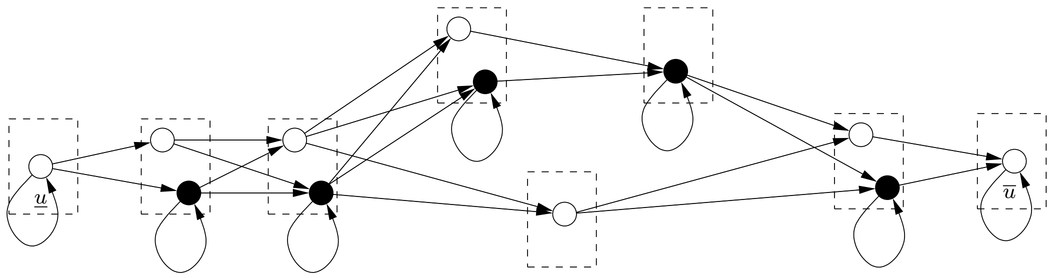


Figure 5: Base graph $G = (V, E)$ with alternative routes. The dashed boxes are independent clones $[u]$.

properties:

(K1) $u, v \in [w], u \neq v \Rightarrow (u, v) \notin A$ (note, loops are allowed inside a class),

(K2) $(u, v) \in A \Rightarrow [u] \times [v] \subseteq A$,

(K3) $[\underline{u}] = \{\underline{u}\}, [\overline{u}] = \{\overline{u}\}$.

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 by $[E] := \{([u], [v]) \colon (u, v) \in E, [u] \neq [v]\}$, where the condition $[u] \neq [v]$ deletes loops in going from E to $[E]$. Then $[G] = ([V], [A])$ denotes the *clone graph* of G which is an acyclic graph without loops. We call the elements of $[V]$ *clone nodes* and the elements of $[A]$ *clone arcs*. For each walk P in G there is a corresponding path in $[G]$, which is induced by the arcs of P. In more detail, if $P = u_1 \dots u_k$, then $[P] = [u_{i_1}] \dots [u_{i_l}]$ where $i_1 = 1$, $i_l = k$, is the sequence of clone nodes of P with duplicates removed. Likewise, if P is seen as a subset of arcs of A, then $[P]$ is the induced arc set of $[A]$.

Let $T = \mathbb{N}$ be the infinite set of time steps.

Definition 2 Let $d \colon A \to \{X \subseteq \mathbb{N}_0 \colon |X| < \infty\}$ be the *running-time function* with $d((u, u)) = \{1\}$ for all loops $(u, u) \in A$. 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. If $G' = (V', A')$ is a subgraph of G_T, then $\operatorname{cl} G'$ denotes the *closure* of G' with

$$\begin{aligned} \operatorname{cl} G' &:= (V^*, A^*), \\ V^* &:= \{(v, t) \in V_T \colon (v', t') \in V', [v'] = [v], t \leq t'\} \cup (V \times \{1\}), \\ A^* &:= (V^* \times V^*) \cap A_T. \end{aligned}$$

So $\operatorname{cl} G'$ may be thought of as the subgraph of G_T induced by all nodes (u, t) for which there is a node $(u', t') \in V'$ with $[u'] = [u]$ and $t \leq t'$. The inclusion of $(V \times \{1\})$ only ensures the presence of at least one node for each $v \in V$.

Let $c_0 \colon A_T \to \mathbb{R}_+$ be the *initial cost-function* on the arcs of G_T and suppose that with respect to this cost function "earlier solutions are not worse". The idea of our dynamic graph generation technique is as follows. During the solution process we solve several shortest-path problems repeatedly w.r.t. some *current cost function* $c^\lambda \colon A_T \to \mathbb{R}$ on G_T. For the next evaluation all previous paths are collected in the set

$$\mathcal{P}^\lambda = \{P \subset A_T : P = (\underline{u}, 1) \dots (\overline{u}, t) \text{ is a shortest path generated in an earlier iteration}\}.$$

The cost function may change in every iteration but has the property, that it may differ arbitrarily from the initial cost-function c_0 only on the finitely many arcs $\bigcup_{P \in \mathcal{P}^\lambda} P$ that have been contained in some shortest-path of an earlier evaluation and may only be larger than c_0 on the other arcs. Situations like this arise, *e. g.*, during the solution of the Lagrangian dual problem or as separation problem in a column generation approach where the cost function c differs from c_0 on those arcs that have been used in an earlier solution by the augmenting costs induced by the Lagrangian multipliers of violated coupling constraints, see Observation 1. We denote by $G_T^\lambda = (V_T^\lambda, A_T^\lambda) = \mathrm{cl}(\bigcup_{P \in \mathcal{P}^\lambda} V_T(P), \bigcup_{P \in \mathcal{P}^\lambda} P)$ the closure of the subgraph of G_T that contains all these paths. By construction it is finite and we assume throughout that the current cost-function satisfies the following two properties

(C1) $\forall\, a \in A_T \setminus A_T^\lambda \colon c^\lambda(a) \geq c_0(a)$,

(C2) For each of the cost functions c^λ and

$$c^C \colon A_T \to \mathbb{R}, c^C(a) := \begin{cases} c^\lambda(a), & \text{if } a \in A_T^\lambda, \\ c_0(a), & \text{otherwise,} \end{cases}$$

there is a shortest path among all paths in G_T that start in $(\underline{u}, 1)$ and end in $\{(\overline{u}, t) \colon t \in \mathbb{N}\}$.

The second condition is required because infinite graphs may fail to possess such shortest paths. A possible way to guarantee the existence of shortest paths is to assign higher costs to later paths in c_0 so that the increase in cost will outgrow all changes in c^λ eventually. The following immediate consequence of the definition of c^C is convenient in the following.

Observation 3 *For c^λ satisfying (C1) there holds*

(C1') $\forall\, a \in A_T \colon c^C(a) \leq c^\lambda(a)$.

Note that by using c^C instead of c^λ we may exploit the given structural properties of c_0 on all arcs $A_T \setminus A_T^\lambda$. In doing so the goal is to enlarge G_T^λ to a slightly larger *current subgraph* $G_T^C = (V_T^C, A_T^C) \supseteq G_T^\lambda$ of G_T so that solving a shortest-path problem w.r.t. c^C on G_T^C is equivalent to solving a shortest-path problem w.r.t. c^C on the infinite graph G_T. If the computed shortest path P w.r.t. c^C on G_T^C happens to satisfy $c^C(a) = c^\lambda(a)$ on all arcs $a \in P$, we have also solved the shortest-path problem w.r.t. c^λ on G_T. Otherwise P is added to $\mathcal{P}^\lambda$ and the shortest path problem will have to be recomputed for a new enlarged G_T^C with respect to a new c^C. Next, we state formally the requirements on the structural properties of G_T^C.

Denote by $\partial A_T^\lambda := \{u \in V_T^\lambda \colon \exists\,(u,v) \in A_T \setminus A_T^\lambda \vee \exists\,(v,u) \in A_T \setminus A_T^\lambda\} \cup (\{\overline{u}\} \times T)$, the set of *boundary nodes* of G_T^λ. A ∂A_T^λ*-path* is a path $w \dots w' \subset A_T \setminus A_T^\lambda$ with $w \in V_T^\lambda$ and $w' \in \partial A_T^\lambda$.

Definition 4 Given G_T let c^λ be a cost-function satisfying (C1) and (C2). Then $G_T^C = (V_T^C, A_T^C)$ with $A_T^C \subseteq A_T$ is a *valid subnetwork* of G_T w.r.t. c^λ if

(S) For all ∂A_T^λ-paths $P = w \dots w'$ there is a w-w'-path

$$P' \subseteq (A_T^C \cup \{((\overline{u},t),(\overline{u},t+1)) : t \in T\}) \setminus A_T^\lambda \text{ with } c_0(P') \leq c_0(P).$$

The following observation verifies that a valid subnetwork is indeed large enough to either certify that the global shortest path problem was solved correctly (if the resulting P is contained in G_T^λ) or to detect that an enlargement of G_T^C may be required (if $P \not\subseteq A_T^\lambda$).

Observation 5 *Assume G_T^C is a valid subnetwork of G_T w.r.t. c^λ and let $u \in V_T^\lambda$, $v \in V_T^\lambda \cup \partial A_T^\lambda$. If P is a shortest u-v-path in G_T^C w.r.t. c^C with $P \subseteq A_T^\lambda$ then P is a shortest u-v-path in G_T w.r.t. c^λ.*

Proof. Let P be a shortest u-v-path in G_T^C w.r.t. c^C with $P \subseteq A_T^\lambda$. Assume P is not a shortest-path in G_T w.r.t. c^λ, then there is a u-v-path $P' \subseteq A_T$ with $c^\lambda(P') < c^\lambda(P)$. We know $P' \setminus A_T^C \neq \emptyset$, because otherwise $c^C(P') \leq c^\lambda(P') < c^\lambda(P) = c^C(P)$ by (C1') and (C2) which contradicts the assumption on P. Because $u \in V_T^\lambda$ and $v \in V_T^\lambda \cup \partial A_T^\lambda$ there must be a first arc $(w_1, w_2) \in P'$, $w_1 \in V_T^\lambda$, $w_2 \notin V_T^\lambda$ and a first reentering arc $(w_1', w_2') \in P' \setminus A_T^\lambda$ with $w_2' \in \partial A_T^\lambda$. By (S) there is a w_1-w_2'-path $Q \subseteq A_T^C \cup \{((\overline{u},t),(\overline{u},t+1)) : t \in T\} \setminus A_T^\lambda$ with $c^C(Q) = c_0(Q) \leq c_0(w_1 P' w_2') \leq c^C(w_1 P' w_2')$, and therefore $c^C(P' w_1 Q w_2' P') \leq c^C(P')$. Continuing like this exchanging all subpaths of P' not part of $A_T^C \cup \{((\overline{u},t),(\overline{u},t+1)) : t \in T\}$ we obtain a u-v-path $\hat{P} \subseteq A_T^C \cup \{((\overline{u},t),(\overline{u},t+1)) : t \in T\}$ with $c^C(P) \leq c^C(\hat{P}) \leq c^C(P') \leq c^\lambda(P') < c^\lambda(P) = c^C(P)$, a contradiction. □

Remark 6 (Configuration networks) The configuration networks of Section 2.2 already fit into the simple framework above and can therefore be generated dynamically. While configuration networks tend to be quite large, because the basic graph G is very dense, the initial cost function c_0 for configuration networks is, fortunately, trivial. Indeed, $c_0 \equiv 0$, because configuration networks are only used for modelling headway-time constraints. Furthermore the only arcs which have changing costs are the configuration arcs $((u,t_u),(v,t_v)) \in A_T$ with $(u,v) \in A^{a,1}$, because they are contained in the configuration constraints (5). If the triangle inequalities

$$\forall\, a \in A^I, \forall\, b_1, b_2, b_3 \in M \times B \times B \colon [t_H(a,b_1,b_3)] \leq [t_H(a,b_1,b_2)] + [t_H(a,b_2,b_3)]$$

hold for the headway times (which can be assumed for technical minimal headway times), it is easy to see that each arc contained in an optimal path in a configuration network has a non-positive weight. Therefore it is sufficient for G_T^C to contain an arbitrary path of weighted length 0 between any two nodes in V_T^λ and ∂A_T^λ that are connected by a path outside A_T^λ. This can be arranged easily.

3.1 A simple valid subnetwork for complex cost structures

For more complex cost functions it must be possible to efficiently expand the current subnetwork G_T^C without using the values of the cost function on arcs currently not in memory. For this, our construction requires a weak assumption on the initial cost-function c_0 given in (C) below. The expansion depends on the computation of some data structures a priori (determined in a preprocessing step before the first shortest-path computation) and is independent of the concrete cost-function c^λ.

In the construction it is central to have firm control on the boundary structure of the graph along any possible route. In order to make the notion of a route precise, we will use, for each path $P = (u_1, t_1) \dots (u_k, t_k) \subseteq G_T$, the notation $[P] = [u_1] \dots [u_k]$ for the path on the clone graph induced by the path $u_1 \dots u_k$ in G.

Definition 7 A path $[u_1] \dots [u_k] \subset [A]$ is called a *route*.

If two paths $P, Q \subseteq G_T$ use the same sequence of clone nodes, *i. e.*, $[P] = [Q]$, they can be compared regarding the time-steps at which they visit the single clone nodes.

Definition 8 Let $P = (u_1, t_1) \dots (u_1, t'_1)(u_2, t_2) \dots (u_2, t'_2)(u_3, t_3) \dots (u_k, t'_k)$ and $Q = (v_1, s_1) \dots (v_1, s'_1)(v_2, s_2) \dots (v_2, s'_2)(v_3, s_3) \dots (v_k, s'_k)$ be two paths, $P, Q \subset G_T$. The path P is *not later* than Q, write $P \leq_T Q$, if

(i) $[P] = [Q]$ and

(ii) $\forall\, i \in \{1, \dots, k\}\colon t_i \leq s_i \wedge t'_i \leq s'_i$.

P is *earlier* than Q, write $P <_T Q$, if $P \leq_T Q$ and $t_i < s_i \vee t'_i < s'_i$ for some $i \in \{1, \dots, k\}$.

So $P \leq_T Q$ if and only if P enters each clone node not later than Q and also leaves each clone node not later than Q. Note that it is perfectly possible to have $P \leq_T Q$ while there are nodes $(u, t) \in V(P)$ and $(u', t') \in V(Q)$ with $[u] = [u']$ and $t > t'$ as long as condition (ii) holds. Obviously, the relation $\leq_T$ introduces a partial ordering on the set of paths.

The basic assumption on the initial cost function c_0 is as follows. Whenever a path P is not later than Q, *i. e.*, $P \leq_T Q$, then it also has a cost that does not exceed that of path Q.

(C) $\qquad P, Q \subseteq A_T, P \leq_T Q \Rightarrow c_0(P) \leq c_0(Q)$.

Note that this condition ensures that G_T always contains a minimal path w.r.t. c^λ.

Definition 9 A ∂A_T^λ-path $Q = (v_1, t_1) \dots (v_h, t_h) \subset A_T \setminus A_T^\lambda$ with $v_1 \neq \overline{u}$ is G_T^λ*-reducible* if there is a (v_1, t_1)-(v_h, t'_h)-path Q' in $A_T \setminus A_T^\lambda$ with $Q' <_T Q$ so that in addition $t'_h = t_h$ if $v_h \neq \overline{u}$. If no such path Q' exists, Q is called G_T^λ*-irreducible.*

Corollary 10 *Suppose G_T^C contains G_T^λ and all G_T^λ-irreducible paths, then G_T^C is a valid subnetwork whenever (C) holds.*

Proof. Let $P = (v_1, t_1) \dots (v_h, t_h)$ be a ∂A_T^λ-path. We have to show, there is a (v_1, t_1)-(v_h, t_h)-path $P' \subseteq (A_T^C \cup \{((\overline{u}, t), (\overline{u}, t+1)) : t \in T\}) \setminus A_T^\lambda$ with $c_0(P') \leq c_0(P)$. If $v_1 = \overline{u}$ or P is irreducible then $P' = P$ is a valid choice, so we may assume $v_1 \neq \overline{u}$ and that P is reducible in the following. Because P is finite it can be reduced until one arrives at a G_T^λ-irreducible path $P' = (v_1, t_1) \dots (v_h, t'_h) \leq_T P$ contained in G_T^C. We may assume $t'_h = t_h$ because if $v_h = \overline{u}$ then the path $P'(v_h, t'_h) \dots (v_h, t_h)$ has the corresponding properties. Condition (C) ensures $c_0(P) \geq c_0(P')$. This is (S). □

The purpose of the remainder of this section is the construction of an appropriate subnetwork G_T^+ to be added to G_T^λ, so that the network $G_T^C = \mathrm{cl}(G_T^\lambda \cup G_T^+)$ contains all irreducible paths and is therefore a valid subnetwork of G_T. When a ∂A_T^λ-path P leaves G_T^C it must allow to construct a path $P' <_T P$ by interrupting P at some arc and inserting arcs of G_T^+ over the same sequence of clone nodes but at earlier time-steps. One can think of G_T^+ as an additional *corridor* of nodes and arcs near, but not intersecting, the active subgraph G_T^λ. This corridor has to be large enough for three aspects.

