Modeling, Analysis, Synthesis, and Simulation of Time-Optimal Train Traffic in Large Networks

Y. Bavafa-Toosi
2/71, Abuzar 11 St, Ahmadabad Ave, Mashhad 9176885464, Iran
Email: ybavafat@yahoo.com

C. Blendinger
DB Systems GmbH, Weilburger Str. 26, D-60326, Frankfurt, FRG
Email: Christoph.Blendinger@bahn.de

V. Mehrmann
Department of Mathematics, MA 4-5, Technical University of Berlin, D-10623, Berlin, FRG
Email: mehrmann@math.tu-berlin.de

A. Steinbrecher
Department of Mathematics, MA 4-5, Technical University of Berlin, D-10623, Berlin, FRG
Email: anst@math.tu-berlin.de

R. Unger
Department of Mathematics, Technical University of Chemnitz, D-09107, Chemnitz, FRG
Email: roman.unger@mathematik.tu-chemnitz.de

Abstract–From a system-theoretic standpoint, a constrained state-space model for train traffic in a large railway network is developed. The novelty of the work is the transformation or rather *reduction* of the directed graph of the network to some parallel lists. *Mathematization* of this sophisticated problem is thus circumvented. All the aspects of a real network (such as that of the German Rail) are completely captured by this model. Some degrees of freedom as well as some robustness can be injected into the operation of the system. The problem of time-optimal train traffic in large networks is then defined and solved using the maximum principle. The solution is obtained by reducing the boundary value problem arising from the time-optimality criterion to an initial value problem for an ordinary differential equation. A taxonomy of all possible switching points of the control actions is presented. The proposed approach is expected to result in faster-than-real-time simulation of time-optimal traffic in large networks and thus facilitation of real-time control of the network by dispatchers. This expectation is quantitatively justified by analysis of simulation results of some small parts of the German Rail Network.

Index Terms–Large networks, train traffic, time optimality, boundary value problem, initial value problem.

The original version of this paper was presented at the 10th IFAC/IFORS/IMACS/IFIP Symposium on Large Scale Systems: Theory and Applications, Osaka, Japan, 2004.

This work was supported in part by the Grant No. 03-MEM4B1 from The Ministry of Research and Technology of Germany.

I. Introduction

Train traffic control is an important issue in modern passenger and freight transportation systems. An enhanced transportation system in the sense of faster transportation, shorter delays, lesser consumed energy, etc. yields in large economical savings and higher customer satisfaction.

Railway networks are at the heart of public transportation systems in many countries. Part of a typical railway network is shown in Figure 1, where the connections A_1-A_2 and B_1-B_2 use the same physical tracks. In this network transportation between A_0 and A_3 can be done through $A_0-A_1-A_2-A_3$, and between B_0 and B_3 via $B_0-B_1-B_2-B_3$, both by different types of trains. From a system-theoretic point of view, the problem of stability (or analogously, safety) can be defined as the avoidance of any crash between trains coming to a join like A_1 and between consecutive trains. The complexity of the problem is better touched when noting that some trains are fast trains and do not stop at all stations, there may be construction sites, trains may break down and block part of the network and there exist schedule-based correspondences between some trains which have to be met. Currently it is also discussed to have variable maximal allowable velocities for every train in different parts of the network.

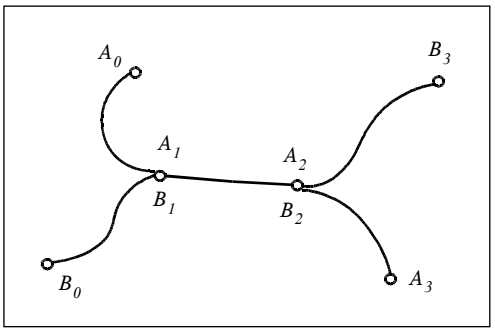

Fig. 1: Part of the Network

A clearly defined schedule alone is not the solution to this problem, since it may not be followed for a number of reasons as discussed above and, furthermore, it happens frequently that trains get out of the planned schedule. In this case, the operation of the network requires fast, real time decisions of the network operator or even automatic model based decision making, based on the current status of the network. To allow such model based decision making and real time control, an appropriate model of the existing network is required, which should be compatible with the existing theory and the real network at hand. Decision making would require a tool that allows the prediction of the dynamics of the whole network based on certain decision, such as having a train wait for a delayed train, or having one train pass another train to proceed faster. Such a model, furthermore, should allow real time computation for large realistic networks of thousands of kilometers of rail and hundreds of trains interacting in a complicated way with each other. This concerns in particular the inclusion of special trains that are added during the normal operation of the network, like cargo trains that do not operate on a regular basis. At this stage there are only very crude tools available, and even in a large network, like that of the German rail network, most of the decision making is still based on human observation of the status of the network. This situation motivates the development of a simulation tool that allows the prediction of the dynamic development of the whole network for a complete day of operation based on the actual status. The status is observed by a continuous stream and updating of train positions in the rail network and a comparison with the schedule based planned positions.

What is necessary in automated decision making or decision support is that at a current time point the actual position of all interacting trains is observed, based on this observation and the current operation plan a simulation is carried out and the evolvement of the network for the remainder of the operation period is predicted. Based on this prediction and a given decision and control algorithm, then the guidance system interacts with the network. Such a process requires real time simulation and

prediction algorithms which in turn require an appropriate mathematical model. The development of such a mathematical model that allows real time simulation and automated decision making is the topic of this paper. In deriving such a model for the real network of the German rail network extra difficulties that have to be dealt with, are decentralized control centers and, in particular, the fact that in the current situation the position information of trains is updated in a very fast way, but the velocity information for most trains is not continuously available. This makes the real time simulation very difficult and requires a completely new approach to the simulation of the network that will be discussed as well. Our new approach will be based on the fast computation of time-optimal controls.

A summary of the existing results and current trends in train traffic can be found in [1],[3]-[8],[11],[13],[16]-[20],[22]-[27] and the references therein. In the broad sense, the design objective in almost all of the existing results is one of the following: (O1) time optimality, (O2) energy optimality, (O3) mixed time and energy optimality, or (O4) model (i.e., trajectory or schedule) reference behavior. Moreover, most of the existing results have employed the maximum principle. In [1] based on the initial and terminal speed values, all optimal regimes and their possible sequences were classified. In [16],[17] the existence and parameterization of the optimal strategy for driving the train along a level track were considered. Numerical methods including genetic algorithms were suggested in a number of works, e.g., [6],[7],[17],[19],[23],[25]. The minimization of energy consumption was addressed e.g.
in like [6],[7],[13],[17],[19],[22]. In [19] the energy-optimal operation of a train on a variable grade profile subject to arbitrary speed restrictions was studies. A multi-objective trajectory (i.e., mixed time and energy) minimization problem was tackled in [26]. Through the goal-coordination method, the optimal schedule problem (i.e., model reference behavior) was handled in [8]. Fuzzy-logic and decentralized control ideas were employed in [18] and [20], respectively. A discrete-event model was also developed in [27].

This work presents and extends the results of [4],[5],[24] on (O1). We present a model for time-optimal train traffic in a schedule-based large network and explain how this model may be used for the purpose of fast simulation. As we have already mentioned above, a major reason for adopting (O1) is the missing velocity information for different trains and the fact that there are only asynchronous position measurements. This lack of information, nevertheless, is overcome by the assumption that the operation of each train follows a bang-bang control. This allows to compute the necessary velocity information in a fast and sufficiently accurate way.

The novelty of the advocated approach is the transformation or rather *reduction* of the network, which in fact is a directed graph, to as many parallel lists as the number of trains. These lists and a very simple interaction procedure between these lists allow a simulation in faster than real time, because much of the information can be pre-computed off-line and just looked up in tables, thus reducing significantly the complexity of the problem. In other words, unlike the existing literature, the derivation of a complicated mathematical model for this sophisticated problem is partially avoided.

The list of every train comprises its physical parameters, the specifications of its path and schedule, and the information which describes the interaction caused by that train on the other trains. Based on this representation, the system of ordinary differential equations (ODEs) describing the dynamics of all the trains is then solved, but real integration of the ODE is only carried out in the case of some interactions. To simulate the network online, in a round-robin fashion, using a fixed step-size, the lists of the trains which have interaction with some others are swept in parallel and by resolving all interactions between them their position and velocity are found versus time. The simulation of the rest of the network, i.e., trains with no interactions, is done in the same manner but beforehand and offline and the actual values are obtained from tables. As will be discussed, this is the advantage of using parallel lists instead of the graph of the network, and results not only in faster-than-real-time simulation of train traffic in large networks, but also in a quantitative analysis and estimate of the simulation time. None of the existing methods has achieved on-line simulation and this is the first work in this field.

The organization of this paper is as follows. In Section II, a constrained state-space model is developed for a network such as that of the German railway. In Section III, the problem of time-optimal train traffic in large networks is defined and solved. The solution is obtained by reducing the boundary

value problem arising from the time-optimality criterion to an initial value problem for an ODE, whereby all the static switching points of the control actions are computed offline. The computational methods for the aforementioned procedures are presented in Section IV, where a taxonomy of all possible switching points of the control actions is also given. In Section V, the general specifications of an actual network like that of the German railway are applied to the proposed model. It is shown that these specifications in fact lead to some simplifications under which the simulation is greatly accelerated. In Section VI it is explained how this model should be used for the purpose of simulation. Section VII is devoted to worked-out examples. The modeling procedure is illustrated in details and some simulation results are offered. The simulation results not only show the pragmatism of the advocated methodology for modeling, analysis, synthesis, and simulation of time-optimal train traffic in large networks, but also quantitatively exhibit the on-line simulation capability of the approach. These are discussed in Section VIII, where concluding remarks and future work are drawn.

II. MODELING

The model is obtained by transforming (reducing) the directed graph of the network to some parallel lists in the following four steps. The first step is as follows: it is assumed that every train has its sole individual path. In order to obtain faster-than-real-time simulation, the following simplifying assumption is made: every train is considered as a single mass point moving on a straight path. Defining $x(t)=\begin{bmatrix} s(t) & v(t) \end{bmatrix}^T$ where $s(t)$ and $v(t)$ denote the position and velocity of a train at time *t*, respectively, the dynamics of the train is governed by the following minimal state-space representation:

$$\dot{x}=\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x+\begin{bmatrix} 0 \\ 1/m^r \end{bmatrix} u \ , \tag{1}$$

where m^r denotes the mass of the train including the effect of its rotational parts, i.e., $m^r > m$, see [5],[21] for details. The total force acting on the system is given by $u=u_{uc}(s,v)+u_c(v)$. The uncontrollable part u_{uc} includes the air resistance, friction and gravitational effect. The controllable part u_c is applied by the driver (or the control system) and is constrained by the maximal braking force $u_{\min}(s(t),v(t))$ approximated by $u_{\min}(s(t),v(t))=ma$ where the deceleration *a* is a position-dependent constant and the maximal thrust approximated by $u_{\max}(v(t))=P_{\max}/v(t)$. In turn, $P_{\max}$ denotes the maximal power of the engine, and is constant over a velocity interval, see [5] and the references therein for more details.

The dynamics of every train is thus governed by (1) subject to the constrained control:

$$u=u_{uc}(s,v)+u_c(v) \ , \tag{2}$$

$$u_c(v)\in[u_{\min}(s(t),v(t)),u_{\max}(v(t))] \ , \tag{3}$$

as well as some other constraints, namely: (C1) the maximal allowable velocity in certain intervals of the line and the restriction that the train cannot move backwards; (C2) reaching some position in a certain time window; and (C3) leaving some position in a certain time window. In this paper the time-window constraints (C2) and (C3) will be omitted for the consideration of time optimality, because they model a more schedule-based driving strategy.

In the second step of modeling, the constraints (C1) are transformed into some mathematical constraints to be embedded into the above-derived model. For this, in the actual network the path of every train is divided into some adjacent intervals, in each a maximal speed being allowed[1]. Hence,

[1] The main factors that determine the beginning and end points of an interval as well as its maximal allowable velocity are the environment, type of the tracks, and trains. For instance, it is clear that near and in stations and in sharp bends some intervals must be properly placed whose maximal allowable velocities are low enough to guarantee safe operation

analogously, it is assumed that the path of every train is divided into some adjacent intervals as in Figure 2. The starting point of interval i is given by s^{i-1} which is the end point of interval $i-1 (i \geq 1)$, s^0 is the starting point and s^n the end point of the path. The maximal allowable velocity over the whole path is thus a piecewise continuous function with possible jumps at the interval boundaries. Therefore $v(s(t)) \in [0, v_{\max}(s(t))]$ in which $v_{\max}(s(t)) = v^i_{\max}(s(t))$ for $s \in [s^{i-1}, s^i]$, $i = 1, \ldots, n$, denotes the maximal allowable velocity in the *i*th interval.

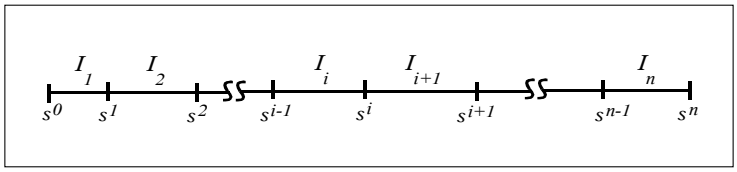

Fig. 2: Intervals

The third modeling step is the transformation of the operating system for safe traffic into some constraints. The safe-traffic operating system or the *safety concept* is as follows. All the pathways of the network are divided into some overlapping *safety blocks*. These safety blocks have some special characteristics, in particular: a) in each safety block at a given time there may be at most one train, and b) only at the end of a safety block a train is allowed to and may have to stop. Leaving and entering these safety blocks is detected by sensors. In order to stop a train at the end of a safety block a red light is signaled and the driver starts to brake at a *braking point* (to be exactly defined in Section IV) which is determined by a Pre-Signal, see Figure 3. Also, in order to let a train resume its journey after a stopping point a green light is signaled at a Main-Signal, see Figure 3.

Here, analogously, for every train the above partitioning is exactly considered over the supposed sole individual path; i.e., every train has its own exclusive safety blocks as in Figure 3. This figure illustrates the situation of two consecutive trains, one of the only two situations in which a crash may occur. In this figure, while the 1st Train is in the block *l*, the 2nd Train is not allowed to enter it which means that it must stop at the end of the block *j*, and only when the 1st Train has left the block *l*, the 2nd Train is allowed to leave the block *j*. These conditions are transformed to the following *simple blocking and unblocking conditions*: <*l*> `BLOCKS` <*j*> and <*l*> `UNBLOCKS` <*j*>.

The other situation for a possible crash is that of two trains coming to a join. In this case also a proper set of blocking and unblocking conditions will prevent the crash. In the actual network, priorities at a join – i.e., which train should pass first when some come to a join together – correspondences, and sequencing of trains are also treated by proper blocking and unblocking conditions at the right time by the dispatchers.

The preceding blocking/unblocking condition is the heart of the safety concept and is applied to the whole network. In this way safe operation in the whole network is always guaranteed regardless of the driving strategy. More precisely, no matter what speed the trains travel on, and more particularly, no matter the trains follow the bang-bang strategy of Section III or not, the safe operation in the whole network is always assured.

---

of trains and safety of the environment, see Example 7.1.