1. It must have an *interception property*, *i. e.*, each path that crosses the corridor can be redirected into the corridor. Fig. 6 shows a sketch of the active subgraph and the corridor G_T^+.
2. The corridor needs to satisfy a *continuation property*, *i. e.*, for every possible entry point into the corridor it has to contain a valid path that leads on to $\overline{u}$.
3. Finally, it must have a *reentrant property*, *i. e.*, for any path P that reenters G_T^C, its redirection and continuation along the corridor must offer an alternative path $P' <_T P$ that meets P before or when P enters ∂G_T^λ.

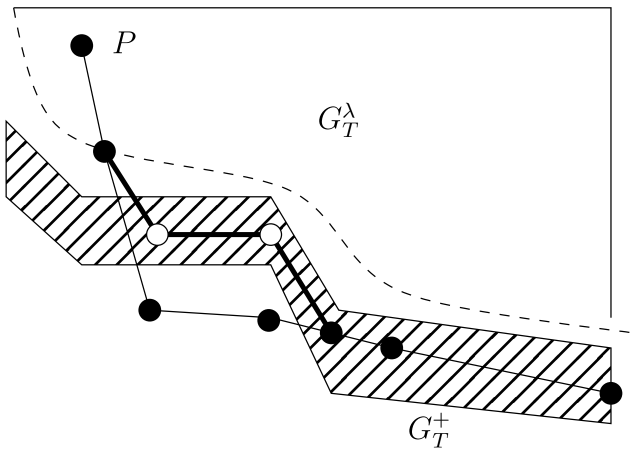


Figure 6: Active subgraph G_T^λ and intercepting region G_T^+. The thin path is redirected to the corridor G_T^+. In the corridor the thick path is the earlier replacement of the original path.

In principle it is not difficult to construct such a graph structure G_T^+. Indeed, as depicted in Fig. 7 it suffices to determine in a preprocessing step a graph structure giving the fastest variant of each route in V towards $\overline{u}$ with respect to *minimal arc times*

$$\underline{d}(u,v) := \min d((u,v)) \quad \text{for } (u,v) \in A.$$

For this, define a t-shifted fastest route graph $F_t = (V_F^t, A_F^t)$ for $t \in \mathbb{Z}$ recursively by $(\overline{u}, t) \in V_F^t$ and

$$\begin{aligned}((u,v) \in A \wedge (u \neq v) \wedge (v,\tau) \in V_F^t)& \\ \Rightarrow ((u, \tau - \underline{d}(u,v)) \in V_F^t \wedge ((u, \tau - \underline{d}(u,v)), (v,\tau)) \in A_F^t).& \quad (8)\end{aligned}$$

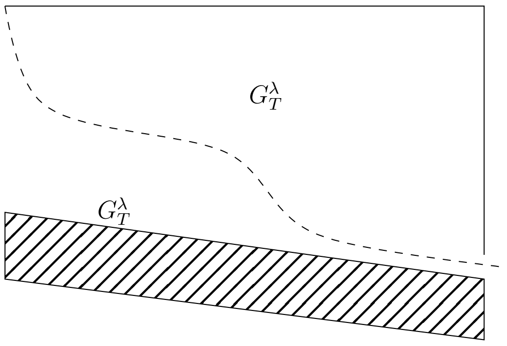


Figure 7: Basic corridor. The graph G_T^C will contain G_T^+ and everything above.

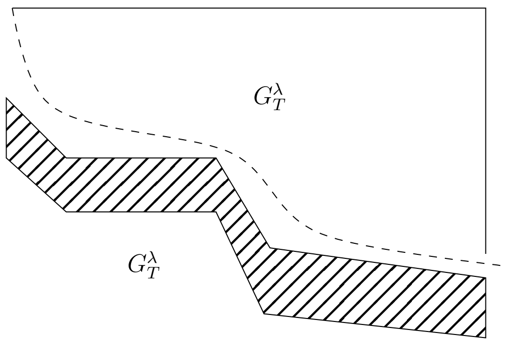


Figure 8: An improved corridor satisfies the same interception properties but is closer to G_T^λ.

Theorem 11 *Let $\underline{t} \in \mathbb{N}$ be the smallest t so that for all $(u,\tau) \in V_F^t$ of $F_t = (V_F^t, A_F^t)$ there holds $\tau > \max\{\tau' \colon (u,\tau') \in V_T^\lambda\}$. Put $\overline{d} := \max \bigcup_{(u,v)\in A} d((u,v))$ and $G_T^+ := (\bigcup_{t=\underline{t}}^{\underline{t}+\overline{d}-1} V_F^t, \bigcup_{t=\underline{t}}^{\underline{t}+\overline{d}-1} A_F^t)$. Then $G_T^C := \operatorname{cl} G_T^+$ is a valid subnetwork whenever (C) holds.*

Proof. Let $P = (w,\tau_w)\ldots(w',\tau_{w'}) \subset A_T \setminus A_T^\lambda$ be a ∂A_T^λ-path. If $P \subset G_T^C$ there is nothing to show, therefore assume $P \not\subseteq G_T^C$. Then there is a first node (v,τ_v) of P with $\tau_v \geq \underline{\tau}_v := \min\{\tau' \colon (v,\tau') \in V_F^{\underline{t}}\}$. Because $(w,\tau_w) \in \partial A_T^\lambda$ we have $\tau_w < \min\{\tau' : (w,\tau') \in V_F^{\underline{t}}\}$, so (v,τ_v) has a direct predecessor (u,τ_u) in P with $\tau_u < \underline{\tau}_u := \min\{\tau' \colon (u,\tau') \in V_F^{\underline{t}}\}$. If $u = v$ Definition 2 implies $\tau_u + 1 = \tau_v = \underline{\tau}_v = \underline{\tau}_u$ and therefore $(v,\tau_v) \in V_F^{\underline{t}}$. Otherwise, because $(u,v) \in A$ and $(v,\underline{\tau}_v) \in V_F^{\underline{t}}$, the definition (8) of $F_{\underline{t}}$ ensures $(u, \underline{\tau}_v - \underline{d}((u,v))) \in V_F^{\underline{t}}$ and we obtain $\tau_v - \overline{d} \leq \tau_u < \underline{\tau}_u \leq \underline{\tau}_v$, thus $\tau_v \in \{\underline{\tau}_v, \ldots, \underline{\tau}_v + \overline{d} - 1\}$. Hence, in both cases $(v,\tau_v) \in F_t$ for some $t \in \{\underline{t}, \ldots, \underline{t} + \overline{d} - 1\}$. Following P until (v,τ_v) and then following the route of P in F_T but using the fastest possible arcs (and no wait arcs) yields a path P' with $P' \subset G_T^C$ and $P' \leq_T P$. Because $P \not\subseteq G_T^C$ this can be strengthened to $P' <_T P$ and P has to end with $w' = \overline{u}$. Therefore P is reducible and the result follows by Corollary 10. $\square$

Looking at the proof we see that the definition of F_t includes more than a mere time expansion of the shortest path tree in $G = (V, A)$ in order to ensure the interception property and the continuation property along *any* route. The reentrant property is not needed here, because no path can outrun a fastest path along the same route. While the structure of the correct F_t might be quite involved, there is actually no need to store or construct the complete structure of F_t. Indeed, due to the closure operation it suffices, *e. g.*, to construct and store for each node $u \in V$ the two values $\min\{\tau \colon (u, \tau) \in F_0\}$ and $\max\{\tau \colon (u, \tau) \in F_0\}$ relative to $(\overline{u}, 0)$ within a preprocessing step, then G_T^C is quickly formed for any given ∂A_T^λ.

The main disadvantage of this approach is that the graph might become unfavorably large for two reasons. First, if the graph contains a sequence of nodes having a very slow fastest route (in our case this would be a path using only stop nodes instead of run nodes), this route is also contained in the current F_T even if there is a much faster clone equivalent path. This may cause the inclusion of time steps well beyond any interest for the optimization task at hand. Second, if the train has long stops at one or several stations as in Fig. 7, one would prefer the corridor to follow the boundary of G_T^λ rather closely, like in Fig. 8, in order to reduce the size of the subgraph G_T^C. In doing so we have to ensure that the three important properties still hold for the improved corridor. This will be considerably more complex than in the simple case above.

Remark 12 In some cases a different cost structure is required in practice. For example, for a freight train only the final arrival time at the last station might be important and runs with minimal waiting periods may be considered advantageous. Observe that in this situation condition (C) is *not* satisfied for freight trains, because a train starting early with many intermediate waiting periods may yield higher costs than a train starting later but passing through all stations without stopping. It is actually this situation, in which the structure of F_T is appropriate for dynamic graph generation. Indeed, there is no path that leaves F_T and later returns to F_T because F_T contains all fastest possible runs, so the construction of G_T^C of Theorem 11 is the best possible choice.

3.2 An improved corridor for reducing the size of valid subnetworks

In this section we describe an approach for an improved corridor that tries to follow the "frontier" of G_T^λ as closely as possible in order to keep the size of the final G_T^C small. The construction of the corridor G_T^+ consists of several steps. First we create for each clone node $[u] \in [V]$ a set of *interception nodes* $V^{[u]}$. Each set $V^{[u]}$ is a template of small subgraphs of $[u] \times \mathbb{Z}$ consisting of nodes (u, δ_u) where δ_u is to be interpreted as a relative time offset to a global time shift $t_{[u]} \in T$ applied to $V^{[u]}$ resulting in a shifted copy $V_{t_{[u]}}^{[u]}$ so that the resulting corridor will satisfy the interception and the continuation property while following the fastest realization of each clone path along the same route. Essentially, this is achieved by choosing along any clone arc $([u], [v]) \in [A]$ the fastest continuation

$$\min \bigcup_{u' \in [u], v' \in [v]} d((u', v'))$$

as a mutual time offset. While this helps to keep the corridor close to the active subgraph G_T^λ if the previously chosen paths do not stop and wait at any clone node, the basic "fastest" corridor will usually be quite far away from G_T^λ whenever the active paths include long waiting periods. So in a second step we improve the corridor by finding larger time-shifts for the interception nodes $V^{[u]}$ that include waiting possibilities so as to stay as close as possible to G_T^λ in order to reduce the size of the subgraph G_T^C. The concrete structure of the intercepting node sets depends only on the structure of G_T and not on the concrete cost function or the structure (and size) of the active subgraph G_T^λ. This ensures the important property, that they can be constructed in a preprocessing phase. We will see that by constructing the $V^{[u]}$ properly, it is not difficult to find time-steps that ensure the interception and the continuation property. The reentrant property, however, will require the addition of some further elements in the end.

In order to facilitate the handling of arc times like $\min \bigcup_{u' \in [u], v' \in [v]} d((u', v'))$ for some typical situations, we will use the following notation throughout,

$$\begin{aligned}
\underline{d}(u,v) &:= \min d((u,v)), & \overline{d}(u,v) &:= \max d((u,v)) \\
\underline{d}(u,[v]) &:= \min\{\underline{d}((u,v'))\colon v' \in [v]\}, & \overline{d}(u,[v]) &:= \max\{\overline{d}((u,v'))\colon v' \in [v]\}, \\
\underline{d}([u],v) &:= \min\{\underline{d}((u',v))\colon u' \in [u]\}, & \overline{d}([u],v) &:= \max\{\overline{d}((u',v))\colon u' \in [u]\},
\end{aligned}$$

$$\begin{aligned}
\underline{d}([u],[v]) &:= \min\{\underline{d}((u',v'))\colon u' \in [u], v' \in [v]\}, \\
\overline{d}([u],[v]) &:= \max\{\overline{d}((u',v'))\colon u' \in [u], v' \in [v]\},
\end{aligned}$$

Likewise it will be convenient to denote the minimal and maximal time-step of any finite node set $X \subset V \times \mathbb{Z}$ by

$$\underline{t}(X) := \min\{t\colon (x,t) \in X\} \quad \text{and} \quad \overline{t}(X) := \max\{t\colon (x,t) \in X\}\,.$$

In the definition of the $V^{[u]}$ below, the terms $\delta_{u'} - \delta_{v'} + \underline{d}([u],[v])$ specify relative time offsets w.r.t. a basic mutual time shift of $\underline{d}([u],[v])$ between $V^{[u]}$ and $V^{[v]}$.

Definition 13 A collection of sets $V^{[u]} \subset [u] \times \mathbb{Z}$, $[u] \in [V]$, is called *interception sets* if the following properties are satisfied,

(H1) *(base nodes property)*
$V^{[\underline{u}]} = [\underline{u}] \times \{0\}$ and $[v] \times \{0\} \subseteq V^{[v]}$ for all $[v] \in [V]$,

(H2) *(continuation property)* for all $([u],[v]) \in [A]$,
for $(u', \delta_{u'}) \in V^{[u]}$ there is a $(v', \delta_{v'}) \in V^{[v]}$ with $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d((u',v'))$,

(H3) *(interception property)* for all $([u],[v]) \in [A]$,
for $(u', \delta_{u'}) \in [u] \times \mathbb{Z}_-$ and $(v', \delta_{v'}) \in [v] \times \mathbb{N}$ so that $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d((u',v'))$,
there is a $(v'', \delta_{v''}) \in V^{[v]}$ with $\delta_{v''} - \delta_{u'} + \underline{d}([u],[v]) \in d((u',v''))$ and $\delta_{v''} \le \delta_{v'}$,

(H4) *(reinterception property)* for all $([u],[v]) \in [A]$,
for $(u', \delta_{u'}) \in [u] \times \mathbb{N}$ and $(v', \delta_{v'}) \in [v] \times \mathbb{N}$ so that $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d((u',v'))$ with $\delta_{v'} \le \overline{t}(V^{[v]})$,
there is a $(v'', \delta_{v''}) \in V^{[v]}$ with $\delta_{v''} - \delta_{u'} + \underline{d}([u],[v]) \in d((u',v''))$ and $\delta_{v''} \le \delta_{v'}$.

For $t \in T$ we define the *shifted interception sets*

$$V_t^{[u]} := \left\{ (u, t + \delta_u) \colon (u, \delta_u) \in V^{[u]} \right\}.$$

Conditions (H1)–(H4) ensure the continuation and interception properties of the interception nodes and therefore of the corridor. In contrast to the rather technical definition each property is motivated by rather straight forward algorithmic necessities to be explained next. For this, assume $([u], [v]) \in A$ and that G_T^+ contains the two sets $V_{t_{[u]}}^{[u]}$, $V_{t_{[v]}}^{[v]}$ with $t_{[v]} - t_{[u]} = \underline{d}([u], [v])$. Fig. 9 shows the corridor for three clone nodes.

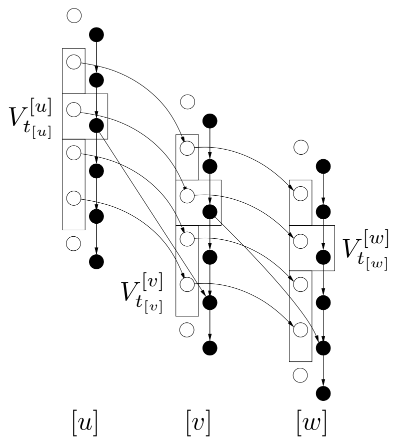


Figure 9: Basic corridor. The nodes within the rectangles form the set $V_{t_{[u]}}^{[u]}$ which contains *all* nodes of $[u]$ at time $t_{[u]}$ (small square) and usually only fastest nodes at the other time steps.

(H1) The *base node property* (Fig. 10) guarantees that if a path P contains a node at time $t_{[u]}$, it can be intercepted at exactly this node. This property is important to intercept paths that cross the corridor via wait-arcs.

(H2) The *continuation property* (Fig. 11) ensures that each path ending in $(u', t_{u'}) \in V_{t_{[u]}}^{[u]}$ can be continued via an arc $((u', t_{u'}), (v', t_{v'}))$ with $(v', t_{v'}) \in V_{t_{[v]}}^{[v]}$, thus being continued in the corridor.

(H3) The *interception property* (Fig. 12) takes care of paths P that use an arc $((u', t_{u'}), (v', t_{v'})) \in A_T(P)$ with $u' \in [u]$, $v' \in [v]$, $t_{u'} \leq t_{[u]}$ and $t_{v'} > t_{[v]}$, *i. e.*, it starts before the time shift $t_{[u]}$ of $V^{[u]}$ but ends after the time shift $t_{[v]}$ of $V^{[v]}$ thereby "jumping" over the initial time line of the corridor. If P then continues with the next node outside G_T^C, property (H3) allows to use the arc $((u', t_{u'}), (v'', t_{v''}))$ with $(v'', t_{v''}) \in V_{t_{[v]}}^{[v]}$ in order to redirect the path into the corridor without using any later nodes in $[v]$. From $(v'', t_{v''})$ on the replacing path of P can be constructed using (H2).