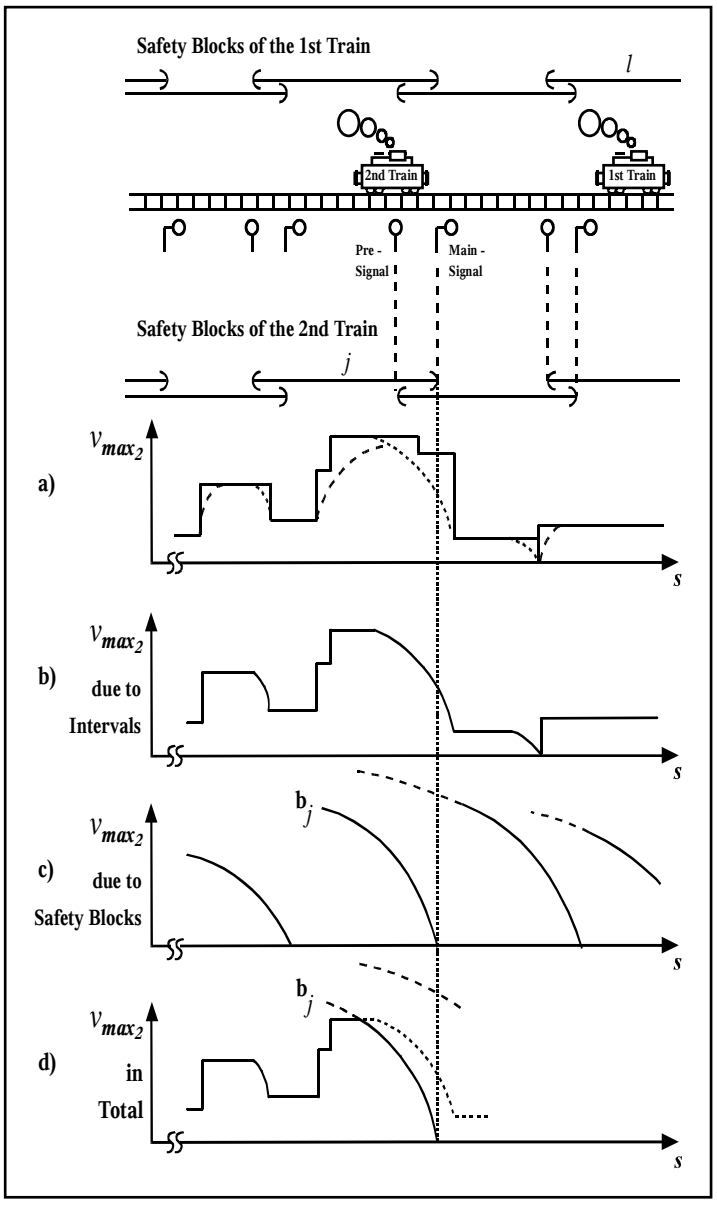

Fig. 3: Safety Concept

In the fourth and last step of modeling, the simple blocking and unblocking conditions are extended to some *extended blocking and unblocking conditions* which address all the explicit times, correspondences, priorities and sequencing in the network. More specifically, the following are some extended blocking/unblocking conditions:

<*a*> UNBLOCKED at SDT,
<*b*> UNBLOCKED at LDT,
<*c*> UNBLOCKED after MaxWT,
<*d*> UNBLOCKS <*e*> after MinWT,

in which SDT, LDT, MaxWT and MinWT stand for the Soonest Departure Time, Latest Departure Time, Maximum Waiting Time, and Minimum Waiting Time, respectively.

Starts of journeys are handled by the first of the above conditions along with the simple condition <*a*> INITIALLY BLOCKED, which together mean that the block *a* is initially blocked and then is unblocked at SDT. Thus, by a proper definition of the block *a* (i.e., the end of the block *a*

should be the starting point of a path) the start of a journey will be addressed. Analogously, correspondences, priorities and sequencing are captured by the first and last of the above rules along with some `INITIALLY BLOCKED` rules and some properly defined safety blocks. In the simulation software, entering and leaving these blocks are detected by some counters.

Remark 2.1: By way of `SDT` and `MinWT` all the time correspondences, priorities and sequencing of the trains are encapsulated in the extended blocking/unblocking conditions.

Remark 2.2: By the introduction of `LDT` and `MaxWT` some degrees of freedom (flexibility) and some robustness are injected into the operation of the whole network. The flexibility is due to the freedom in the starting time from the first and midway stations. The robustness is achieved because no train will wait unreasonably long (in theory, forever) for another train if it does not meet its scheduled correspondence/priority with that train. This is *somehow* analogous to the lower and upper bounds of the system performance in robust quantitative feedback theory (QFT) [15]. This robustification will be further interpreted in Remark 3.3.

Based on the above developments, the dynamics of a large railway network with n_z trains is given by the following constrained state-space model:

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_{n_z} \end{bmatrix} = \begin{bmatrix} A_1 & & 0 \\ & \ddots & \\ 0 & & A_{n_z} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n_z} \end{bmatrix} + \begin{bmatrix} B_1 u_1 \\ \vdots \\ B_{n_z} u_{n_z} \end{bmatrix}, \tag{4}$$

subject to the simple and extended blocking and unblocking conditions, where for $i=1,\ldots,n_z$:

$$x_i = \begin{bmatrix} s_i \\ v_i \end{bmatrix}, \tag{5}$$

$$A_i = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \tag{6}$$

$$B_i = \begin{bmatrix} 0 \\ 1/m_i^r \end{bmatrix}, \tag{7}$$

$$v_i(s(t)) \in [0, v_{i_{\max}}(s(t))], \tag{8}$$

$$u_i = u_{i_{uc}}(s, v) + u_{i_c}(v), \tag{9}$$

$$u_{i_c}(v) \in [u_{i_{\min}}(s(t), v(t)), u_{i_{\max}}(v(t))]. \tag{10}$$

The contribution of the above model is worth paraphrasing. This is a *decentralized* model, which has been obtained by transforming or, more exactly, reducing the graph of the network to some *parallel lists*. The *interactions* between the trains, which are not transparent in this decentralized model, have been encapsulated in the *simple and extended blocking and unblocking conditions*. As will be seen, it is this model that enables: i) faster-than-real-time simulation of the network, and ii) a quantitative analysis and estimate of the simulation time. In turn, these will enable online control and decision making in the whole network, as discussed in Section I.

It is noteworthy that the second feature – a quantitative measure of the simulation time – is very desirable. On the one hand, it quantitatively *proves* (see Section VIII) the on-line simulation capability of the advocated methodology and on the other hand, especially in the status quo that decision making is done manually, it guarantee a time for dispatchers to ponder and carefully consider the situation in order to make the right decision.

Some remarks are in order.

Remark 2.3: The model is generic and can be used to establish an operating system for new railway networks. More precisely, it can be used for analysis and synthesis of existing networks. To this end, the following points must be considered: (P1) Interval placement: This depends on the specific actual network, its location, neighborhood and physical restrictions which govern the maximal allowable velocities. (P2) Safety block placement: This depends on the specific actual network, but must in any case guarantee the safe operation of the whole network. Every station (or allowable midway stopping point) should be the end point of some safety block.

Remark 2.4: The total force acting on a train is given by (9). The controllable part u_c depends on the type of the train (i.e., the engine dynamics) and the dynamics of the network. It will be shortly explained, see the remainder of the paper, that it is either full thrust, full brake, or such that the maximal allowable velocities in the intervals are kept. The uncontrollable part u_{uc} is given by $u_{uc} = u_{af} + u_g$, where u_{af} accounts for the air resistance and friction, and u_g denotes the gravitational effect. The air resistance and friction depend on the type of the train and are *nonlinear* functions of its velocity; they should be found by experiment and identification. (Some details about this nonlinear function for different trains of the German rail can be found in [5],[24],[25] and the references therein.) The gravitational effect is given by $mg\rho$ where m is the mass of the train, g is the gravity acceleration and ρ is the slope of the track which depends on its position. In case of an acclivity ($\rho<0$) it opposes acceleration and in case of a declivity ($\rho>0$) it helps acceleration.

Remark 2.5: It is clear that there are some degrees of freedom in the interval and safety block placements. It is currently under investigation how to exploit this freedom to achieve some other design objectives like fuel efficiency, wear and tear (of trains and tracks) minimization, and economical implementation. Note that for economical implementation the number of safety blocks should be the minimum possible, since every safety block is equipped with two sensors at its beginning and end which communicate with some other sensors and also with the dispatching center. Moreover, the implementation of variable maximal allowable velocities for different trains and the effect of maximal allowable velocities on passengers comfort are being studied [25].

The modeling procedure and the above points and remarks will be illustrated in details in Example 7.1.

In the following section, the problem of time-optimal traffic in large networks is addressed. Prior to this, some explanations about Figure 3 are in order. This figure illustrates the situation of two consecutive trains. As explained before, every train is assumed to have its own exclusive safety blocks, as depicted on top of the figure. In (a), solid lines are the given maximal allowable velocities in intervals, previously denoted by $v_{\max}(s(t)) = v^i_{\max}(s(t))$ for $s \in [s^{i-1}, s^i]$, $i = 1, \ldots, n$. Dashed lines are the curves on which the 2nd train speeds up, and dotted lines are the curves on which the 2nd train slows down (see next section, Definitions 3.1 and 3.2). It is clear that in order to meet the maximal allowable velocity at the beginning of a proceeding interval, the 2nd train must slow down on the dotted lines, and thus (b) is the maximal allowable velocity due to (the braking curves, Definition 3.2, of the) intervals. On the other hand, as stated before, at the end of a safety block a train is allowed to and may have to stop, and thus (c) is the maximal allowable velocity due to safety blocks. Finally, (d) illustrates the maximal allowable velocity in total which is the minimum of the above three velocities (a),(b),(c), whenever a curve in (c) is to be realized. In this scenario b_j is realized due to the safety block concept explained in the third step of modeling. This will be completely clarified by Remarks 4.1 and 4.2, read through.

III. TIME–OPTIMAL TRAFFIC

Problem 3.1: The problem of time-optimal traffic (ride) for one train is to minimize $t^n - t^0$ subject to

(4)-(10) for $n_z = 1$, where t^0 and t^n denote the times at the beginning and the end of the journey, and thus $t^n - t^0$ is the total traveling time.

Based on the classical theory of optimal control [12],[14] (maximum principle), it is clear that the solution is given by a bang-bang control subject to the constraints. In other words, the control switches between its extremal values unless a constraint is reached in which case the control action is dictated by the constraint. The numerical computations to find the *switching points* – where the control switches between its extremal values or where the constraints are reached or left – are given in Section IV.

In this context, it is not relevant how long a train has to wait at a station or a midway point (due to a constraint or a break-down) or when it restarts after a stop; this merely leads to an offset in the total time.

Remark 3.1: To find the control action and the switching points for all the objectives (O1)-(O4), in fact a boundary value problem should be solved which can be done by finite element, finite difference or multiple shooting methods [10]. For the case of time-optimality, however, this task is greatly simplified: the static switching points are computed simply and offline by reducing the boundary value problem to an initial value problem for an ODE, see Section IV.

Problem 3.2: The problem of time-optimal train traffic for a large railway network defined by (4)-(10) is to minimize $t_i^n - t_i^0$ for $i = 1, \ldots, n_z$, i.e., all the individual traveling times, under all the constraints – i.e., maximal allowable velocities, safe interaction, correspondences, priorities and sequencing. As in the case of a time-optimal ride for one train, the control action of every train is given by a bang-bang control unless a constraint is reached in which case the control is dictated by the constraint.

Remark 3.2: Due to the freedom mentioned in the Remark 2.2, the traffic in the network is not uniquely specified in the sense that there is some freedom in the departure times from the first and midway stations.

Remark 3.3: The bang-bang control strategy *alone* does not close any feedback loop over the system, i.e., it is a kind of open loop control or schedule prescription. (It is currently under investigation how to close the feedback loop theoretically, see Section VIII.) Nevertheless, the introduction of `LDT` and `MaxWT` (Remark 2.2) sort of closes an *inherent* (not a *control*) feedback loop. Recall that without this a minimum quantity (the time of a time-optimal ride) could become infinity. Thus, precisely speaking, this robustification makes the problem mathematically well defined.

Remark 3.4: In the literature the problem of time-optimal traffic in large networks has been defined in different ways. These include the minimization of the total traveling time of the network ($\sum_{i=1}^{n_z} t_i^n - t_i^0$) and the minimization of individual or total delays as well as waiting times at stations.

The following definitions are helpful in the forthcoming development.

Definition 3.1: An *acceleration curve* is a path on which a train accelerates with full thrust. It can be either *static* or *dynamic*. A static acceleration curve starts from either the beginning of an interval or the end point of a safety block. The former is called an interval acceleration curve and the latter a safety block acceleration curve. As will be seen in the next section, all static acceleration curves may be determined offline. A dynamic acceleration curve starts from a dynamic reverse point (see next section) and has to be determined during the simulation process.

Definition 3.2: A *braking curve* is a path on which a train decelerates with full brake. If it ends at the end of an interval, it is called an interval braking curve and if it ends at the end point of a safety block,

it is called a safety block braking curve. A braking curve has a *static* nature and can always be found offline.

For an illustration of acceleration and braking curves see Figure 4. In this figure, $D2-A2$, $D3-A3$, $D5-C1$, $D6-C3$ and $D6-C4$ are interval acceleration curves, $D1-A1$ and $D4-C2$ are safety block acceleration curves, $B2-E1$, $C1-E3$, $C2-E3$, $C4-E4$, $B3-E6$, $F4-E6$ and $B4-E9$ are interval braking curves, $B1-E2$, $C3-E5$, $F1-E8$, $F2-F3-E7$ and $F5-E10$ are safety block braking curves, and $F1-F2$, $F3-F4$ and $F5-G1$ are dynamic acceleration curves.

IV. Computation of Switching Points

Numerical algorithms for the computation of all possible switching points are presented in this Section. To this end, first a taxonomy of all possible switching points in the time-optimal ride is presented. A switching point may be either *static* or *dynamic*.

A static switching point can be found offline and is one of the following: (SS1) A *static reaching point* where an interval velocity constraint is reached after speeding up on a static acceleration curve by full thrust; (SS2) a *leaving point* or a *braking point* where an interval velocity constraint is left (after being on the constraint) by full brake and a braking curve starts; (SS3) a *static reverse point* where the control reverses from full thrust on a static acceleration curve to full brake on a braking curve; (SS4) an *at least turn-on point* which is the beginning of a static acceleration curve; (SS5) an *at most turn-off point* which is the end of a braking curve.

A dynamic switching point cannot be computed offline (see Remark 4.3) and is one of the following: (DS1) a *dynamic reverse point* where the control reverses from full brake on a safety block braking curve to full thrust on a dynamic acceleration curve, or vice versa, where the control switches from full thrust on a dynamic acceleration curve to full brake on a braking curve. More precisely, suppose a train is on a safety block braking curve. If suddenly the blocking condition is removed, then the control must be switched from full brake to full thrust. This point is called a dynamic reverse point whence a dynamic acceleration curve starts. If this dynamic acceleration curve ends on a braking curve, then this point is also called a dynamic reverse point; (DS2) a *dynamic reaching point* where a dynamic acceleration curve ends on an interval velocity constraint.

For an illustration of static and dynamic switching points see Figure 4. In this figure, $A1-A3$ are static reaching points, $B1-B4$ are leaving points, $C1-C4$ are static reverse points, $D1-D6$ are at least turn-on points, $E1-E10$ are at most turn-off points, $F1-F5$ are dynamic reverse points, and $G1$ is a dynamic reaching point.

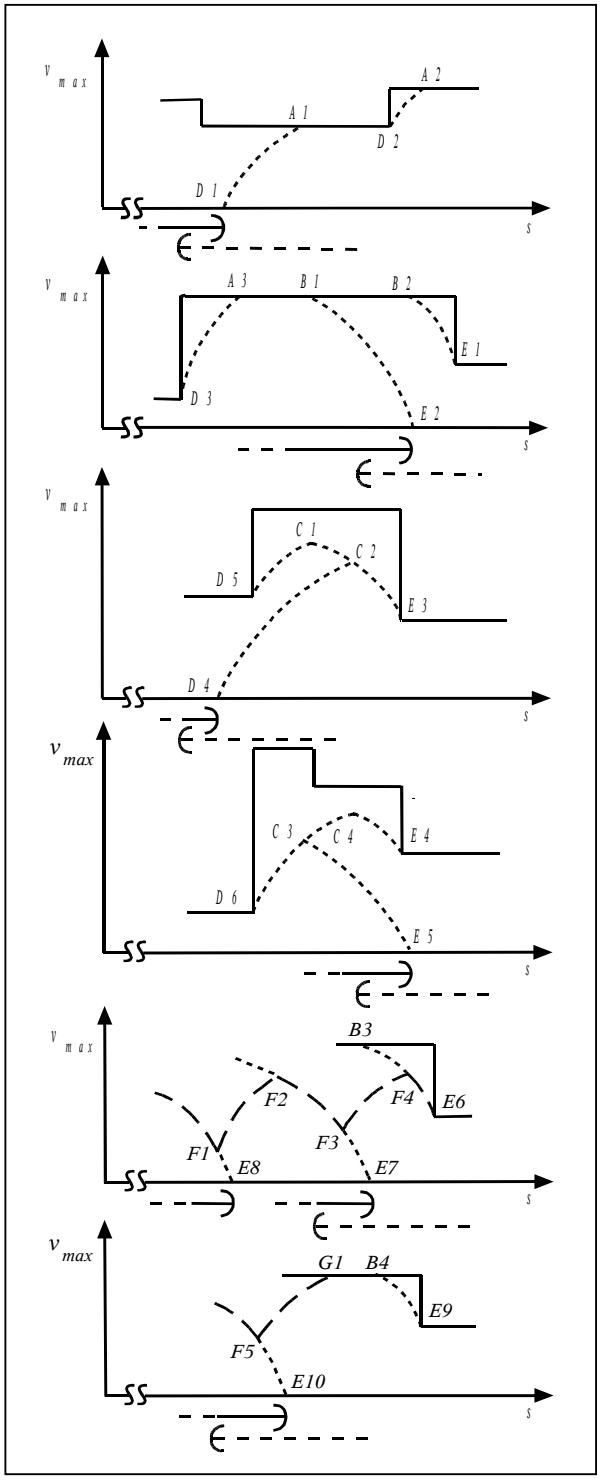

Fig. 4: Acceleration/Braking Curves and Switching Points

The above taxonomy provides the language of the proceeding development to find the switching points.