(H4) The *reinterception property* (Fig. 13) helps to intercept paths that run, in part, beyond the time line given by the $t_{[u_i]}$ (within or outside of G_T^C) and then touch G_T^C in a node v' later than $t_{[v']}$ that is added due to the closure operation, before leaving G_T^C in the next node. By redirecting the path from v' to $v'' \in V_{t_{[v']}}^{[v']}$ it visits $[v']$ no later and can be continued within G_T^C by (H2).

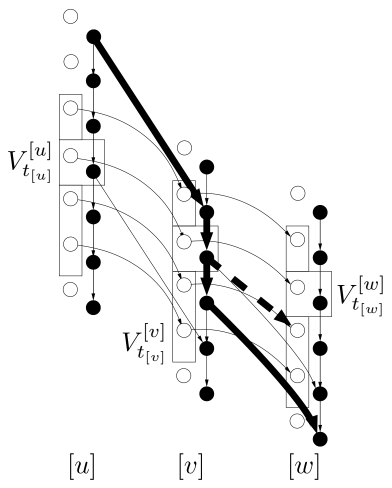


Figure 10: Property (H1). The thick path crosses the corridor using wait-arcs and can be intercepted at a node from $[v] \times t_{[v]} \subseteq V_{t_{[v]}}^{[v]}$.

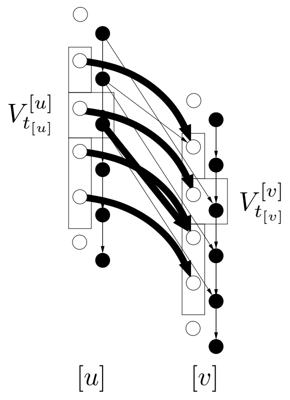


Figure 11: Property (H2). Each path leading on to a node in $V_{t_{[u]}}^{[u]}$ can be continued via one of the thick arcs to a node in $V_{t_{[v]}}^{[v]}$.

The intercepting node sets can be constructed inductively. In order to present one concrete algorithmic possibility we need the following notion.

Definition 14 Let $([u],[v]) \in [A]$, and $(u', \delta_{u'}) \in [u] \times \mathbb{Z}$. The *canonical successor* $N^{[v]}(u', \delta_{u'})$ of $(u', \delta_{u'})$ is a node $(v', \delta_{v'}) \in [v] \times \mathbb{Z}$ with $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) = \underline{d}(u',[v])$.

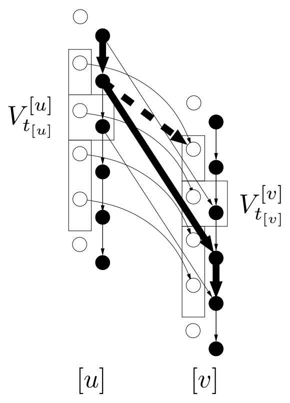


Figure 12: Property (H3). The thick path jumping over the corridor using an arc $((u, t_u), (v, t_v)), t_u < t_{[u]}, t_v > t_{[v]}$ can be redirected using the dashed arc into $V^{[v]}_{t_{[v]}}$.

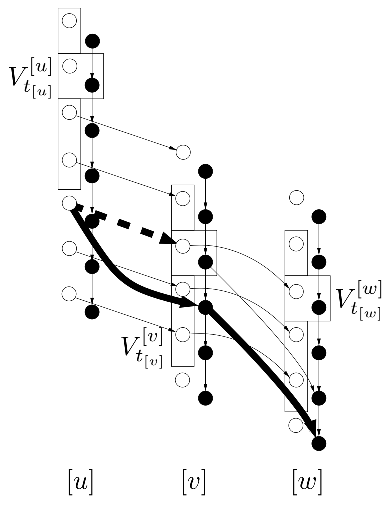


Figure 13: Property (H4). The thick path touches the set $V^{[v]}_{t_{[v]}}$ coming from a node (u, t_u) with $t_u > t_{[u]}$ and sometimes it has to be redirected using the dashed arc into the set $V^{[v]}_{t_{[v]}}$.

The canonical successor is therefore a fastest continuation from a certain node (u, δ_u) to $[v]$. A mapping $N^{[v]}\colon ([u] \times \mathbb{Z}) \to ([v] \times \mathbb{Z})$ satisfying the definition above is easily provided and is considered as given input data in the following.

Algorithm 15 *Construction of interception node sets (in topological order).*

```
1. $V^{[\underline{u}]} := [\underline{u}] \times \{0\}$,
2. For unprocessed $[v] \in [V]$ with $V^{[u]}$ computed for all $([u], [v]) \in [A]$,
    a) $W^{[v]}_1 := [v] \times \{0\}$,
```

b) $W_2^{[v]} := \{N^{[v]}(u', \delta_{u'})\colon ([u],[v]) \in [A], (u', \delta_{u'}) \in V^{[u]}\}$,

c) $W_3^{[v]} := \{N^{[v]}(u', \delta_{u'})\colon ([u],[v]) \in [A], (u', \delta_{u'}) \in [u] \times \mathbb{Z}_-, \exists\, (v', \delta_{v'}) \in [v] \times \mathbb{N},$
$\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d((u',v'))\}$,

d) $W_4^{[v]} := \{N^{[v]}(u', \delta_{u'})\colon ([u],[v]) \in [A], (u', \delta_{u'}) \in [u] \times \mathbb{N}, \exists\, (v', \delta_{v'}) \in [v] \times \mathbb{N},$
$\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d((u',v')), \delta_{v'} \leq \overline{t}(W_2^{[v]} \cup W_3^{[v]})\}$,

e) $V^{[v]} := W_1^{[v]} \cup W_2^{[v]} \cup W_3^{[v]} \cup W_4^{[v]}$.

The algorithm essentially constructs the interception sets in topological order starting from $\underline{u}$ by adding further nodes to the sets until all properties are fulfilled. A simple but important fact is that all sets $V^{[u]}$, $[u] \in [V]$, remain bounded.

Observation 16 *The sets $V^{[u]}$ constructed by Algorithm 15 are finite interception sets.*

Proof. While (H1) and (H2) can be verified directly, (H3) and (H4) hold because $N^{[v]}(u', \delta_{u'})$ provides the node $(v'', \delta_{v''})$ with smallest possible $\delta_{v''}$. Finiteness follows because $[G]$ and all sets $d(a)$, $a \in A$, are finite. □

Note that in general these sets may become quite large if the time sets $d(a)$, $a \in A$, have a complex structure. In practice, most instances have well structured running-times and the sets $V^{[u]}$ remain relatively small. The following observation gives an example for this under the reasonable assumption that there is a canonical "best" node $N^{[v]}$ for all $[v] \in [V]$ which is used by fastest paths in G_T. In the case of the train-graphs, the run-nodes have this property, because usually passing through a station is faster than stopping.

Observation 17 *Suppose for all $[v] \in [V]$ there is a node $N^{[v]} \in [v]$ so that*

$$\forall\, ([u],[v]) \in [A], \forall\, (u', \delta_{u'}) \in [u] \times \mathbb{Z}\colon N^{[v]}(u', \delta_{u'}) \in \{N^{[v]}\} \times \mathbb{Z}, \tag{9}$$

and

$$\forall\, ([v],[w]) \in [A]\colon \min d(N^{[v]}, N^{[w]}) = \underline{d}([v],[w]). \tag{10}$$

Then the sets $V^{[v]}$, $[v] \in [V]$, as constructed by Algorithm 15 fulfill

$$V^{[v]} \subseteq ([v] \times \{0\}) \cup (\{N^{[v]}\} \times \{-\overline{d}, \dots, \overline{d}\})$$

where

$$\overline{d} := \max\{\underline{d}(u,v)\colon (u,v) \in A\}.$$

Proof. The claim is certainly true for $V^{[\underline{u}]}$. So let $[v] \in [V]$ and assume as induction hypothesis the claim holds for all $V^{[u]}$ with $([u],[v]) \in [A]$. Let $(v', \delta_{v'}) \in V^{[v]}$. First we observe

$$\text{for all } (u', \delta_{u'}) \in [u] \times \mathbb{Z} \text{ with } (v', \delta_{v'}) = N^{[v]}(u', \delta_{u'}) \text{ there holds } \delta_{u'} \leq \delta_{v'}, \tag{11}$$

which is clear by definition of $N^{[v]}(u', \delta_{u'})$, because $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) = \underline{d}(u',[v]) \geq \underline{d}([u],[v])$. If $(v', \delta_{v'}) \in ([v] \times \{0\})$, the claim is clear. Otherwise $(v', \delta_{v'}) \in W_i^{[v]}$ for some $i \in \{2,3,4\}$ which implies $v' = N^{[v]}$ by (9). We consider three cases:

1. $(v', \delta_{v'}) \in W_2^{[v]}$, then there is a $(u', \delta_{u'}) \in V^{[u]}$ for some $([u],[v]) \in [A]$ with $(v', \delta_{v'}) = N^{[v]}(u', \delta_{u'})$. On the one hand the induction hypothesis on $V^{[u]}$ implies $\delta_{v'} \geq \delta_{u'} \geq -\overline{d}$. On the other hand if $u' \neq N^{[u]}$ we have by assumption $\delta_{u'} = 0$ and therefore
$$\delta_{v'} - \underbrace{\delta_{u'}}_{=0} + \underbrace{\underline{d}([u],[v])}_{\geq 0} \leq \overline{d}, \text{ hence } \delta_{v'} \leq \overline{d}.$$
If $u' = N^{[u]}$, (10) implies $\min d((u', N^{[v]})) = \underline{d}([u],[v])$ and therefore
$$\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) = \underline{d}([u],[v]) \quad \Leftrightarrow \quad \delta_{v'} = \delta_{u'}, \text{ thus } \delta_{v'} \leq \overline{d}.$$

2. $(v', \delta_{v'}) \in W_3^{[v]}$, then there must be a $(u', \delta_{u'}) \in [u] \times \mathbb{Z}_-$ and a $(\tilde{v}, \delta_{\tilde{v}}) \in [v] \times \mathbb{N}$ with $\delta_{\tilde{v}} - \delta_{u'} + \underline{d}([u],[v]) \in d((u', \tilde{v}))$. Because $\delta_{\tilde{v}} > 0$ we have
$$\underbrace{\delta_{\tilde{v}}}_{>0} - \delta_{u'} + \underbrace{\underline{d}([u],[v])}_{\geq 0} \leq \overline{d}, \text{ hence } \delta_{u'} > -\overline{d},$$
which implies by (11)
$$\delta_{v'} \geq \delta_{u'} > -\overline{d},$$
and similarly
$$\delta_{v'} - \underbrace{\delta_{u'}}_{\leq 0} + \underline{d}([u],[v]) \leq \overline{d}, \text{ thus } \delta_{v'} \leq \overline{d}.$$

3. $(v', \delta_{v'}) \in W_4^{[v]}$, then there must be a $(u', \delta_{u'}) \in [u] \times \mathbb{N}$ and a $(\tilde{v}, \delta_{\tilde{v}}) \in [v] \times \mathbb{N}$ with $\delta_{\tilde{v}} - \delta_{u'} + \underline{d}([u],[v]) \in d((u', \tilde{v}))$ and $\delta_{\tilde{v}} \leq \overline{t}(W_2^{[v]} \cup W_3^{[v]}) \leq \overline{d}$ (by the previous two cases). Furthermore, like in the previous case, $\delta_{\tilde{v}} > 0$ implies $\delta_{u'} > -\overline{d}$ and together with (11) we have $-\overline{d} < \delta_{u'} \leq \delta_{v'}$. □

The corridor giving the fastest continuation along any clone path corresponds to the case where the concrete time shifts $t_{[u]}$ for the sets $V^{[u]}$ are chosen so that $V_{t_{[u]}}^{[u]} \subset V_T \setminus V_T^{\lambda}$ for $[u] \in [V]$ and so that for $([u],[v]) \in [A]$ the difference of time shifts is $t_{[v]} - t_{[u]} = \underline{d}([u],[v])$. In this case the properties (H1)–(H4) permit a construction of G_T^+ similar to the fastest route approach of the previous section.

Remark 18 In fact, the difference between the fastest route approach of the previous section and the fastest continuation corridor above is caused by the structural properties of the clone nodes. Using clone nodes it is sufficient to include the fastest paths along each sequence of clones instead of each sequence of nodes. For this corridor the property still holds that no path leaving the graph can return (this would require an even faster arc). If the cost function is defined in terms of clone nodes, *i. e.*, depends only on the sequence of clone nodes and the corresponding time steps but not on the specific choice of nodes, the constructed graph is already a valid subnetwork of G_T. In particular, this is true for cost functions satisfying (C), but also other cost functions like those mentioned in Remark 12 are possible.

In order to improve the corridor we allow larger time shifts. These must be selected carefully in order to preserve the continuation and interception properties. Larger mutual time offsets than $\underline{d}([u],[v])$ typically require the availability of a waiting node $v' \in [v]$ with $(v', v') \in A$ that allows to bridge the time difference. The possibility to use such waiting paths is incorporated below in the extended versions of the properties (H1)–(H4) formulated w.r.t. two sets $V^{[u]}_{t_{[u]}}$ and $V^{[v]}_{t_{[v]}}$ with $([u],[v]) \in [A]$ and sufficiently large $t_{[u]}, t_{[v]} \in T$. In fact, (H1)–(H4) imply those extended properties if $t_{[v]} - t_{[u]} = \underline{d}([u],[v])$ which ensures the existence of feasible time-shifts (we will proof these statements formally below).

Definition 19 Let $([u],[v]) \in [A]$ and let $t_{[u]}$, $t_{[v]} \in T$ satisfy $V^{[u]}_{t_{[u]}}, V^{[u]}_{t_{[u]}} \subset V_T \setminus V^{\lambda}_T$. The two shifted interception sets $V^{[u]}_{t_{[u]}}$ and $V^{[v]}_{t_{[v]}}$ *match for* $([u],[v])$ if the following conditions hold

(H1') $[v] \times \{t_{[v]}\} \subseteq V^{[v]}_{t_{[v]}}$,

(H2') *(extended continuation property)*
for $(u', t_{u'}) \in V^{[u]}_{t_{[u]}}$ there exists a path $P = (u', t_{u'})(v', t_{v'}) \ldots (v', t'_{v'}) \subset A_T \setminus A^{\lambda}_T$ with $(v', t'_{v'}) \in V^{[v]}_{t_{[v]}}$,

(H3') *(extended interception property)*
for $(u', t_{u'}) \in [u] \times T$ with $t_{u'} \leq t_{[u]}$ and $(v', t_{v'}) \in [v] \times T$ with $t_{v'} > t_{[v]}$ and $((u', t_{u'}), (v', t_{v'})) \in A_T$ there is a path $P' = (u', t_{u'})(v'', t_{v''}) \ldots (v'', t'_{v''}) \subset A_T \setminus A^{\lambda}_T$ with $(v'', t'_{v''}) \in V^{[v]}_{t_{[v]}}$ and $t_{v''} \leq t_{v'}$,

(H4') *(extended reinterception property)*
for $(u', t_{u'}) \in [u] \times T$ with $t_{u'} > t_{[u]}$ and $(v', t_{v'}) \in [v] \times T$ with $t_{[v]} < t_{v'} \leq \bar{t}(V^{[v]}_{t_{[v]}})$ and $((u', t_{u'}), (v', t_{v'})) \in A_T$ there is a path $P' = (u', t_{u'})(v'', t_{v''}) \ldots (v'', t'_{v''}) \subset A_T \setminus A^{\lambda}_T$ with $(v'', t'_{v''}) \in V^{[v]}_{t_{[v]}}$ and $t_{v''} \leq t_{v'}$.

Fig. 14 illustrates the extended properties.

Observation 20 *Let* $([u],[v]) \in [A]$ *and* $t_{[u]}, t_{[v]} \in T$ *such that* $V^{[u]}_{t_{[u]}}, V^{[v]}_{t_{[v]}} \subset V_T \setminus V^{\lambda}_T$ *and* $t_{[v]} - t_{[u]} = \underline{d}([u],[v])$. *Then* $V^{[u]}_{t_{[u]}}$ *and* $V^{[v]}_{t_{[v]}}$ *match.*

Proof. Note that for $t_{[v]} - t_{[u]} = \underline{d}([u],[v])$ the condition $((u', \delta_{u'} + t_{[u]}), (v', \delta_{v'} + t_{[v]})) \in A_T$ is equivalent to $\delta_{v'} - \delta_{u'} + \underline{d}([u],[v]) \in d((u', v'))$. For $V^{[u]}_{t_{[u]}}, V^{[v]}_{t_{[v]}} \subset V_T \setminus V^{\lambda}_T$ the definition of shifted interception nodes, $V^{[u]}_{t_{[u]}} = \{(u', \delta_{u'} + t_{[u]}) \colon (u', \delta_{u'}) \in V^{[u]}\}$, and properties (H1)–(H4) therefore ensure that each of the paths required by (H1')–(H4') can indeed be realized by a single arc in A_T. This arc cannot be in A^{λ}_T because the corresponding head node is in $V^{[v]}_{t_{[v]}} \subset V_T \setminus V^{\lambda}_T$. □

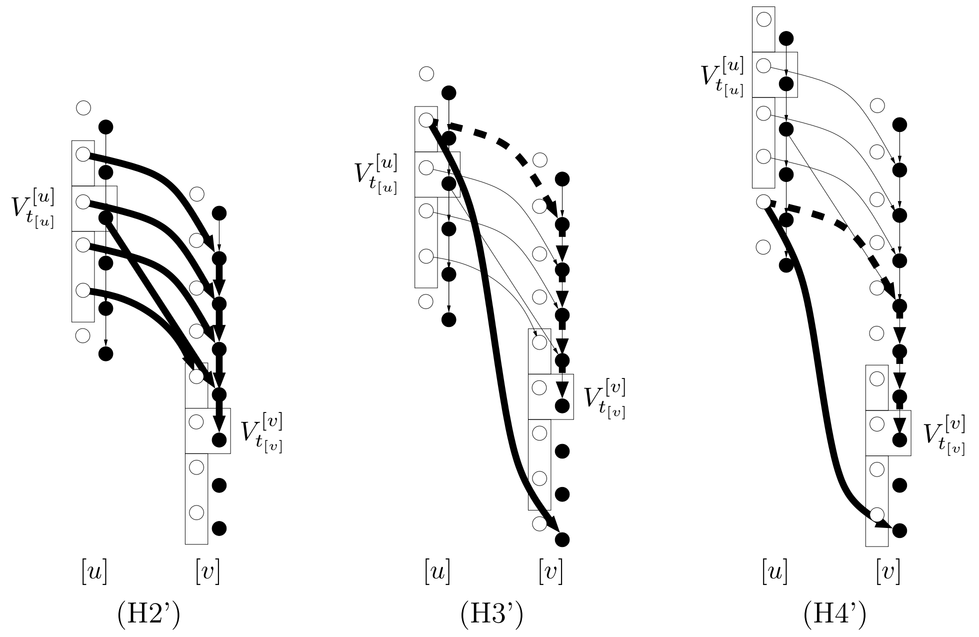


Figure 14: Properties (H2')–(H4') are the same as (H2)–(H4) but instead of direct connecting arcs additional waiting arcs may be used as long as the path is contained in $G_T \setminus G_T^\lambda$. In particular, (H3') and (H4') are relevant if different running-times are possible between successive nodes, *i. e.*, $|d((u,v))| > 1$ for some $u \in [u], v \in [v]$.

Considering now a clone node with several outgoing clone arcs, it will, in general, not be possible to have just one copy of an interception set so as to connect to all successive interception sets. Therefore, for each clone arc $([u],[v]) \in [A]$ a matching time shift $t_{[u]}$ will have to be available within a set of time shifts $T^{[u]}$ for each time shift $t_{[v]}$ in the set of time shifts $T^{[v]}$ of $[v]$. If $[u]$ contains a waiting node $u' \in [u]$ with (u', u') it may be possible to collapse some of them and it is one goal of the design to keep the sets $T^{[u]}$ small.

Definition 21 Let $([u],[v]) \in [A]$ and let, for $[w] \in \{[u],[v]\}$, $T^{[w]} \subset T$ satisfy $V_t^{[w]} \subset V_T \setminus V_T^\lambda$ for $t \in T^{[w]}$. The two sets $T^{[u]}$ and $T^{[v]}$ *match*, if for every $t_{[v]} \in T^{[v]}$ there is a matching $t_{[u]} \in T^{[u]}$, *i. e.*, $V^{[u]}_{t_{[u]}}$ and $V^{[v]}_{t_{[v]}}$ match.

A family $\mathfrak{T} = \{T^{[u]} \subset T\}_{[u]\in[V]}$ of nonempty finite sets is called *admissible time shifts (for interception sets $V^{[u]}$, $[u] \in [V]$, and G_T^λ)* or simply *admissible* if for each $([u],[v]) \in [A]$ the sets $T^{[u]}$ and $T^{[v]}$ match.

Note that for $([u],[v]) \in [A]$ the requirement for matching sets $T^{[u]}$ and $T^{[v]}$ is not symmetric: not every $t \in T^{[u]}$ needs a matching $t' \in T^{[v]}$. The existence of admissible time shifts is not difficult to see but essential.

Observation 22 *For any G_T^λ there exist admissible time shifts $T^{[u]} \subset T$, $[u] \in [V]$.*

Proof. Define the sets in reverse topological order starting with $T^{[\overline{u}]} = \{\tau\}$ for some $\tau \in \mathbb{N}$ large enough. For each $[u] \in [V]$ with the sets $T^{[v]}$ already defined for all $([u],[v]) \in [A]$, put $T^{[u]} = \{t - \underline{d}([u],[v]) \colon t \in T^{[v]}, ([u],[v]) \in [A]\}$. By Observation 20 and τ large enough this ensures that for each $([u],[v]) \in [A]$ the sets $T^{[u]}$ and $T^{[v]}$ match.□

Admissible time shifts for G_T^λ allow to find matching time shifts $t_{[u_i]}$ along any path $P = [u_1] \dots [u_k] \subset [G]$ starting with an arbitrary $t_{[u_k]} \in T^{[u_k]}$ in the last node of the path.

Proposition 23 *Let $\mathfrak{T} = \{T^{[u]}\}_{[u]\in[V]}$ be admissible and let $[u_1] \dots [u_k] \subseteq [A]$ be a clone-path. For each $t_{[u_k]} \in T^{[u_k]}$ there exist $t_{[u_i]} \in T^{[u_i]}$, $i = 1, \dots, k-1$, so that $t_{[u_i]}$ and $t_{[u_{i+1}]}$ match. Given any such choice, for all $j, j' \in \{1, \dots, k\}$, $j \leq j'$, and all $(v_j, t'_j) \in V_{t_{[u_j]}}^{[u_j]}$ there is a path*

$$P = (v_j, t'_j)(v_{j+1}, t_{j+1}) \dots (v_{j+1}, t'_{j+1}) \dots (v_{j'}, t'_{j'}) \subset A_T \setminus A_T^\lambda$$

with $(v_i, t'_i) \in V_{t_{[u_i]}}^{[u_i]}$ for all $i \in \{j, \dots, j'\}$.

Proof. Pick any $t_{[u_k]} \in T^{[u_k]}$ and continue recursively for $i = k-1, \dots, 1$ by picking a $t_{[u_i]} \in T^{[u_i]}$ so that $V_{t_{[u_i]}}^{[u_i]}$ and $V_{t_{[u_{i+1}]}}^{[u_{i+1}]}$ match (this is possible by definition because the $T^{[u]} \subset T$, $[u] \in [V]$, are admissible). The second claim now follows from applying (H2') iteratively starting from (v_j, t'_j). □

Given admissible time shifts $\mathfrak{T}$ for G_T^λ, we consider the graph $G_T^{\mathfrak{T}} = (V_T^{\mathfrak{T}}, A_T^{\mathfrak{T}}) = \operatorname{cl}(\bigcup_{[u]\in[V]} V_{\max T^{[u]}}^{[u]}, \emptyset)$. The following technical proposition justifies the first part of our construction, namely, each path leaving $G_T^{\mathfrak{T}}$ without returning to $G_T^{\mathfrak{T}}$ can be intercepted and replaced by a path staying in $G_T^{\mathfrak{T}} \setminus G_T^\lambda$.

Proposition 24 *Given admissible time shifts $\mathfrak{T} = \{T^{[u]}\}_{[u]\in[V]}$ for interception sets $V^{[u]}$, $[u] \in [V]$, and G_T^λ, let $G_T^{\mathfrak{T}} = \operatorname{cl}(\bigcup_{[u]\in[V]} V_{\max T^{[u]}}^{[u]}, \emptyset)$ and*

$$P = (u_1, t_1) \dots (u_1, t'_1)(u_2, t_2) \dots (u_2, t'_2) \dots (u_k, t'_k) \subset A_T \setminus A_T^\lambda$$

be a path with $k > 1$, $[u_i] \neq [u_j]$ for all $1 \leq i < j \leq k$, $(u_1, t_1) \in \partial A_T^\lambda \cap V_T^{\mathfrak{T}}$ and $t'_k > t_{[u_k]}$ for some $t_{[u_k]} \in T^{[u_k]}$. Then there is a path $P' = (u_1, t_1)P' \leq_T P$ with $P' \subset A_T \setminus A_T^\lambda$ having its last node in $V_{t_{[u_k]}}^{[u_k]}$.

Proof. Let $t_{[u_1]}, \dots, t_{[u_k]}$ denote time-shifts associated with the clone-path P of $[P]$ as defined by Proposition 23. Let $(u_i, t) \in V_T(P)$ be the node with

$$t = \min\left\{t'_i, \bar{t}\left(V_{t_{[u_i]}}^{[u_i]}\right)\right\} \quad \text{and} \quad t_j > \bar{t}\left(V_{t_{[u_j]}}^{[u_j]}\right) \text{ for } i < j \leq k$$

(this node exists because $t_1 \leq t_{[u_1]} \leq \bar{t}(V_{t_{[u_1]}}^{[u_1]})$). Depending on this time t we have to consider three cases to construct the replacing path P'.

1. If $t_i \leq t < t_{[u_i]}$ then $(u_i, t) \neq (u_k, t'_k)$. Because $t + 1 \leq t_{[u_i]} \leq \bar{t}(V^{[u_i]}_{t_{[u_i]}})$ the choice of t implies $t = t'_i$ and $((u_i, t), (u_{i+1}, t_{i+1})) \in P$ with $t_{i+1} > \bar{t}(V^{[u_{i+1}]}_{t_{[u_{i+1}]}}) \geq t_{[u_{i+1}]}$. Now property (H3') implies the existence of a path $Q_1 = (u_i, t) \ldots (\tilde{u}_{i+1}, \tilde{t}_{i+1}) \subset A_T \setminus A^\lambda_T$ with $(\tilde{u}_{i+1}, \tilde{t}_{i+1}) \in V^{[u_{i+1}]}_{t_{[u_{i+1}]}}$ and $\tilde{t}_{i+1} < t_{i+1}$. By Proposition 23 there is a path $Q_2 = (\tilde{u}_{i+1}, \tilde{t}_{i+1}) \ldots (\tilde{u}_k, \tilde{t}_k) \subset A_T \setminus A^\lambda_T$ with $Q_2 \leq_T (u_{i+1}, t'_{i+1})P$. Connecting these parts together we get a path $P' = P(u_i, t)Q_1(\tilde{u}_{i+1}, \tilde{t}_{i+1})Q_2(\tilde{u}_k, \tilde{t}_k)$ with the property $P' \leq_T P$.

2. If $t_i \leq t_{[u_i]} \leq t$ then $(u_i, t_{[u_i]}) \in V_T(P)$ and (H1') implies $(u_i, t_{[u_i]}) \in V^{[u_i]}_{[t_{u_i}]}$. If $[u_i] = [u_k]$ then $P' = P(u_i, t_{[u_i]})$ satisfies the requirements. Otherwise $[u_i] \neq [u_k]$ and we use Proposition 23 to get a path $Q = (u_i, t_{[u_i]}) \ldots (\tilde{u}_k, \tilde{t}_k) \subset A_T \setminus A^\lambda_T$ with $(\tilde{u}_k, \tilde{t}_k) \in V^{[u_k]}_{[t_{u_k}]}$. The choice of t ensures $Q \leq_T (u_i, t_{[u_i]})P(u_k, t'_k)$. The path $P' := P(u_i, t_{[u_i]})Q(\tilde{u}_k, \tilde{t}_k)$ has the desired property.

3. If $t_{[u_i]} < t_i \leq t$ we know by assumption that $(u_i, t_i) \neq (u_1, t_1)$. Depending on whether the time step t'_{i-1} of the predecessor node (u_{i-1}, t'_{i-1}) is not greater or greater than $t_{[u_{i-1}]}$ either (H3') or (H4') ensure that there is a path $Q_1 := (u_{i-1}, t'_{i-1}) \ldots (\tilde{u}_i, \tilde{t}_i) \subset A_T \setminus A^\lambda_T$ with $(\tilde{u}_i, \tilde{t}_i) \in V^{[u_i]}_{t_{[u_i]}}$ and $\tilde{t}_i \leq t_i$. Using Proposition 23 there is a path $Q_2 = (\tilde{u}_i, \tilde{t}_i) \ldots (\tilde{u}_k, \tilde{t}_k)$ with $Q_2 \leq_T (u_i, t_i)P$ and the path
$$P' := P(u_{i-1}, t'_{i-1})Q_1(\tilde{u}_i, \tilde{t}_i)Q_2(\tilde{u}_k, \tilde{t}_k)$$
has the desired property. □

As mentioned before, the node sets $\bigcup\{V^{[u]}_{t_{[u]}}\}$ are the main building part of V^+_T for a path $[P]$, but this is not sufficient by itself. Indeed, if we do not include the fastest paths along all node sequences, it may well happen that paths may reenter the graph $G^{\mathfrak{T}}_T$, in particular in regions where the difference in time shifts is big. Rigorously, a ∂A^λ_T-path P is called *reentering* if it leaves $G^{\mathfrak{T}}_T$, *i. e.* $V_T(P) \cap (V_T \setminus V^{\mathfrak{T}}_T) \neq \emptyset$, and later returns to $V^{\mathfrak{T}}_T$. For such a path the following situation might arise. We would like to replace the part of P that is not contained in $G^{\mathfrak{T}}_T$ by a path P'. Assume (w, t_w) is the first node of P when P *returns* to $G^{\mathfrak{T}}_T$. Then we require $P'(w', t'_w) \leq_T P(w, t_w)$ with $w' \in [w]$. Because the remaining part of P, namely $(w, t_w)P$, may be contained in G^λ_T it cannot be replaced in general. But this implies that the replacing path P' has to meet P in (w, t_w) while satisfying $P' \leq_T P$. In some cases this is simply impossible within $G^{\mathfrak{T}}_T$ (see Fig. 15 for an example).

The last step of the construction of G^C_T therefore adds some further nodes and arcs to $G^{\mathfrak{T}}_T$ in those situations where connections as described above may be missing. In fact, they can conveniently be described as those nodes of G^λ_T-irreducible paths that are not contained in $V^{\mathfrak{T}}_T$.

Definition 25 Let $\mathfrak{T} = \{T^{[u]}\}_{[u] \in [V]}$ be admissible and put $G^{\mathfrak{T}}_T = \mathrm{cl}(\bigcup_{[u] \in [V]} V^{[u]}_{\max T^{[u]}}, \emptyset)$. A node $(u, t_u) \in V_T \setminus V^{\mathfrak{T}}_T$ is *bad (for $G^{\mathfrak{T}}_T$)* if it is contained in an G^λ_T-irreducible path. The set of all bad nodes is denoted by $B^{\mathfrak{T}}$.