A. Static Reaching Points

To comply with the bang-bang control policy, a train must accelerate with full thrust to reach the constraint whenever its velocity is less than the velocity constraint. Therefore it is considered that the velocity of the train is less than the velocity constraint in interval *i*. The train must speed up and the

reaching point will be in say interval j ($j \geq i$; if interval i is short and/or the acceleration force is not enough, then the train reaches the constraint in interval $j > i$, otherwise $j = i$). To find the static reaching point, the time $\hat{t}$ when the velocity reaches the velocity constraint, i.e., when $v(\hat{t}) = v_{\max}(s(\hat{t}))$ will be found. This time instant is the root of the monotone function $\sigma_r(t) = v_{\max}(s(t)) - v(t)$ which can be computed by any root-finding method like the bisection, Newton's, or fixed-point iteration method [10]. In the following, the bisection method is adopted.

Algorithm 4.1.1 [5]: Bisection method to compute the time instant when a velocity constraint is reached (from below).

Input: Accuracy tolerance ε, starting step-size $\Delta(t)$ and initial values t_0, s_0 and v_0 (corresponding to the given initial point in the interval i).
Output: Numerical value for the time instant t_r^j and the position s_r^j where the constraint is reached in interval j ($j \geq i$).

1. Set $\hat{t}_0 = t_0$ and $x_0 = \begin{bmatrix} s_0 & v_0 \end{bmatrix}^T$.
2. Set $\hat{t}_1 = \hat{t}_0 + \Delta(t)$.
3. Compute $\sigma_r(\hat{t}_1)$ by solving the initial value problem:

$$\dot{x} = Ax + B\left(u_{uc}(s(t), v(t)) + u_{\max}(v(t))\right), \; t \in \left[\hat{t}_0, \hat{t}_1\right] \quad , \quad x(\hat{t}_0) = x_0 \, . \tag{11}$$

4. If $|\sigma_r(\hat{t}_1)| < \varepsilon$, then set: $t_r^j := \hat{t}_1$, $s_r^j := s(\hat{t}_1)$ and STOP.
5. If $\sigma_r(\hat{t}_1) < 0$, the constraint has been violated. Divide the step-size $\Delta(t) := \frac{1}{2}\Delta(t)$.
6. If $\sigma_r(\hat{t}_1) > 0$, the constraint has not been reached yet. Set: $x_0 := x(\hat{t}_1)$, $\hat{t}_0 := \hat{t}_1$.
7. GOTO Step 2.

B. *Leaving Points*

To comply with the velocity constraint at the beginning of an interval whose maximal allowable velocity is less than that of the preceding interval (or to stop at the end of a safety block), the constraint in the/a preceding interval must be left at a leaving point, or, there must be a static reverse point in the/a preceding interval. Thus, if the velocity constraint drops at the end of interval *k*, i.e., $v_{\max}^{k+1}(s^k + 0) < v_{\max}^k(s^k - 0)$, (or if the end point of a safety block is in interval *k*) it is necessary to find the corresponding leaving point in say interval j ($j \leq k$; if interval k is short and/or the braking force is not enough, then the train must start deceleration in interval $j < k$, otherwise $j = k$). It is also assumed that the would-be leaving point in interval *j* is after its reaching point, if any. (Otherwise, if the would-be leaving point in interval *j* is found to be before its reaching point, then in fact neither a reaching point nor a leaving point but a static reverse point is in that interval. This situation is addressed in the next subsection.)

A leaving point is found by backward computation. The *k*th interval braking curve is addressed in the following formulation. (A safety block braking curve is treated in a similar way). Analogous to the case of a reaching point, Algorithm 1 can be used with:

$$\dot{x} = Ax + B\left(-u_{uc}(s(t), v(t)) - u_{\min}(v(t))\right) \; , \; x(0) = [0, v_{\max}(s^k + 0)]^T \, , \tag{12}$$

and the initial values $t = 0$ and $s = 0$. The solution will be the length of the braking curve, i.e., $s_b = s^k - s_l^j$, and the absolute time t_b spent on the braking curve. The actual position of the leaving

point is thus given by $s_l^j = s^k - s_b$. The actual time of the leaving point is given by $t_l^j = t_r^j + t_c^j$ where t_r^j is known from Section IV.A and t_c^j is the time spent on the constraint in interval *j*, i.e., t_c^j is the root of $\sigma_c(t) = s(t) - s_l^j$ in which $s(t)$ is the solution of the initial value problem $\dot{s}(t) = v_{\max}(s(t))$, $s(t_r^j) = s_r^j$.

C. *Static Reverse Points*

As stated before, if the would-be leaving point in an interval is found to be before its reaching point, then in that interval there is neither a reaching point nor a leaving point but a static reverse point. A static reverse point is given by the intersection of a braking curve and a static acceleration curve. It is clear that the velocity at a static reverse point is less than or equal to the interval velocity constraint at that point.

To find the static reverse point in interval *j*, the trajectory of the velocity v_{fw} of the forward initial value problem (11) and the trajectory of the velocity v_{bw} of the backward initial value problem (12) are intersected. To this end, the time instant $\bar{t}_1$ at which $v_{fw}(\bar{t}_1) = v_{bw}(\bar{t}_1)$ and $s_{fw}(\bar{t}_1) + s_{bw}(\bar{t}_1) = s^k$ has to be determined. Thus, $\bar{t}_1$ is the answer if it is the root of $\sigma_{rev}(\bar{t}_1) = v_{fw}(\hat{t}_1) - v_{bw}(\bar{t}_1)$ subject to $s_{fw}(\hat{t}_1) = s^k - s_{bw}(\bar{t}_1)$ where $v_{fw}(\hat{t}_1)$ is the solution of the forward initial value problem (11) and $s_{bw}(\bar{t}_1)$ and $v_{bw}(\bar{t}_1)$ are the solutions of the backward initial value problem (12). The above procedure is performed by the following Algorithm.

Algorithm 4.3.1 [5]: Bisection method to compute the time instant when the control reverses from full thrust to full brake.

Input: Accuracy tolerance ε, starting step-size $\Delta(t)$, position s_0, velocity v_0 and time t_0 at a given point in interval *i*, the velocity constraint $v^{k+1}(s^k + 0)$ and the position s^k at the beginning of interval $k+1$.
Output: Numerical value for the time instant t_{rev}^j, the position s_{rev}^j and the velocity v_{rev}^j where control reverses from full thrust to full brake.

1. Set $\bar{t}_0 = 0$.
2. Set $\bar{t}_1 = \bar{t}_0 + \Delta(t)$.
3. Solve the initial value problem (12) for $t \in [\bar{t}_0, \bar{t}_1]$ and $x(\bar{t}_0) = \left[0, v_{\max}(s^k + 0)\right]^T$ to find $s_{bw}(\bar{t}_1)$ and $v_{bw}(\bar{t}_1)$.
4. Solve the initial value problem (11) for $t \in [\hat{t}_0, \hat{t}_1]$ and $x(\hat{t}_0) = [s_0, v_0]^T$ to find the time $\hat{t}_1$ at which $s_{fw}(\hat{t}_1) = s^k - s_{bw}(\bar{t}_1)$. At this time instant compute $v_{fw}(\hat{t}_1)$.
5. Compute $\sigma_{rev}(\bar{t}_1) = v_{fw}(\hat{t}_1) - v_{bw}(\bar{t}_1)$.
6. If $|\sigma_{rev}(\bar{t}_1)| < \varepsilon$ then set:
   $t_{rev}^j := \hat{t}_1$,
   $t_b := \bar{t}_1$,
   $s_{rev}^j := s_{fw}(\hat{t}_1)$, $\quad (= s^k - s_{bw}(\bar{t}_1))$
   $v_{rev}^j := v_{fw}(\hat{t}_1)$, $\quad (= v_{bw}(\bar{t}_1))$
   STOP.
7. If $\sigma_{rev}(\bar{t}_1) < 0$, then the static reverse point has not been reached yet. Set: $\bar{t}_0 := \bar{t}_1$.

8. If $\sigma_{rev}(\bar{t}_1) > 0$, the static reverse point has been passed. Divide the step-size: $\Delta(t) := \frac{1}{2}\Delta(t)$.

9. GOTO 2.

Thus, the static switching points (SS1)-(SS3) are computed offline by the above numerical algorithms. However, it is clear that the at least turn-on and the at most turn-off points (SS4) and (SS5) may be found even analytically.