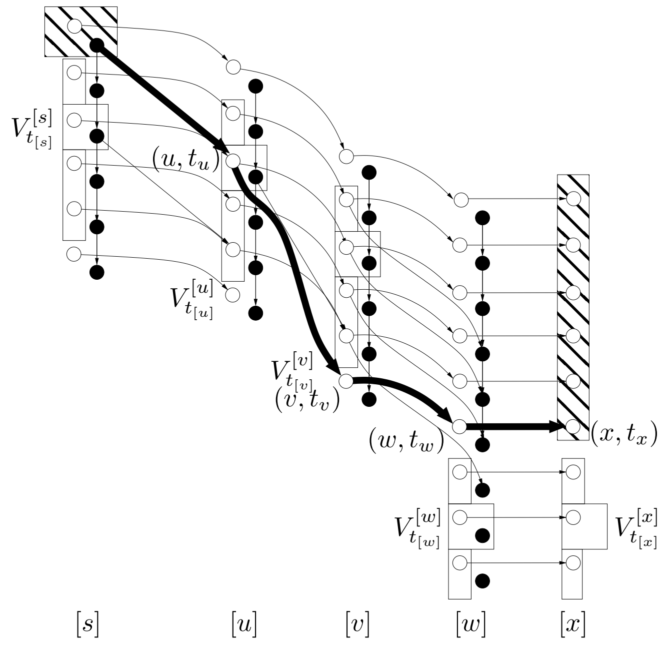


Figure 15: Reentering paths. The hatched region denotes G^{λ}_{T}. The thick ∂A_T-path $P = (s, t_s)(u, t_u)(v, t_v)(w, t_w)(x, t_x)$ has no replacing path because there is no path from (u, t_u) to (w, t_w) within $V^{[u]}_{t_{[u]}}$, $V^{[v]}_{t_{[v]}}$ and $V^{[w]}_{t_{[w]}}$. Therefore the node (v, t_v) has to be included in G^{C}_{T}.

Observation 26 *The last node of any G^{λ}_{T}-irreducible path is contained in $G^{\mathfrak{T}}_{T}$. In particular,*

$$\bar{t}(B^{\mathfrak{T}} \cap ([\overline{u}] \times T)) \leq \bar{t}(V^{\mathfrak{T}}_{T} \cap ([\overline{u}] \times T)) = \bar{t}(V^{[\overline{u}]}_{\max T^{[\overline{u}]}}). \tag{12}$$

Proof. Because $\bar{t}(V^{\lambda}_{T} \cap ([u] \times T)) < \bar{t}(V^{[u]}_{\max T^{[u]}})$ holds for each $[u] \in [V]$, the requirement for $(u, t_u) \in \partial A^{\lambda}_{T}$ to satisfy $t_u > \bar{t}(V^{[u]}_{\max T^{[u]}})$ implies $u = \overline{u}$. So it suffices to show that any ∂A^{λ}_{T}-path P ending in $(\overline{u}, t_{\overline{u}})$ with $t_{\overline{u}} > \bar{t}(V^{[u]}_{\max T^{[u]}})$ is G^{λ}_{T}-reducible. For this, use Proposition 24 with respect to the path P to find a path $Q <_T P$ in $A_T \setminus A^{\lambda}_{T}$ ending in a node $(\overline{u}, t^{0}_{\overline{u}}) \in V^{[\overline{u}]}_{t_{[\overline{u}]}}$. Because $(\overline{u}, \overline{u}) \in A$ and $t^{0}_{\overline{u}} \leq \bar{t}(V^{[u]}_{\max T^{[\overline{u}]}}) < t_{\overline{u}}$, this path can be extended to $Q(\underline{u}, t^{0}_{\overline{u}}) \ldots (\underline{u}, t_{\overline{u}}) <_T P$ and therefore P is reducible. □

This shows that the set of bad nodes is finite. To obtain a valid subnetwork it suffices to add all bad nodes to $G^{\mathfrak{T}}_{T}$, because then all G^{λ}_{T}-irreducible paths are included in this graph.

Corollary 27 *The set of bad nodes satisfies $B^{\mathfrak{T}} \subseteq [V] \times \{1, \ldots, \bar{t}(V^{\mathfrak{T}}_{T})\}$ and $\operatorname{cl}(V^{\mathfrak{T}}_{T} \cup B^{\mathfrak{T}}, \emptyset)$ is a valid subnetwork whenever (C) holds.*

Proof. By Observation 26 any node $(u, t_u) \in B^{\mathfrak{T}}$ lies on an irreducible path $P(v, t_v)$ for some end node $(v, t_v) \in V^{\mathfrak{T}}_{T}$. Because $\underline{d}([u], [v]) \geq 0$, any predecessor (u', t'_u) of (v, t_v) on

P must satisfy $t'_u \le t_v$, so $t_u \le t_v \le \bar{t}(V_T^{\mathfrak{T}})$. Because $\mathrm{cl}(V_T^{\mathfrak{T}} \cup B^{\mathfrak{T}}, \emptyset)$ is finite and contains all G_T^λ-irreducible paths, the result follows by Corollary 10. $\square$

Given upper bounds on the time indices of bad nodes belonging to successor clone nodes, a rough bound on the bad nodes of the clone node itself is readily deduced via the usual fastest continuation argument.

Observation 28 *Suppose $\mathfrak{T} = \{T^{[u]}\}_{[u]\in[V]}$ is admissible. Given $[u] \in [V] \setminus \{[\overline{u}]\}$ and for each $([u],[v]) \in [A]$ a bound $\bar{b}_{[v]} \ge \bar{t}((V_T^{\mathfrak{T}} \cup B^{\mathfrak{T}}) \cap ([v] \times T))$, then any $t_{[u]} \in T^{[u]}$ matching some $t_{[v]} \in T^{[v]}$ with $([u],[v]) \in [A]$ satisfies*

$$\bar{t}(V_{t_{[u]}}^{[u]}) \le \bar{t}(V_{t_{[v]}}^{[v]}) - \underline{d}([u],[v]) \le \bar{b}_{[v]} - \underline{d}([u],[v]). \tag{13}$$

Furthermore, any node $(u', t_{u'}) \in [u] \times T$ lying on a G_T^λ-irreducible path using an edge $((u', t'_{u'}), (v, t_v))$ with $[v] \ne [u]$ fulfills $t'_u \le \bar{b}_{[v]} - \underline{d}([u],[v])$. In particular, if

$$\bar{t}(V_{\max T^{[u]}}^{[u]}) \le \max_{([u],[v])\in[A]} \{\bar{b}_{[v]} - \underline{d}([u],[v])\}, \tag{14}$$

then

$$\bar{t}((V_T^{\mathfrak{T}} \cup B^{\mathfrak{T}}) \cap ([u] \times T)) \le \max_{([u],[v])\in[A]} \{\bar{b}_{[v]} - \underline{d}([u],[v])\}. \tag{15}$$

Proof. (13) follows directly from (H2'). Thus, the requirement stated in (14) is no relevant restriction. In order to show (15) suppose $(u', t_{u'}) \in (V_T^{\mathfrak{T}} \cup B^{\mathfrak{T}})$ with $u' \in [u]$. If $(u', t_{u'}) \in V_T^{\mathfrak{T}}$, we have $t'_u \le \bar{t}(V_{\max T^{[u]}}^{[u]})$ and the bound follows from (14), so we may assume $(u', t_{u'}) \in B^{\mathfrak{T}}$. By Observation 26 any G_T^λ-irreducible path P containing $(u', t_{u'})$ must have an end node in $V_T^{\mathfrak{T}}$. Because $G_T^{\mathfrak{T}}$ is closed and $t_{u'} > \bar{t}(V_T^{\mathfrak{T}} \cap ([U] \times T))$, this end node cannot be in $[u] \times T$. Therefore P must be of the form $P(u', t_{u'}) \ldots (u', t'_{u'})(v', t_{v'})P$ with $v' \in [v]$ for some $([u],[v]) \in [A]$ and $t_{u'} \le t'_{u'} \le t_{v'} - \underline{d}([u],[v])$. Either $(v', t_{v'}) \in V_T^{\mathfrak{T}}$ or $(v', t_{v'}) \in B^{\mathfrak{T}}$ (because it lies on the irreducible path P). In both cases (15) holds. $\square$

The next result presents a feasible interval of admissible larger time shifts with respect to a successor clone node that contains at least one waiting node. These larger time shifts allow to bring $G_T^{\mathfrak{T}}$ closer to G_T^λ but do not yet help to reduce the bound on the time indices of the bad nodes.

Proposition 29 *Given a node $v \in V$ with $(v, v) \in A$, a time shift $t_{[v]} \in T$ with $V_{t_{[v]}}^{[v]} \subset V_T \setminus V_T^\lambda$ and an arc $([u],[v]) \in [A]$, put*

$$\begin{aligned}
\underline{\delta}_{[u]} &:= \min\{\delta_{u'} \colon (u', \delta_{u'}) \in V^{[u]}\} \quad (\le 0),\\
\bar{\delta}_{[u]} &:= \max\{\delta_{u'} \colon (u', \delta_{u'}) \in V^{[u]}\} \quad (\ge 0),\\
\underline{\tau}_{[u]}^v &:= \max\left\{1 + \bar{t}(V_T^\lambda \cap ([u] \times T)), \bar{t}(V_T^\lambda \cap ([v] \times T)) - \underline{d}([u], v)\right\} - \underline{\delta}_{[u]}, \qquad (16)\\
\bar{d}_{[u]}^v &:= \max\left\{\bar{d}([u],[v]), \bar{\delta}_{[u]} + \max_{u'\in[u]} \underline{d}(u', v)\right\}. \qquad (17)
\end{aligned}$$

For any t with
$$\underline{\tau}^v_{[u]} \le t \le t_{[v]} - \overline{d}^v_{[u]} \tag{18}$$
the sets $V^{[u]}_t$ and $V^{[v]}_{t_{[v]}}$ match.

Proof. Suppose t satisfies (18). Note that for $(u', t_{u'}) = (u', t+\delta_{u'}) \in V^{[u]}_t$ with $(u', \delta_{u'}) \in V^{[u]}$ we have $t_{u'} \overset{(18)}{\ge} \underline{\tau}^v_{[u]} + \underline{\delta}_{[u]} \overset{(16)}{\ge} 1 + \bar{t}(V^\lambda_T \cap ([u] \times T))$, so $V^{[u]}_t \cap V^\lambda_T = \emptyset$ as required in Definition 19. Condition (H1') is satisfied, because $V^{[v]}$ satisfies (H1). To see that condition (H2') holds, observe that any node $(u', t_{u'}) \in V^{[u]}_t$ — in fact, any node $(u', t_{u'})$ with $t_{u'} \in [t + \underline{\delta}_{[u]}, t + \overline{\delta}_{[u]}]$ — has an arc $((u', t_{u'}), (v, t_v)) \in A_T \setminus A^\lambda_T$ with $t_v - t_{u'} \in [\underline{d}([u], v), \max_{u'' \in [u]} \underline{d}(u'', v)]$, so

$$t_v \ge t_{u'} + \underline{d}([u], v) \ge \underline{\tau}^v_{[u]} + \underline{\delta}_{[u]} + \underline{d}([u], v) \overset{(16)}{\ge} \bar{t}(V^\lambda_T \cap ([v] \times T))$$

and

$$t_v \le t_{u'} + \max_{u'' \in [u]} \underline{d}(u'', v) \le t + \overline{\delta}_{[u]} + \max_{u'' \in [u]} \underline{d}(u'', v) \overset{(17)}{\le} t + \overline{d}^v_{[u]} \overset{(18)}{\le} t_{[v]}.$$

Therefore this arc can be continued by a path $(v, t_v) \dots (v, t_{[v]}) \in A_T \setminus A^\lambda_T$ with $(v, t_{[v]}) \in V^{[v]}_{t_{[v]}}$ by (H1'). This proves (H2') and, by the same line of arguments, (H4'), as well. Because $t_{[v]} - t \overset{(18)}{\ge} \overline{d}^v_{[u]} \overset{(17)}{\ge} \overline{d}([u], [v])$, there can be no arcs that give rise to requirements in (H3') and so (H3') holds. This shows that $V^{[u]}_t$ and $V^{[v]}_{t_{[v]}}$ match. □

One possibility to obtain better bounds on the time indices of bad nodes is to show that beyond a certain time index all nodes must be part of G^λ_T-reducible paths. The corridor $G^{\mathcal{T}}_T$ offers good possibilities to design earlier paths that lead up to the same clone node at which the path in question is still outside $G^{\mathcal{T}}_T$. If it is possible to link into the original path again coming from a slightly earlier node within this clone node, the path is shown to be reducible. In order to do this only on basis of local information for all larger time steps, we need to require the possibility to wait at a node that offers a sufficiently slow connection to the next node on the path. The precise requirements in view of several possible clone node successors are collected in the next result.

Proposition 30 *Let $\mathcal{T} = \{T^{[u]}\}_{[u] \in [V]}$ be admissible time shifts and $([u], [v]) \in [A]$ be an arc such that the following properties hold:*

(i) There exists a $t_{[u]} \in T^{[u]}$ so that for all $v' \in [v]$ with $(v', v') \in A$ and for all $(u', t_{u'}) \in V^{[u]}_{t_{[u]}}$ there is an arc $((u', t_{u'}), (v', t_{v'})) \in A_T$ with $t_{v'} \ge \bar{t}(V^\lambda_T \cap ([v] \times T))$.

(ii) For each $([v], [w]) \in [A]$ there is a node $v_{[w]} \in [v]$ with $(v_{[w]}, v_{[w]}) \in A$ so that $\overline{d}(v_{[w]}, w') \ge \overline{d}([v], w')$ for all $w' \in [w]$.

Then there is no G_T^λ-irreducible path containing an arc $((u', t_{u'}), (v', t_{v'}))$ with $u' \in [u]$, $v' \in [v]$ and

$$t_{u'} > \max\left\{\bar{t}(V_{t_{[u]}}^{[u]}),\, \bar{t}^{t_{[u]}}_{([u],[v])} + \max_{([v],[w])\in[A]}\{\bar{d}(v_{[w]}, [w]) - \underline{d}([v],[w])\} - \underline{d}([u],[v])\right\} \tag{19}$$

where

$$\bar{t}^{t_{[u]}}_{([u],[v])} := \max_{\substack{v'\in[v],\\ (v',v')\in A}} \max_{(u',t_{u'})\in V_{t_{[u]}}^{[u]}} \min\{t_{v'} \geq \bar{t}(V_T^\lambda \cap ([v]\times T))\colon ((u',t_{u'}),(v',t_{v'})) \in A_T\}. \tag{20}$$

Proof. Given $([u], [v]) \in [A]$ so that (i) and (ii) hold, let P be a ∂A_T^λ-path containing an arc $((u', t_{u'}), (v', t_{v'}))$ with $u' \in [u]$, $v' \in [v]$ and $t_{u'}$ satisfying (19). Then

$$t_{v'} \geq t_{u'} + \underline{d}([u],[v]) > \bar{t}^{t_{[u]}}_{([u],[v])} + \max_{([v],[w])\in[A]}\{\bar{d}(v_{[w]},[w]) - \underline{d}([v],[w])\} \geq \bar{t}(V_T^\lambda \cap ([v]\times T)). \tag{21}$$

Use Proposition 24 with respect to the path $P(u', t_{u'})$ and $t_{[u]}$ to find a path $Q <_T P(u', t_{u'})$ in $A_T \setminus A_T^\lambda$ ending in a node $(u'', t_{u''}) \in V_{t_{[u]}}^{[u]}$ ($t_{u''} < t_{u'}$ by (19)).

First consider the case $(v', v') \in A$. By (i) and (20) there is an arc $((u'', t_{u''}), (v', t_{v'}^0)) \in A_T$ with $\bar{t}(V_T^\lambda \cap ([v]\times T)) \leq t_{v'}^0 \leq \bar{t}^{t_{[u]}}_{([u],[v])}$ and by (21) $t_{v'} > \bar{t}^{t_{[u]}}_{([u],[v])}$, thus there is a path $Q' = (u'', t_{u''})(v', t_{v'}^0)\ldots(v', t_{v'}) \subset A_T \setminus A_T^\lambda$. Hence, $Q(u'', t_{u''})Q'(v', t_{v'})P <_T P$ and P is reducible.

So we may assume $(v', v') \notin A$. Note that $(v', t_{v'}) \notin V_T^\lambda$ because $t_{v'} > \bar{t}(V_T^\lambda \cap ([v]\times T))$ by (21). Thus, it has a successor $(w', t_{w'})$ in P with $[w'] \neq [v]$ and

$$\begin{aligned} t_{w'} - \bar{d}(v_{[w']}, w') &\geq t_{v'} + \underline{d}([v],[w']) - \bar{d}(v_{[w']}, w') \\ &\geq t_{v'} - \max_{([v],[w])\in[A]}\{\bar{d}(v_{[w]},[w]) - \underline{d}([v],[w])\} \overset{(21)}{>} \bar{t}^{t_{[u]}}_{([u],[v])}. \end{aligned}$$

In consequence and using assumption (ii) there is an arc $((v_{[w']}, t'_{v_{[w']}}), (w', t_{w'})) \in A_T$ with $t_{w'} - t'_{v_{[w']}} = \bar{d}(v_{[w']}, w')$, thus $\bar{t}^{t_{[u]}}_{([u],[v])} < t'_{v_{[w']}} \leq t_{v'}$. Exploiting (i) as in the first case this allows to find a path $Q' = (u'', t_{u''})(v_{[w']}, t_{v_{[w']}})\ldots(v_{[w']}, t'_{v_{[w']}}) \subset A_T \setminus A_T^\lambda$ with $Q' <_T (u', t'_u)P(v', t'_v)$. Now $Q(u'', t_{u''})Q'(v_{[w']}, t'_{v_{[w']}})(w', t_{w'})P <_T P$, so P is reducible.□

The requirements of the result seem rather stringent but are quite natural in practice where there is typically just one waiting node in each clone node, and where starting from a waiting position takes longer than continuing in full speed. Of course, the improved bound along $([u], [v])$ is helpful in $[u]$ only if there is no other successor of $[u]$ giving rise to a worse bound.

An algorithmic possibility to construct admissible time shifts and upper bounds on the time indices of bad nodes is as follows (it uses the notations introduced in Proposition 29 and Proposition 30).

Algorithm 31 *Given interception node sets $V^{[u]}$, $[u] \in [V]$, find time shifts and upper time limits that include all bad nodes for constructing G_T^C.*

```
1 Construct lower limits for the time shifts (in topological order):
  $\underline{t}_{[\underline{u}]} := 1 + \overline{t}(V_T^\lambda \cap ([\underline{u}] \times T))$.
  For each $[u] \in [V] \setminus \{[\underline{u}]\}$ with $\underline{t}_{[v]}$ computed for all $([v],[u]) \in [A]$, compute

      $\underline{t}_{[u]} := \max \left\{ 1 + \overline{t}(V_T^\lambda \cap ([u] \times T)) - \underline{\delta}_{[u]}, \max_{([v],[u]) \in [A]} \{ \underline{t}_{[v]} + \underline{d}([v],[u]) \} \right\}.$   (22)

2 Construct time shifts and upper time limits (in reverse topological order):
  Set final values for $[\overline{u}]$: $T^{[\overline{u}]} := \{\underline{t}_{[\overline{u}]}\}$, $\overline{t}^{[\overline{u}]} := \overline{t}(V^{[\overline{u}]}_{\max T^{[\overline{u}]}})$.
  For each $[u] \in [V] \setminus \{[\overline{u}]\}$ with $T^{[v]}$, $\overline{t}^{[v]}$ computed for all $([u],[v]) \in [A]$:
  2.1 Initialize $T^{[u]} \leftarrow \emptyset$, $\overline{t}^{[u]} \leftarrow \overline{t}(V_T^\lambda \cap ([u] \times T))$.
  2.2 For each $([u],[v]) \in [A]$:
      2.2.1 If $[v]$ contains no node $v'$ with $(v',v') \in A$, update
            $T^{[u]} \leftarrow T^{[u]} \cup \{t - \underline{d}([u],[v]) : t \in T^{[v]}\}$,
            $\overline{t}^{[u]} \leftarrow \max\{\overline{t}^{[u]}, \overline{t}^{[v]} - \underline{d}([u],[v])\}$,
            and continue with the next arc $([u],[v])$ in 2.2.
      2.2.2 Compute a common time shift connectable to all $v' \in [v]$ with $(v',v') \in A$:

            $\underline{\tau}_{([u],[v])} := \max \left\{ \underline{t}_{[u]}, \overline{t}(V_T^\lambda \cap ([v] \times T)) - \min_{\substack{v' \in [v],\\ (v',v') \in A}} \underline{d}([u],v') - \underline{\delta}_{[u]} \right\},$   (23)

            $\overline{d}^{[v]}_{[u]} := \max \left\{ \overline{d}([u],[v]), \overline{\delta}_{[u]} + \max_{\substack{v' \in [v],\\ (v',v') \in A}} \max_{u' \in [u]} \underline{d}(u',v') \right\},$   (24)

            $t_{([u],[v])} := \underline{\tau}_{([u],[v])} + \overline{d}^{[v]}_{[u]},$   (25)

      2.2.3 Ensure a matching time shift for each $t \in T^{[v]}$ by the update
            $T^{[u]} \leftarrow T^{[u]} \cup \{\underline{\tau}_{([u],[v])} : t_{([u],[v])} \leq t \in T^{[v]}\} \cup \{t - \underline{d}([u],[v]) \colon t_{([u],[v])} > t \in T^{[v]}\}$.
      2.2.4 If $\max T^{[v]} \geq t_{([u],[v])}$ and $[v]$ contains for each $([v],[w]) \in [A]$ a $v_{[w]} \in [v]$
            with $(v_{[w]}, v_{[w]}) \in A$ so that $\overline{d}(v_{[w]}, w') \geq \overline{d}([v], w')$ for all $w' \in [w]$, then
            2.2.4.1 determine an upper bound on the time index of bad nodes:

                    $\Delta_{([u],[v])} := \max_{([v],[w]) \in [A]} \{ \overline{d}(v_{[w]}, [w]) - \underline{d}([v],[w]) \} - \underline{d}([u],[v]),$   (26)

                    $\overline{\tau}^{([u],[v])} := \max \left\{ \overline{t}(V^{[u]}_{\underline{\tau}_{([u],[v])}}), \overline{t}^{\underline{\tau}_{([u],[v])}}_{([u],[v])} + \Delta_{([u],[v])}, \overline{t}(V^{[u]}_{\max T^{[u]}}) \right\}.$   (27)
```

```
        2.2.4.2 If $\overline{\tau}^{([u],[v])} < \overline{t}^{[v]} - \underline{d}([u],[v])$ then update
                    $\overline{t}^{[u]} \leftarrow \max\{\overline{t}^{[u]}, \overline{\tau}^{([u],[v])}\}$,
                and continue with the next arc $([u],[v])$ in 2.2.
    2.2.5 Update  $\overline{t}^{[u]} \leftarrow \max\{\overline{t}^{[u]}, \overline{t}^{[v]} - \underline{d}([u],[v])\}$.
3 $G_T^C := \mathrm{cl}(\bigcup_{[u]\in[V]}[u] \times \overline{t}^{[u]}, \emptyset)$.
```

Remark 32 1. The more involved quantities of this algorithm do not depend on G_T^λ and can be precomputed (*e. g.*, $\overline{d}_{[u]}^{[v]}$, $\Delta_{([u],[v])}$ and the validity of the condition of step 2.2.4 are constants), or can be computed much simpler than the corresponding definitions suggest (e.g., $\overline{t}(V_{\underline{\tau}_{([u],[v])}}^{[u]}) = \underline{\tau}_{([u],[v])} + \overline{\delta}_{[u]}$ and for the given choices, $\overline{t}_{([u],[v])}^{\underline{\tau}([u],[v])} = \underline{\tau}_{([u],[v])} + \max_{v'\in[v],(v',v')\in A} \max_{(u',\delta_{u'})\in V^{[u]}}(\delta_{u'} + \underline{d}(u',v'))$ by (29) below) that we avoided so as to highlight the connection to the original objects. In fact, the algorithm turns out to be very efficiently implementable (see Observation 33 below).

2. In many situations the $T^{[u_i]}$ and $\overline{t}^{[u_i]}$ computed by Algorithm 31 will not yield the smallest possible graph G_T^C. Indeed, several conditions could be stated or computed in more sophisticated manner. The current choices are meant to keep the presentation somewhat manageable without significant loss in performance for most practical situations.

Observation 33 *The update of the $T^{[u]}$ and $\overline{t}^{[u]}$ for a new G_T^λ requires*

$$O\left(|[A]| \cdot \max_{[u]\in[U]} |T[u]|\right)$$

steps and only depends on the size and structure of the clone graph $[G]$. In particular, if $[G]$ is a path, $\max_{[u]\in[U]}|T[u]| = 1$ and the running time is linear in the number of clone nodes.

Proof. In view of Remark 32, the most expensive step of the algorithm is 2.2.3, which runs in $O(\max_{[u]\in[U]}|T^{[u]}|)$ and is executed $O(|[A]|)$ times. If $[G]$ is a path then $|T^{[\overline{u}]}| = 1$ by step 2 and because $\forall\,[u] \in [V] \setminus \{\overline{u}\}$: $|\{([u],[v]) \in [A]\}| = 1$ exactly one time step is added to the set $T^{[u]}$ either in step 2.2.1 or 2.2.3. □

Theorem 34 *For any G_T^λ and interception sets $V^{[u]}$, $[u] \in [V]$, Algorithm 31 computes a valid subnetwork $G_T^C = \mathrm{cl}(\bigcup_{[u]\in[V]}[u] \times \overline{t}^{[u]}, \emptyset)$ whenever (C) holds.*

Proof. We first show that $\mathcal{T} = \{T^{[u]}\}_{[u]\in[V]}$ is admissible. In order to see for every $[u] \in [V]$ and $t \in T^{[u]}$ that $V_t^{[u]} \cap V_T^\lambda = \emptyset$, it suffices to prove $\min T^{[u]} \geq \underline{t}_{[u]}$, because $\underline{t}_{[u]} \geq 1 + \overline{t}(V_T^\lambda \cap ([u] \times T)) - \underline{\delta}_{[u]}$ by (22). For $[\overline{u}]$ this holds because $T^{[\overline{u}]} = \{\underline{t}_{[\overline{u}]}\}$. For all others, $\min T^{[u]} \geq \underline{t}_{[u]}$ follows either by induction in reverse topological order via the second term in the max of (22) or it is required explicitly in (23). Next we show by induction in reverse topological order of the nodes $[u]$ that for any $([u],[v]) \in [A]$

each $t_{[v]} \in T^{[v]}$ has a matching $t_{[u]} \in T^{[u]}$. For $[\overline{u}]$ there is nothing to show, so let $[u] \in [V]$, $[u] \neq [\overline{u}]$. When $([u],[v])$ is treated in step 2.2, the computation of $T^{[v]}$ is already complete. If $[v]$ contains no v' with $(v',v') \in A$, then step 2.2.1 puts a matching $t_{[u]}$ into $T^{[u]}$ for each $t_{[v]} \in T^{[v]}$ by Observation 20. If there is a $v' \in [v]$ with $(v',v') \in A$, step 2.2.3 does the same for all $t_{[v]} \in T^{[v]}$ with $t_{[v]} < t_{([u],[v])}$, so it remains to show that $\underline{\tau}_{([u],[v])}$ matches all $t_{[v]} \in T^{[v]}$ with $t_{[v]} \geq t_{([u],[v])}$. For this, observe that by (23) and (16) as well as (24) and (17)

$$\underline{\tau}_{([u],[v])} \geq \max_{\substack{v' \in [v],\\ (v',v') \in A}} \underline{\tau}_{[u]}^{v'} \quad \text{and} \quad \overline{d}_{[u]}^{[v]} = \max_{\substack{v' \in [v],\\ (v',v') \in A}} \overline{d}_{[u]}^{v'}.$$

Thus, the condition $t_{[v]} \geq t_{([u],[v])} = \underline{\tau}_{([u],[v])} + \overline{d}_{[u]}^{[v]}$ implies

$$\max_{\substack{v' \in [v],\\ (v',v') \in A}} \underline{\tau}_{[u]}^{v'} \leq \underline{\tau}_{([u],[v])} \leq t_{[v]} - \max_{\substack{v' \in [v],\\ (v',v') \in A}} \overline{d}_{[u]}^{v'}.$$

Therefore Proposition 29 proves that $t_{[u]} = \underline{\tau}_{([u],[v])}$ and $t_{[v]}$ match in this case. This establishes that $\mathcal{T}$ constructed by Algorithm 31 is admissible.

It remains to show that $\overline{t}^{[u]} \geq \overline{t}((V_T^{\mathcal{T}} \cup B^{\mathcal{T}}) \cap ([u] \times T))$ holds for all $[u] \in [V]$, because then G_T^C contains all G_T^λ-irreducible paths and is therefore a valid subnetwork given (C) by Corollary 10. The claim holds for $[\overline{u}]$ by the first line of step 2 and Observation 26. Once more using induction in reverse topological order let $[u] \in [V]$, $[u] \neq [\overline{u}]$, and suppose that $\overline{t}^{[v]} \geq \overline{t}((V_T^{\mathcal{T}} \cup B^{\mathcal{T}}) \cap ([v] \times T))$ holds for all $([u],[v]) \in [A]$. By Observation 28, for each $([u],[v]) \in [A]$ any node $(u', t_{u'})$ with $u' \in [u]$ lying on a G_T^λ-irreducible path passing a node of $[v]$ must satisfy $t_{u'} \leq \overline{t}^{[v]} - \underline{d}([u],[v])$. This proves the bound for all bad nodes lying on an irreducible path leading on to nodes $[v]$ that are treated in steps 2.2.1 or 2.2.5. If, for some $[v]$, $\overline{t}^{[u]}$ is updated within step 2.2.4.2 then $\underline{\tau}_{([u],[v])} \in T^{[u]}$ by step 2.2.3. Now observe that in step 2.2.4.2 $[v]$ satisfies the requirements of Proposition 30 and $t_{[u]} = \underline{\tau}_{([u],[v])}$ guarantees that for all $v' \in [v]$ with $(v',v') \in A$ and all $(u', t_{u'}) \in V^{[u]}_{\underline{\tau}_{([u],[v])}}$ there is an arc $((u', t_{u'}), (v', t_{v'})) \in A_T$ with $t_{v'} \geq \overline{t}(V_T^\lambda \cap ([v] \times T))$. Indeed, for the arc $((u', t_{u'}), (v', t_{v'})) \in A_T$ with $t_{v'} - t_{u'} = \underline{d}(u', v')$ we obtain

$$t_{v'} \geq t_{u'} + \underline{d}([u], v') \overset{(u',t_{u'}) \in V^{[u]}_{\underline{\tau}_{([u],[v])}}}{\geq} \underline{\tau}_{([u],[v])} + \underline{\delta}_{[u]} + \underline{d}([u], v') \overset{(23)}{\geq} \overline{t}(V_T^\lambda \cap ([v] \times T)).$$

Because the definition (27) of $\overline{\tau}^{([u],[v])}$ just matches or exceeds the right hand side of (19), Proposition 30 asserts the validity of the bound $\overline{\tau}^{([u],[v])}$ for irreducible paths using an edge from $[u]$ to $[v]$. Finally, whenever $\overline{t}^{[u]}$ is updated for any $[v]$, the corresponding update ensures, either via Observation 28 or explicitly in (27), that $\overline{t}^{[u]} \geq \overline{t}(V^{[u]}_{\max T^{[u]}})$ for each current $T^{[u]}$. Thus, $\overline{t}^{[u]} \geq \overline{t}((V_T^{\mathcal{T}} \cup B_T^{\mathcal{T}}) \cap ([u] \times T)$ holds for all $[u] \in [V]$, completing the proof. □

In the case of general routing graphs it is difficult to obtain reasonable bounds on the number of time steps included on top of G_T^λ, because a clone node that is not needed at all within G_T^λ may be required to be included with high time indices in $G_T^{\mathfrak{T}}$, because it has a potential predecessor having high time indices on G_T^λ. Likewise, the necessity to include fastest paths along all, in particular also along the slowest clone route, may already induce a high lower bound $\underline{t}_{[u]}$ far off from $\overline{t}(V_T^\lambda \cap ([u] \times T))$.

Useful bounds seem only to be obtainable when $[G]$ is restricted to a path, which is actually the case in our practical application. A specialized algorithm for path graphs $[G]$ would exploit that all time shift sets $T^{[u]}$ hold a single time value $t_{[u]}$, but we refrain from this here, because we would like to highlight in what way the given Algorithm 31 improves on the construction of Theorem 11 already in the case of $[G]$ being a clone path. In general, closeness to G_T^λ requires the availability of clone nodes with waiting nodes. We first show for every clone arc $([u],[v])$ that has a $v' \in [v]$ with $(v',v') \in A$, that the time shifts of $T^{[u]}$ matching a time shift of $T^{[v]}$ are within a constant of $\overline{t}(V_T^\lambda \cap ([v] \times T))$.