Remark 4.1: Only if the corresponding blocking and unblocking conditions emerge will the safety blocks related static switching points materialize. For example, in Figure 3 only if the 2nd train enters the safety the block *j* while the 1st Train is in the safety block *l* does the switching point due to the braking curve of the safety block *j*, i.e., b_j , and thus b_j itself, materialize. Materialization of a safety block related switching point may prevent that of some switching points. For instance, in Figure 4, materialization on `D1-A1` does not prevent that of `D2`, but materialization of `E2` does prevent that of `B2`.

Remark 4.2: The maximal allowable velocity (Figure 3, d) is given by the minimum of the velocities dictated by the interval velocity constraints and interval braking curves (Figure 3, b) and by the safety block braking curves (Figure 3, c) whenever and whichever materialized (Remark 4.1).

Remark 4.3: A dynamic reverse point which is the beginning of a dynamic acceleration curve may emerge (due to an unblocking condition) only in the course of online simulation. Thus, it cannot be computed offline. However, the end of a dynamic acceleration curve (which is either a dynamic reaching point or a dynamic reverse point) can be computed semi-offline. In other words, when the beginning of a dynamic acceleration curve emerges, its end can be computed before it materializes. To this end, whenever the beginning of a dynamic acceleration curve emerges, its end is found like a static reaching point or a static reverse point but in the course of online simulation.

V. Actual Networks

In an actual network, like that of the German railway, some specifications may be relaxed. More precisely, in many real networks the following assumptions can be made: (A1) Within all intervals the maximal allowable velocity is a constant function of the position (as shown in Figure 3 and Figure 4); (A2) the acceleration and braking forces are constant functions of the position and continuous functions of the velocity; (A3) at the end point of intervals jumps in the controllable force u_c are realizable.

It is easy to check that under these assumptions there is no need to solve an initial value problem to find a braking curve. Instead, a braking curve is given by the following simple algebraic formulae $v(s) = a(t - t_0) + v(s_0) = \sqrt{v^2(s_0) + 2a(s - s_0)}$ in which the constant a is the maximal deceleration and (s, v, t) and (s_0, v_0, t_0) denote the position, velocity and time of two of its points. In particular, (s_0, v_0) can be chosen the end point of the braking curve, since the at most turn-off points are known without any computations. Moreover, provided $u_c = u_{\max}(v)$ is enough to satisfy a velocity constraint, there is no need to solve an ODE and the following simple algebraic formula holds on an interval velocity constraint: $v = at$.

Remark 5.1: It is observed that only on acceleration curves one needs to solve initial value problems. As soon as a train reaches its maximal allowable velocity (Remark 4.2) its next position and velocity are given by some inexpensive algebraic computations. Consequently, the aforementioned assumptions are in effect some simplifications under which the simulation of the network is accelerated significantly.

VI. Simulation Strategy

To simulate the network, i.e., to find the position and velocity of every train versus time, a simulation software must be written. Such an industrial software is currently under development. Already available is a pilot simulation program by which some small-scale but real networks have been simulated. These results will be presented in the next Section. Prior to this, we explain the simulation strategy in the following.

The input to the simulation software is all the information of the network supplied through an interface e.g. an XML file [2],[9]. Inside the simulation program the interface file is read using e.g. a SAX-oriented XML C++ parser [2],[9]. It is sorted so that the simple blocking and unblocking conditions which are induced by the motion of each train on the motion of other trains are identified and categorized in the data structure of that train. Inside the simulation program, the data structure is like that of some nested classes. There is a big class of trains with n_z' trains as its objects, where n_z' denotes the total number of trains having interaction with some others. (This will be later defined in Definitions 6.1-6.3, read through.) Each object, i.e., each train, includes the specifications of the train ID and its wagons plus four classes, the class of locomotives, the class of intervals, that of safety blocks, and that of all extended and simple blocking and unblocking conditions (which are caused by its motion). In turn, each object of the class of locomotives includes the class of its power-velocity data in addition to some other specifications (like its mass – describing its dynamics) needed to solve (4)-(10).

The simulation is based on the flowchart given in Figure 5. In this figure β (of a safety block) denotes the indicator function of a safety block defined as the number of blockings of that safety block (see Section VII, Example 7.1), and $v_{\max}^{i}$ denotes the maximal allowable velocity (Remark 4.2). The simulation starts by offline computation of all possible static switching points, the initial value of the indicator functions of all safety blocks and the initial maximal allowable velocity of all the trains. Then online simulation starts as follows. In a round-robin fashion for every train the following is done with the local step-size $\Delta(t^l)$: By checking the current velocity and the indicator function of the current safety block, it is checked if the train can go forward or has to wait. If it has to wait the program goes to the next train. If it can go forward, it is checked if the train is on its maximal allowable velocity. Hence, there are the following two cases:

a) The train is not on its maximal allowable velocity. Thus, the train must accelerate with full thrust (represented by $a_{\max}^{+}$) on a static/dynamic acceleration curve. Then it is checked if the maximal allowable velocity is violated. If not, the program goes to the next train. Otherwise, the above computation is corrected (using interpolation) so that the corresponding static reaching point/static reverse point/dynamic reaching point/dynamic reverse point is arrived at; then the program goes to the next train.

b) The train is on the maximal allowable velocity. Then it is checked if the train is on a braking curve. If so, the train decelerates with full brake (represented by $a_{\min}^{-}$); if not, the train moves with the same speed. In either case it is then checked if the train is still on its maximal allowable velocity. If so, the program goes to the next train. If not, the above computation is corrected (using interpolation) such that the corresponding at most turn-off point or leaving point is arrived at, respectively; then the program goes to the next train.

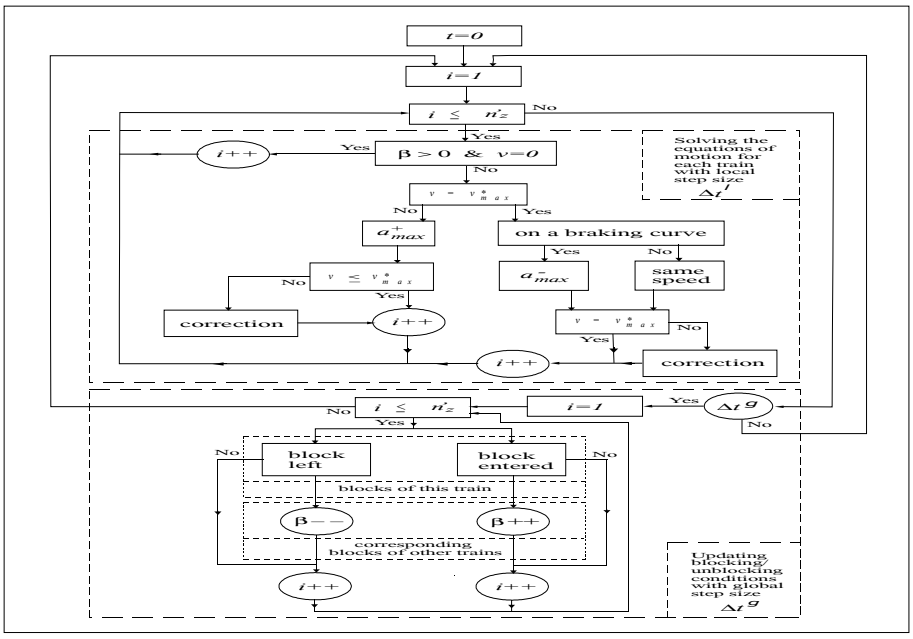

Fig. 5: Flowchart of the Simulation Strategy

After each round of the above procedure, it is checked if the global step-size $\Delta(t^g)$ has been reached. If not, the above procedure is repeated. If so, the following is done for every train. It is checked if a safety block has been left or entered. If not, the program goes to the next train. If so, the

indicator functions of the corresponding safety blocks of other trains which are affected by the current safety block of this train, if any, are decremented or incremented, respectively, considering the time. (Note that the extended blocking and unblocking conditions incorporate some explicit times.) Then the program goes to the next train. It should be noted that at this stage, that all the indicator functions of the safety blocks are checked for possible updates, the corresponding maximal allowable velocities (Remark 4.2) are updated if necessary. The simulation ends when all the trains have reached their end points.

The purpose of the introduction of the two local and global step-sizes is to speed up the simulation. By the local step-size selection strategy, the desired accuracy of the simulation of each train motion is achieved. By the global step-size it is checked if a safety block is entered or left. Since in general a safety block is not entered or left at a local step-size and because the exact time a safety block is entered or left is not needed, the approximate time a safety block is entered or left is found via the global step-size which is larger than the local step-size.