Observation 35 *Suppose that $[G] = [\underline{u} = u_1][u_2]\ldots[\overline{u} = u_k]$ is a path and that Algorithm 31 executes step 2.2.3 for an arc $([u],[v]) \in [A]$. Then*

$$\max T^{[u]} \leq 1 + \overline{t}(V_T^\lambda \cap ([v] \times T)) + \overline{d}([u],[v]) - 2\underline{d}([u],[v]) + \overline{\delta}_{[u]} - \underline{\delta}$$

where $\underline{\delta} = \min\{\underline{\delta}_{[u_i]} : i = 1,\ldots,k\}$ with $\underline{\delta}_{[u]}$ and $\overline{\delta}_{[u]}$ defined in Proposition 29.

Proof. By induction on step 1 of Algorithm 31 there holds for $i = 2,\ldots,k$

$$\underline{t}_{[u_i]} \leq 1 + \overline{t}(V_T^\lambda \cap ([u_i] \times T)) - \underline{\delta},$$

because $\overline{t}(V_T^\lambda \cap ([u_i] \times T)) \geq \overline{t}(V_T^\lambda \cap ([u_{i-1}] \times T)) + \underline{d}([u_{i-1}],[u_i])$ due to $[G]$ being a path. Thus, by (23) and $\underline{\delta} \leq 0$, the value $\underline{\tau}_{([u],[v])}$ may be bounded by

$$\underline{\tau}_{([u],[v])} \leq 1 + \overline{t}(V_T^\lambda \cap ([v] \times T)) - \underline{\delta} - \underline{d}([u],[v]), \tag{28}$$

and from (24) we obtain $\overline{d}_{[u]}^{[v]} \leq \overline{d}([u],[v]) + \overline{\delta}_{[u]}$. Therefore, in step 2.2.3 a $t_{[v]} \in T^{[v]}$ with $t_{[v]} < t_{([u],[v])} = \underline{\tau}_{([u],[v])} + \overline{d}_{[u]}^{[v]}$ gives rise to a time shift

$$t_{[u]} = t_{[v]} - \underline{d}([u],[v]) \leq 1 + \overline{t}(V_T^\lambda \cap ([v] \times T)) - \underline{\delta} - 2\underline{d}([u],[v]) + \overline{d}([u],[v]) + \overline{\delta}_{[u]}.$$

Because this bound is greater than the upper bound on $\underline{\tau}_{([u],[v])}$, this completes the proof. □

In the same way, a bound can be derived for the value $\overline{t}^{[u]}$ whenever Algorithm 31 reaches step 2.2.4.1.

Observation 36 *Suppose that $[G] = [\underline{u} = [u_1][u_2]\ldots[\overline{u} = u_k]$ is a path and that Algorithm 31 executes step 2.2.4.1 for an arc $([u_i],[u_{i+1}]) \in [A]$. Then*

$$\overline{t}^{[u_i]} \leq 1 + \overline{t}(V_T^\lambda \cap ([u_{i+1}] \times T)) + \sum_{j=i}^{i+1}(\overline{d}([u_j],[u_{j+1}]) - \underline{d}([u_j],[u_{j+1}])) \\ - \underline{d}([u_i],[u_{i+1}]) + 2\overline{\delta}_{[u_i]} - \underline{\delta}$$

where $\underline{\delta} = \min\{\underline{\delta}_{[u_i]} : i = 1,\ldots,k\}$ with $\underline{\delta}_{[u]}$ and $\overline{\delta}_{[u]}$ defined in Proposition 29.

Proof. By (26) and $[G]$ being a path we have

$$\Delta_{([u_i],[u_{i+1}])} \leq \overline{d}([u_{i+1}],[u_{i+2}]) - \underline{d}([u_{i+1}],[u_{i+2}]) - \underline{d}([u_i],[u_{i+1}]).$$

In order to bound the value $\overline{t}^{\underline{\tau}_{([u_i],[u_{i+1}])}}_{([u_i],[u_{i+1}])}$ defined in (20) let $(u',t_{u'}) \in V^{[u_i]}_{\underline{\tau}_{([u_i],[u_{i+1}])}}$, and $((u',t_{u'}),(v',t_{v'})) \in A_T$ with $(v',v') \in A$, then

$$\begin{aligned}
t_{v'} &\geq t_{u'} + \underline{d}([u_i],v') \\
&\geq t_{u'} + \min_{\substack{v'' \in [u_{i+1}],\\ (v'',v'') \in A}} \underline{d}([u_i],v'') \\
&\geq \underline{\tau}_{([u_i],[u_{i+1}])} + \underline{\delta}_{[u_i]} + \min_{\substack{v'' \in [u_{i+1}],\\ (v'',v'') \in A}} \underline{d}([u_i],v'') \\
&\overset{(23)}{\geq} \overline{t}(V_T^\lambda \cap ([u_{i+1}] \times T)).
\end{aligned}$$

Therefore, by (20),

$$\begin{aligned}
\overline{t}^{\underline{\tau}_{([u_i],[u_{i+1}])}}_{([u_i],[u_{i+1}])} &= \max_{\substack{v' \in [u_{i+1}],\\ (v',v') \in A}} \max_{(u',t_{u'}) \in V^{[u_i]}_{\underline{\tau}_{([u_i],[u_{i+1}])}}} (t'_u + \underline{d}(u',v')) \\
&= \max_{\substack{v' \in [u_{i+1}],\\ (v',v') \in A}} \max_{(u',\delta_{u'}) \in V^{[u_i]}} (\underline{\tau}_{([u_i],[u_{i+1}])} + \delta_{u'} + \underline{d}(u',v')) \qquad (29) \\
&\leq \underline{\tau}_{([u_i],[u_{i+1}])} + \overline{\delta}_{[u_i]} + \max_{\substack{v' \in [u_{i+1}],\\ (v',v') \in A}} \max_{(u',\delta_{u'}) \in V^{[u_i]}} \underline{d}(u',v') \\
&\leq \underline{\tau}_{([u_i],[u_{i+1}])} + \overline{\delta}_{[u_i]} + \overline{d}([u_i],[u_{i+1}]) \\
&\overset{(28)}{\leq} 1 + \overline{t}(V_T^\lambda \cap ([u_{i+1}] \times T)) - \underline{\delta} - \underline{d}([u_i],[u_{i+1}]) + \overline{\delta}_{[u_i]} + \overline{d}([u_i],[u_{i+1}]).
\end{aligned}$$

Now observe that $\overline{t}^{\underline{\tau}_{([u_i],[u_{i+1}])}}_{([u_i],[u_{i+1}])} + \Delta_{([u_i],[u_{i+1}])} \leq \overline{\tau}$ where

$$\overline{\tau} = 1 + \overline{t}(V_T^\lambda \cap ([u_{i+1}] \times T)) + \sum_{j=i}^{i+1} (\overline{d}([u_j],[u_{j+1}]) - \underline{d}([u_j],[u_{j+1}])) - \underline{d}([u_i],[u_{i+1}]) + 2\overline{\delta}_{[u_i]} - \underline{\delta}.$$

Furthermore, by (28),

$$\overline{t}(V^{[u_i]}_{\underline{\tau}_{([u_i],[u_{i+1}])}}) = \underline{\tau}_{([u_i],[u_{i+1}])} + \overline{\delta}_{[u_i]} \leq 1 + \overline{t}(V_T^\lambda \cap ([u_{i+1}] \times T)) - \underline{\delta} - \underline{d}([u_i],[u_{i+1}]) + \overline{\delta}_{[u_i]} \leq \overline{\tau}.$$

Finally, because of Observation 35,

$$\begin{aligned}
\overline{t}(V^{[u_i]}_{\max T^{[u_i]}}) &= \max T^{[u_i]} + \overline{\delta}_{[u_i]} \\
&\leq 1 + \overline{t}(V_T^\lambda \cap ([u_{i+1}] \times T)) + \overline{d}([u_i],[u_{i+1}]) - 2\underline{d}([u_i],[u_{i+1}]) + 2\overline{\delta}_{[u_i]} - \underline{\delta} \\
&\leq \overline{\tau}.
\end{aligned}$$

Comparing this to (27) and step 2.2.4.2 completes the proof. □

Thus, if waiting nodes follow in reasonably close succession and these waiting nodes do indeed require more time to proceed to the next clone node than the other nodes of the clone, then the generated subnetwork will exceed G_T^λ only within a reasonably small time offset.

3.3 Dynamic Generation of Train Graphs

The train-graphs of the model described in Section 2.2 fit exactly in the framework developed in the previous section. Let $G^r = (V^r, E^r)$ be the basic graph of a train $r \in R$. The clone sets of G^r are formed by the run and wait nodes of a certain station, *i. e.*, for $(u, b) \in V^r$ the clone set is defined as $[(u, b)] := \{(u, b') \in V^r : b' \in \{run, stop\}\}$. It is easy to check that (K1)–(K3) are fulfilled by these clone sets. Furthermore the cost-function satisfies property (C).

Observation 37 *The time-expanded train-graph $G_T^r = (V_T^r, A_T^r)$ and the cost function $c_0(e) := w_e^r$, $e \in A_T^r$, of (6) satisfy (C).*

Proof. Let

$$P := (u_1, t_1) \dots (u_1, t_1')(u_2, t_2) \dots (u_k, t_k')$$

and

$$Q := (v_1, s_1) \dots (v_1, s_1')(v_2, s_2) \dots (v_k, s_k')$$

be two paths with $P \leq_T Q$. By definition of w_e^r the cost of P is

$$c_0(P) = \sum_{i=1}^{k-1} w^r_{((u_i,t_i'),(u_{i+1},t_{i+1}))}$$

and similary

$$c_0(Q) = \sum_{i=1}^{k-1} w^r_{((v_i,s_i'),(v_{i+1},s_{i+1}))}.$$

Because $P \leq_T Q$ there holds $t_i \leq s_i$ for $i \in \{2, \dots, k\}$ and by definition of the clone sets we have

$$\forall\, i \in \{1, \dots, k\} \colon \exists\, \hat{u}_i \in U(r) \colon u_i, v_i \in \{\hat{u}_i\} \times \{run, stop\}.$$

Again by definition of w_e^r it follows

$$\begin{aligned} w^r_{((u_i,t_i'),(u_{i+1},t_{i+1}))} &= \alpha^{m(r)} \cdot l_e \cdot \sum_{\hat{t}=t^r_{\min}(\hat{u}_{i+1})}^{t_{i+1}} \hat{t} \\ &\leq \alpha^{m(r)} \cdot l_e \cdot \sum_{\hat{t}=t^r_{\min}(\hat{u}_{i+1})}^{s_{i+1}} \hat{t} \\ &= w^r_{((v_i,s_i'),(v_{s+1},s_{i+1}))}, \end{aligned}$$

which in turn implies

$$c_0(P) \leq c_0(Q). \qquad \square$$

Putting all together, the framework of Section 3.2 can be applied directly. Note that for our practical instances the assumptions of Observation 17 hold, so the structure of the independent node sets can be given explicitly. Furthermore, $[G^r]$ is indeed a path with most clone nodes containing waiting nodes that have slower connections than run nodes. Thus, the bounds of Observation 36 hold.

4 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 approach 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 recompute the relaxation with partially fixed train paths. Suppose that for all trains all decisions have been fixed up to time step $t_s \in \mathbb{N}_0$. From this time step on the relaxation can be expected to be a good indicator for integer solutions within a maximal *rounding horizon* interval $H^r \in \mathbb{N}$ following t_s. Therefore, similar to a trust region approach, each train is fixed at most up to the first node having time index at least $t_s + H^r$. In order to provide some information beyond the rounding horizon, the model also includes some further time-steps $H^a \in \mathbb{N}$, the *look-ahead horizon.* Up to time step $t_s + H^r + H^a$ all coupling constraints will be separated, beyond $t_s + H^r + H^a$ all trains are allowed to use their local fastest path regardless of feasibility. Note that dynamic graph generation may possibly use fewer or also add some further time-steps in the model of each train depending on the structure of the actual solution. Because the number of time-steps by which the horizon is moved is not fixed a priori (only bounded from above by H^r), we call this approach *dynamic rolling horizon.*

In each horizon iteration, we used the following general framework for our rounding heuristics.

First we identify some conflicts in the relaxed solution. Given a train $r \in R$ and an infrastructure arc $a = (u, v) \in A^I$ on its route, let $\bar{t}(r, a) = \sum_{e=(((u,b),t),((v,b'),t')) \in A_T^r} x_e \cdot t$ denote the average flow time of train r on arc a in the relaxation. 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.*, $|\bar{t}(r_2, a) - \bar{t}(r_1, a)| \leq C^H(r_1, r_2, a)$ where $C^H(r_1, r_2, a)$ is some constant that usually depends on the headway-times between r_1, r_2 on a. $\mathcal{C} = \{C = (r_1, r_2, a, t) \colon r_1, r_2$ are in conflict on a at time $t = \min\{\bar{t}(r_2, a), \bar{t}(r_1, a)\}\}$ 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 and including 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, *i. e.*, the train will only be routed further once the next relaxation has been computed and any conflicts with this train after this arc simply block the progress of the corresponding conflicting trains. The following is a rough outline of this algorithm.

1. Initialize horizon start-time $t_s = 0$, choose $H^r, H^a \in \mathbb{N}$, simulation end-time t_{end}.
2. Create the model for time-steps $\{t_s, \dots, t_s + H^r + H^a\}$.
3. Solve the relaxation, determine all conflicts $\mathcal{C}$.
4. 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 up to and including 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 4,
5. Release all fixed arcs that start beyond time-step $t_s + H^r$,
6. Set $t_s := \max\{t' \in \mathbb{N}_0 \colon \text{ all trains are fixed up to } t'\}$.
7. Unless $t_s \geq t_{\text{end}}$, goto 2.

Note that if, in some major iteration, the first node without fixed exit arc of a train $r \in R$ is not the first station of that train, the arcs $((\sigma^r, t), (\sigma^r, t+1)) \in A^r_T$ are assigned high costs, because in this situation we want the train to start at its first possible time-step and waiting at its first station is only allowed using ordinary wait arcs (in contrast to the very first station of a train where it is allowed to wait arbitrarily without causing any conflicts). If some train has to use one of those arcs in order to have a feasible path in its network, this means it is impossible to find a conflict free solution (*i. e.*, the train is in a dead-lock situation) and the algorithm returns solutions which contain conflicts of that kind. We will later interpret these conflicts as (infeasible) waiting at the first unfixed station.

5 Load Balancing Functions

A main flaw of approximated solutions obtained via Lagrangian 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 $\beta^r_e > 0$ representing the amount of capacity measured in time steps that the usage of e would block. *E. g.*, if all headway-times of e would be two time-steps, then $\beta^r_e = 3$ 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

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

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

$$0 \in \operatorname{Argmin}\{f^{a,t}_b(z) \colon z \in \mathbb{R}_+\}, \tag{30}$$

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

$$\sum_{r \in R} \sum_{e=((u,b),(u',b)) \in \bigcup A^r} \sum_{t=t_0}^{t+\Delta} \beta^r_e x^r_{(((u,b),t),((u',b'),t'))} \leq z^{a,t}, a = (u,u') \in A^I, u \neq u'. \tag{31}$$

Note, condition (30) allows to formulate the constraint (31) 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 dynamic graph generation. In our implementation we choose piecewise linear convex functions $f^{a,t}_b$ that balance free minutes in a certain time-interval against train-minutes.