Remark 6.1: The local and global step-sizes are two positive constants. To choose them the following should be considered: a) the desired accuracy of the simulation (inverse relation), and b) the desired speed of the simulation (inverse relation). Note that the accuracy itself depends on the ODE in question. This specifically means that every train (every ODE) has its own (maximum allowable) local step-size. However, there is a unique local step-size for the network (used in the simulation software) which is the minimum of the maximum allowable local step-sizes, see [5],[10],[24] for more details and some comparative studies.

The philosophy behind n_z' (in Figure 5) can be explained as follows. In a real network not every train is interacting with some others. Moreover, none of the trains that have interaction with some other trains is all the time interacting with them. The following definitions, which are useful in the ensuing discussions, are a classification of trains versus interaction.

Definition 6.1: The trains having no interaction with any other train are called the *non-interactive* trains.

Definition 6.2: The trains having interaction with some other train(s) are the so-called *interactive* trains. The total number of them is denoted by n_z'. (In actual large-scale networks $n_z' << n_z$, where n_z denotes the total number of trains.)

Definition 6.3: The interactive trains while interacting are called the *active-interactive* trains. (Note that they form a non-empty, non-full subset of interactive trains, and that the size and members of this subset depend on time, more precisely, on the status of the whole network at any given time.)

Remark 6.2: In a large-scale system with n_z , $n_z' >> 1$ trains (in the German railway $n_z \simeq 4000$), each causing in average *N* (extended and simple) blocking and unblocking conditions, the list of all blocking and unblocking conditions has $n_z' N$ elements. This list must be swept for each train at each global step size. By the classification of blocking/unblocking conditions as explained in the first paragraph of this section, each train will have its exclusive list with (about) *N* elements; thus, $(n_z' - 1)N \simeq n_z' N$ supernumerary elements are expunged from that list of each train. The simulation is thus highly accelerated.

Remark 6.3: Only the interactive trains are included in the flowchart of online simulation. In other words, the simulation of non-interactive trains is done offline and this greatly expedites the online simulation. Some general picture about the number of active-interactive, interactive and non-interactive trains in a typical large-scale network is offered in Section VII.

Remark 6.4: The simulation of non-interactive trains can be done using the same flowchart as that in Figure 5. The only difference is that in the lower dashed box (entitled “Updating blocking/unblocking

conditions") there are no safety blocks of other trains. Note that this box is considered, since there may be some safety blocks of the same train due to extended blocking and unblocking conditions.

Remark 6.5: Based on this strategy, faster-than-real-time simulation of the simulation is obtained, because: a) there is no graph to go through, there are some lists in parallel; the approach is well suited for parallel processing and can be implemented on parallel processors, b) all possible static switching points and maximal allowable velocities are computed offline, c) only on acceleration curves there are initial value problems to be solved – as soon as a train gets on its maximal allowable velocity its position and velocity in the next iteration are obtained by some inexpensive algebraic computations, and d) some lengthy and superfluous computations are circumvented by d1) the inclusion of interactive trains only, and d2) the introduction of the global and local step sizes and the classification of the (extended and simple) blocking and unblocking conditions in the data structure of every train. As will be discussed in Section VII, the above-mentioned expected computational speed is realistic.

VII. Worked out Examples

To illustrate the pragmatism of the proposed methodology for modeling and simulation of time-optimal train traffic in large networks, some *real* small-scale networks (which are parts of the German railway network) have been simulated by a pilot simulation program [24]. The implementation in the real operating centers of the German railway system is currently being considered.

Prior to giving some simulation results, the general procedure and guidelines of Section II are illustrated in a concrete example. To this end, the construction of a model and its associated simple and extended blocking and unblocking conditions for a given physical network are detailed.

Example 7.1: Given the network in Figure 6, which can represent part of a real network. In this network, ST-A, ST-B, and ST-X stand for Station A, Station B, and Station X, respectively, and M-F represents Merge F. M-F and ST-X are connected by one track only. From among the many trains traveling in this network only six are considered, which have the following routes: 1st Train and 2nd Train: ST-A—M-F—ST-X—y1,2, 3rd Train and 4th Train: ST-B—M-F—ST-X—y3,4, 8th Train and 9th Train: x8,9—ST-X—y8,9. Besides, P-I, P-J, P-M, and P-N denote Point I, Point J, Point M, and Point N, respectively. These points refer to some points on the routes where there are some sharp bends. In particular, P-I and P-N are on some bridges on which a train is not allowed to stop but to pass.

It is clear that in case of no safety observation there is a possibility of crash between any two or more of 1st Train, 2nd Train, 3rd Train, and 4th Train, and also between 8th Train and 9th Train, but not between these two groups since ST-X is a station not a join (for the paths of these two groups of trains). As such, a model is required for this network, which will be built in the ensuing development in line with the modeling steps of Section II.

Step 1. We consider every train as a mass point moving on its own supposed sole individual path/track which itself is considered as a straight line. However, due to lack of space, and in order to emphasize that trains with the same route share the same track, in Figure 7 only one track is depicted for any such trains. Also, for the sake of clarity a train is shown by a wagon not by a mass point.

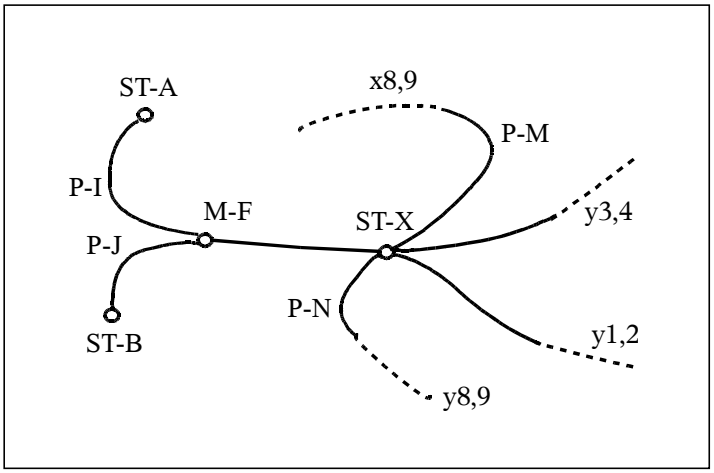

Fig. 6: A representative part of a real network

Step 2. The path of every train is divided into some adjacent intervals, in each a maximal velocity being allowed. As stated in Section I, it is possible and it is currently under investigation to have different maximal allowable velocities for different trains. However, for the sake of simplicity, and without loss of generality, we suffice to a unique maximal allowable velocity for each point of the tracks. Thus, as depicted in Figure 7, for instance, all the trains passing through ST-B—M-F have the same maximal allowable velocity on ST-B—M-F, and all the trains passing through M-F—ST-X have the same maximal allowable velocity on M-F—ST-X. Moreover, note that the intervals containing P-I, P-J, P-M, and P-N have reasonably small maximal allowable velocities because of the sharp bends.

Step 3: The supposed sole individual path of every train is divided into some overlapping safety blocks, as shown in Figure 7. In this figure, in the exclusive index i, j (written over each safety block) the first argument (*i*) refers to the train number and the second argument (*j*) refers to the number of that safety block, where $i \in \{1,2,3,4,8,9\}$ and $j \in \{1,2,...\}$. We will refer to the *j*th safety block of the *i*th train by $< i, j >$.

Moreover, for the sake of clarity, the beginning and end points of those safety blocks which are physically the same are connected by dashed lines, e.g., <1,8>,<2,8>,<3,7>,<4,7>. Besides, such safety blocks which *also* represent trains with the same route have the same arguments, e.g., <1,4>,<2,4>; <3,8>,<4,8>; <8,18>,<9,18>. It is clear that another scheme may also be used to index the safety blocks.

It is also noted that because P-I and P-N refer to some bridges, no safety block ends on these points. In fact, the end points of the safety blocks containing P-I and P-N are far enough so that in case a train has to stop in those safety blocks, it does not stop on the bridge.

The safety concept can simply be explained as follows. While Train 1 is in <1,3>, Train 2 is not allowed to enter <2,3> (physically the same as <1,3>) which means that it must stop at the end of <2,1>. When Train 1 has left <1,3>, Train 2 is allowed to enter <2,3> which means that it can leave <2,1>. The same applies to <1,*k*> and <2,*k*-2> for the rest of the path. Thus,

$k \in \{3,4,...\}$:
<1,*k*> `BLOCKS` <2,*k*-2>,
<1,*k*> `UNBLOCKS` <2,*k*-2>.