6 Numerical Results

For our tests we considered a real-world instance of the German railway network. It consists of about 10% of the whole network and comprises roughly Baden-Wuerttemberg. The available data is based on a real timetable for one day. It is not clear, however, whether the data admits feasible solutions within the requirements of the model. We extracted instances with a planning period of six hours each, starting every hour beginning with 3 o'clock in the morning up to 3 o'clock in the afternoon. The infrastructure network has 1776 nodes and 3852 edges. Table 1 shows the size of the infrastructure network and the number of trains to be scheduled. The weight factors α^m, $m \in M$, of (6) were set to

$$\alpha^m = \begin{cases} 30 & m \text{ is a long-distance passenger train-type,} \\ 20 & m \text{ is a short-distance passenger train-type,} \\ 1 & m \text{ is a freight train-type,} \end{cases}$$

which can, *e. g.*, be interpreted as one minute delay of a long-distance passenger train being as expensive as 30 minutes time-saving of a freight train. Therefore it encourages early runs of passenger trains which is a general rule for our practical instances.

start time	long-distance trains	short-distance trains	freight trains
3	58	1374	706
4	72	1730	728
5	87	1997	709
6	97	2239	669
7	107	2335	643
8	114	2334	614
9	111	2243	579
10	109	2149	595
11	108	2200	632
12	109	2375	646
13	115	2445	676
14	120	2499	671
15	115	2367	621

Table 1: Instances

The Lagrangian dual problem (7) has been solved using the ConicBundle [15] library implementing a bundle method to solve problems of type

$$\min_{\substack{\lambda_1 \geq 0 \\ \lambda_2 \text{ free}}} f(\lambda)$$

where $f(\lambda)$ is a convex function given by a first-order oracle, *i. e.*, for given λ the oracle returns $f(\lambda)$ and a subgradient $g(\lambda)$ of f at λ, which is appropriate for (7).

In order to demonstrate the benefits of dynamic graph generation, Table 2 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 only counts the arcs of the train-graphs, not the arcs of the configuration networks. This is because configuration networks grow very fast: A configuration network on an arc with five trains where each train has all 4 possible running behaviours,

maximal number of time-steps	static	dynamic
3600s	3654905	579152
7200s	9476644	830430
10800s	17573262	1195572

Table 2: Arc count for static and dynamic graphs

contains for a period of 60 time steps about

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

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. Also note that without dynamic graph generation we have to limit the number of time steps contained in the model to a reasonably large number. In contrast, with dynamic graph generation an infinite number of time steps can be handled, because all networks automatically grow only as far as needed. But even with dynamic graph generation no network will grow infinitely because there always exists the conflict-free solution where for a prespecified sequence of the trains each train uses the network exclusively right after the final arrival of the preceding train. Because the overall number of trains is always finite there is a maximal time step that no train will ever exceed. The important property of dynamic graph generation is that no a priori estimation of this bound is required anymore.

We tested our algorithm with different settings for the rounding horizon size and the look-ahead horizon size. All computations were performed on an Intel Core i7 CPU with 2.67 GHz and 12GB RAM without using parallelization. Dynamic graph generation has been used for all tests.

Unfortunately our tests showed that up to now the algorithm is not able to find a conflict-free timetable satisfying the time window restrictions (we do not even know if one exists). The main motivation for introducing load-balancing functions was to obtain relaxations that are easier to deal with in the rounding heuristics. Therefore we compared the different parameter settings w.r.t. the number of unresolved conflicts in the timetable as well as delays in passenger trains.

The diagrams in Fig. 16 show the solution qualities for all instances using a maximal rounding horizon of 600 seconds and an additional horizon look ahead size of 3000 seconds, *i. e.*, the model covers one hour. Diagrams (a) and (c) display the number of long-distance and short-distance trains, resp., that have a delay of more than 3 minutes at some stopping station when compared with the predefined timetable, and diagrams (b) and (d) give the average of the worst delay over all stations in seconds of those trains that have more than 3 minutes delay in at least one of the two cases (with or without load-distribution). The fifth diagram (e) shows the number of unresolved conflicts, *i. e.*, the number of violated capacity constraints in the final rounded solution. Finally diagram (f) displays the average number of seconds by which the planned freight trains reach their final stations earlier than the freight trains in the original data.

By diagram (a) the number of late long-distance trains is reduced by load balancing in most but not in all cases. At the same time the average delay of late long distance trains decreases in most cases. For short-distance trains the benefit is even better. The number of short-distance trains with delay is reduced (c) as well as the average delay of late short-distance trains in all but two cases (d). Note that on average only about 5% of the long-distance trains and 4% of the short distance trains have a delay at all. Diagram (e) indicates that in the majority of all cases the use of load-balancing functions also helps the rounding heuristic to find solutions with less unresolved conflicts leading to dead-lock situations in the dynamic horizon approach. Note that a count of 40 conflicts on average amounts to more than 98% of all trains having feasible timetables. The price of load balancing is a smaller saving in average freight train running times (f). Indeed, this should be expected, because load-balancing functions lead to less trains in the same time-interval on a certain track, therefore reserving more additional buffer space already in the relaxed solutions.

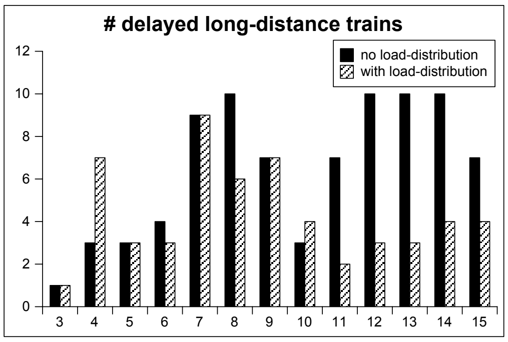
In order to identify the influence of different horizon-sizes we solved the instances for different values of the rounding and look-ahead horizons. The diagrams of Figure 17 correspond to those of Figure 16 but now display statistics over these values for the 13 instances for each setting of rounding horizon size and look-ahead horizon size. For each test-case, the small horizontal bar in the middle shows the median, 75% of all result values lie below the upper bound of the box and 75% lie above the lower bound of the box. Finally the small bars at the top and the bottom give the maximum and minimum value, respectively. All computations used load-balancing.

The results indicate that the choice of the rounding horizon has a significant impact on the solution quality. The main cause for this might be that the relaxation tends to use too much capacity compared to rounded integer solutions. In consequence, the development predicted by the relaxation differs significantly from realizable variants shortly after conflicting situations. Indeed, after most conflicts the relaxation is a rather bad adviser and may mislead rounding heuristics that try to follow it. A short rounding horizon is more likely to avoid this unfavorable situation at the cost of a possibly increased number of reevaluations of the relaxation.

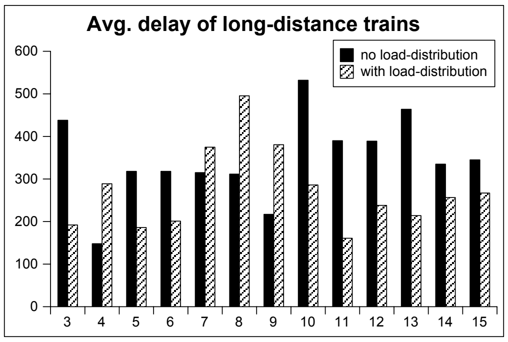
References

[1] R. Borndörfer and T. Schlechte. Models for railway track allocation. In C. Liebchen, R. K. Ahuja, and J. A. Mesa, editors, *ATMOS 2007*, Dagstuhl, Germany, 2007. IBFI, Schloss Dagstuhl, Germany.

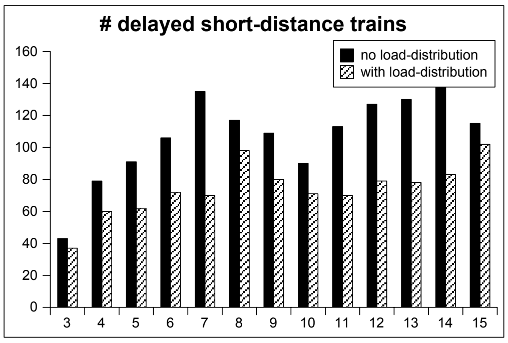
[2] R. Borndörfer and T. Schlechte. A suitable model for a bi-criteria optimization approach to railway track allocation. ZIB-Report 08-22, ZIB, 2008.



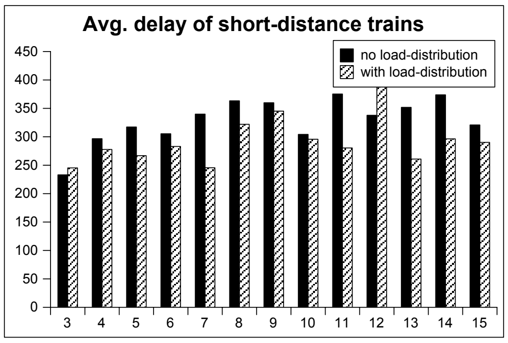
(a) Number of long-distance trains with more than 3 minutes delay



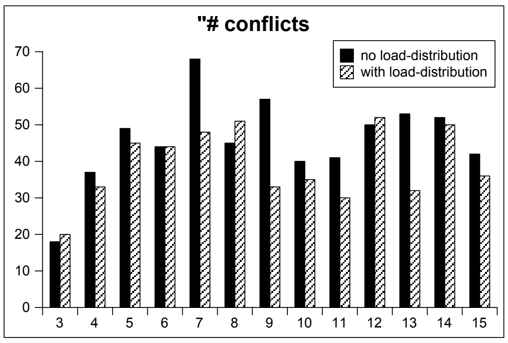
(b) Average delay in seconds of long-distance trains with more than 3 minutes delay



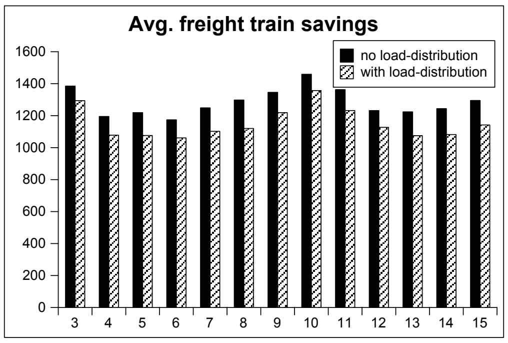
(c) Number of short-distance trains with more than 3 minutes delay



(d) Average delay in seconds of short-distance trains with more than 3 minutes delay

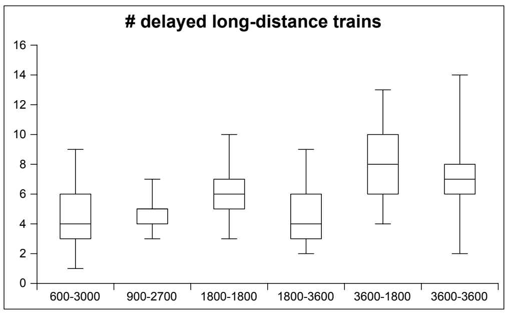


(e) Number of unresolved capacity conflicts

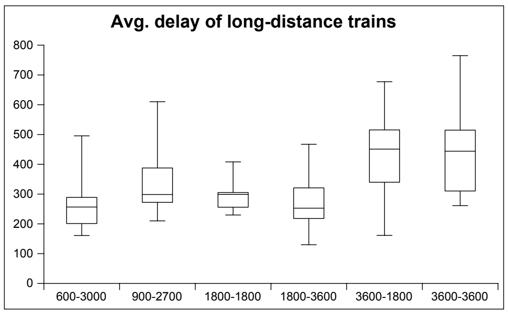


(f) Average saving of freight travel times in seconds compared with predefined timetable

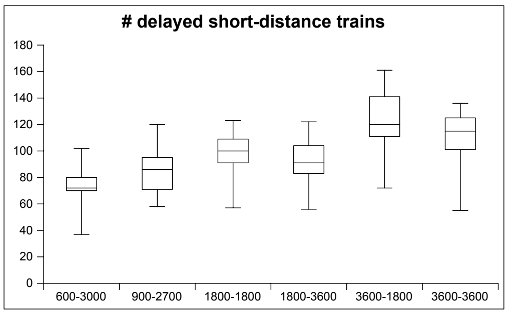
Figure 16: Comparison of solutions for runs with maximal 600 seconds of rounding horizon size and 3000 seconds of look-ahead horizon-size, with and without the use of load-balancing functions.



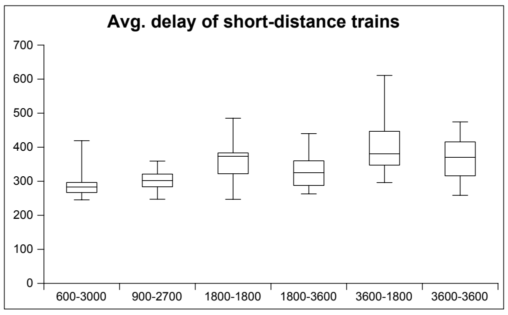
(a) Number of long-distance trains with more than 3 minutes delay



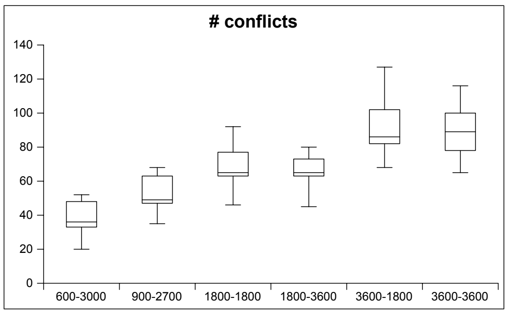
(b) Average delay in seconds of long-distance trains with more than 3 minutes delay



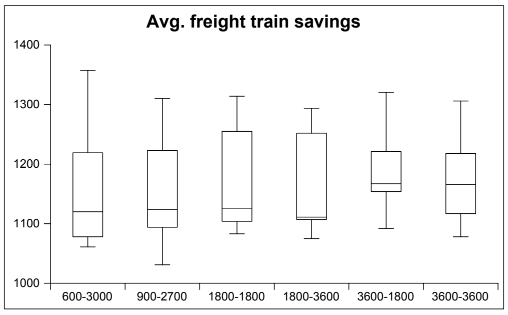
(c) Number of short-distance trains with more than 3 minutes delay



(d) Average delay in seconds of short-distance trains with more than 3 minutes delay



(e) Number of unresolved capacity conflicts



(f) Average saving of freight travel times in seconds compared with predefined timetable

Figure 17: Comparison of solutions for runs with load-balancing function and different maximal rounding horizon sizes and additional look-ahead sizes.

[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] V. Cacchiani, A. Caprara, and P. 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] G. Caimi, M. Fuchsberger, M. Laumanns, and K. Schüpbach. Periodic railway timetabling with event flexibility. *Netw.*, 57:3–18, January 2011.

[6] G. C. Caimi. *Algorithmic decision support for train scheduling in a large and highly utilised railway network.* PhD thesis, ETH Zürich, 2009.

[7] G. C. Caimi, M. Fuchsberger, M. Laumanns, K. Schüpbach, and S. Wörner. The periodic service intention as a conceptual framework for generating timetables with partial periodicity. In *ISROR Proceedings, 2009*, 2009.

[8] 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.

[9] A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Oper. Res.*, 50(5):851–861, 2002.

[10] A. Caprara, M. Monaci, P. Toth, and P. L. Guida. A lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Appl. Math.*, 154(5):738–753, 2006.

[11] D. Delling and G. Nannicini. Core routing on dynamic time-dependent road networks. *INFORMS Journal on Computing (to appear)*, 2011.

[12] S. Feltenmark and K. C. Kiwiel. Dual applications of proximal bundle methods, including Lagrangian relaxation of nonconvex problems. *SIAM J. Optim.*, 10(3):697–721, 2000.

[13] F. Fischer and C. Helmberg. Dynamic graph generation and dynamic rolling horizon techniques in large scale train timetabling. In T. Erlebach and M. Lübbecke, editors, *Proceedings of the 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 14 of *OpenAccess Series in Informatics (OASIcs)*, pages 45–60, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[14] F. Fischer, C. Helmberg, J. Janßen, and B. Krostitz. Towards solving very large scale train timetabling problems by lagrangian relaxation. In M. Fischetti and P. Widmayer, editors, *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany, 2008. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

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

[16] C. Liebchen. *Periodic Timetable Optimization in Public Transport.* PhD thesis, Technical University Berlin, 2006.

[17] G. Nannicini, P. Baptiste, G. Barbier, D. Krob, and L. Liberti. Fast paths in large-scale dynamic road networks. *Comput. Optim. Appl.*, 45(1):143–158, 2010.

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

[19] M. Schachtebeck. *Delay Management in Public Transportation: Capacities, Robustness, and Integration.* PhD thesis, Universität Göttingen, 2010.

[20] A. Schöbel. Integer programming approaches for solving the delay management problem. In *Algorithmic Methods for Railway Optimization*, number 4359 in Lecture Notes in Computer Science, pages 145–170. Springer, 2007.

[21] D. Schultes and P. 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.

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