On the other hand, the converse is also valid if Train 2 is leading Train 1. That is, we must also have,

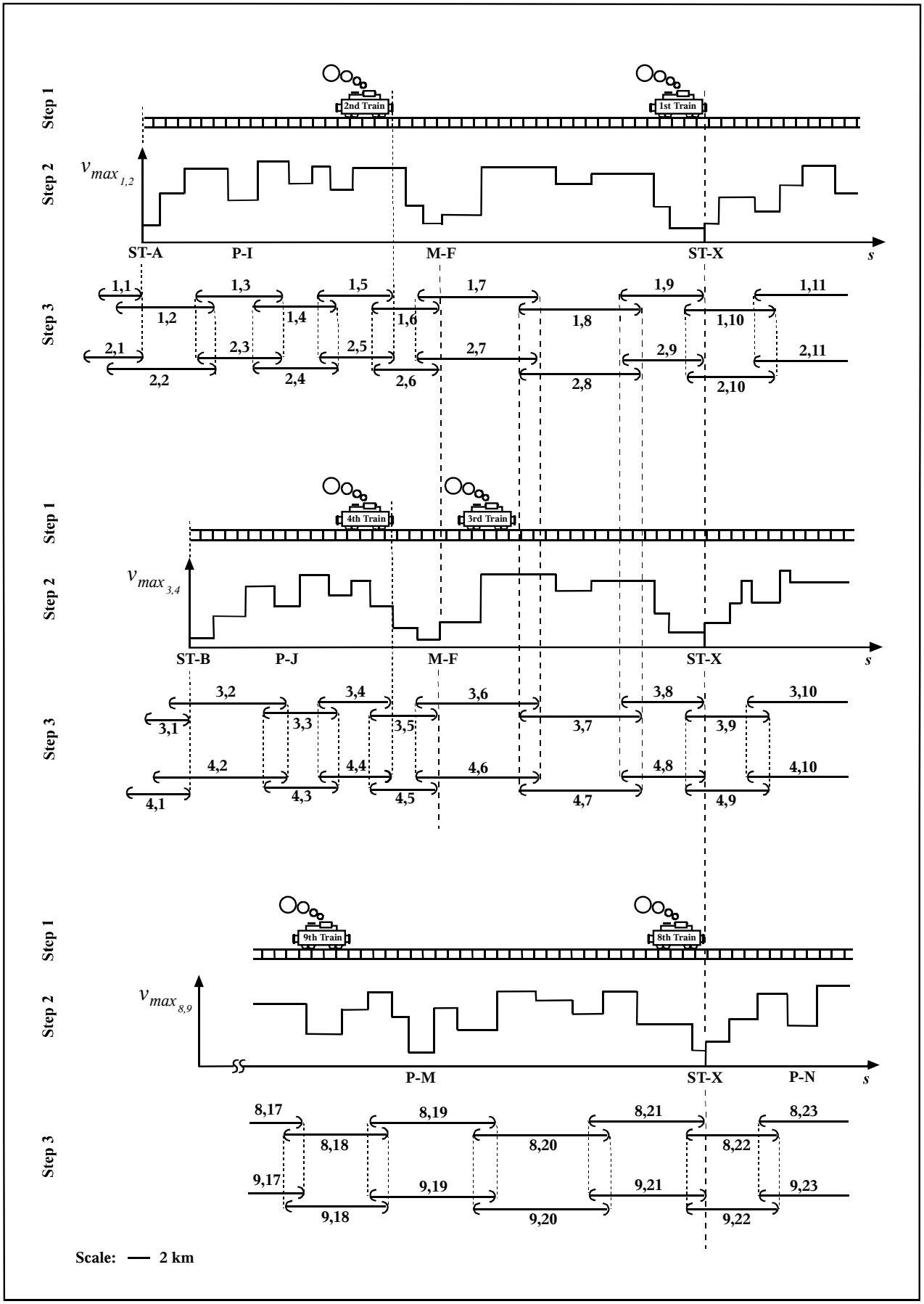

Fig. 7: The first three steps of modeling of the network in Figure 6. The length scale does not apply to the trains. In the given scale a train should be shown by a point " . " but it is depicted as a wagon for the sake of clarity.

$k \in \{3,4,...\}$:
<2,*k*> BLOCKS <1,*k*-2>,
<2,*k*> UNBLOCKS <1,*k*-2>.

In practice both of the above sets of rules are needed even if sequencing enforce only one situation, that e.g. Train 1 leads Train 2. The reason is that if the required sequencing is not followed for any reason, safety must still be observed.

Similar arguments apply to the other four trains. Consequently,

$k \in \{3,4,...\}$:
<3,*k*> BLOCKS <4,*k*-2>,
<3,*k*> UNBLOCKS <4,*k*-2>,

$k \in \{3,4,...\}$:
<4,*k*> BLOCKS <3,*k*-2>,
<4,*k*> UNBLOCKS <3,*k*-2>,

and, assuming that Trains 8 and 9 share the same physical safety blocks with the same arguments throughout their path,

$k \in \{3,4,...\}$:
<8,*k*> BLOCKS <9,*k*-2>,
<8,*k*> UNBLOCKS <9,*k*-2>,

$k \in \{3,4,...\}$:
<9,*k*> BLOCKS <8,*k*-2>,
<9,*k*> UNBLOCKS <8,*k*-2>.

The proceeding sets of simple blocking and unblocking conditions guarantee the safe operation of trains with the same route up to their interaction due to M-F—ST-X. To resolve this, consider Trains 1 and 3, where the former leads the latter. The following set of rules is needed:

<1,7> BLOCKS <3,4>,
<1,7> UNBLOCKS <3,4>,
<1,8> BLOCKS <3,5>,
<1,8> UNBLOCKS <3,5>,
<1,9> BLOCKS <3,6>,
<1,9> UNBLOCKS <3,6>,
<1,10> BLOCKS <3,7>,
<1,10> UNBLOCKS <3,7>.

If Train 3 leads Train 1, the following set of rules is needed:

<3,6> BLOCKS <1,5>,
<3,6> UNBLOCKS <1,5>,
<3,7> BLOCKS <1,6>,
<3,7> UNBLOCKS <1,6>,
<3,8> BLOCKS <1,7>,
<3,8> UNBLOCKS <1,7>,
<3,9> BLOCKS <1,8>,
<3,9> UNBLOCKS <1,8>.

With a similar argument, it is clear that in practice both of the above sets of rules are required.

Similar rules are needed for Trains 1 and 4, as follows:

<1,7> BLOCKS <4,4>,
<1,7> UNBLOCKS <4,4>,
<1,8> BLOCKS <4,5>,
<1,8> UNBLOCKS <4,5>,
<1,9> BLOCKS <4,6>,
<1,9> UNBLOCKS <4,6>,
<1,10> BLOCKS <4,7>,
<1,10> UNBLOCKS <4,7>,

<4,6> BLOCKS <1,5>,
<4,6> UNBLOCKS <1,5>,
<4,7> BLOCKS <1,6>,
<4,7> UNBLOCKS <1,6>,
<4,8> BLOCKS <1,7>,
<4,8> UNBLOCKS <1,7>,
<4,9> BLOCKS <1,8>,
<4,9> UNBLOCKS <1,8>.

Analogously, the following rules are required for Trains 2 and 3:

<2,7> BLOCKS <3,4>,
<2,7> UNBLOCKS <3,4>,
<2,8> BLOCKS <3,5>,
<2,8> UNBLOCKS <3,5>,
<2,9> BLOCKS <3,6>,
<2,9> UNBLOCKS <3,6>,
<2,10> BLOCKS <3,7>,
<2,10> UNBLOCKS <3,7>,

<3,6> BLOCKS <2,5>,
<3,6> UNBLOCKS <2,5>,
<3,7> BLOCKS <2,6>,
<3,7> UNBLOCKS <2,6>,
<3,8> BLOCKS <2,7>,
<3,8> UNBLOCKS <2,7>,
<3,9> BLOCKS <2,8>,
<3,9> UNBLOCKS <2,8>.

All the above rules together will guarantee safe operation in the network, i.e., there will not be any crash between consecutive trains or those coming to a join. Nevertheless, they do not address any schedule. This will be handled in the sequel.

Step 4. In this step initialization of the journeys, sequencing, correspondences and priorities at joins are examined in details, but for the sake of brevity not for all trains.

Initialization: Suppose Train 1 is to start at $t = t_1^0$. The sequel set of extended blocking and unblocking conditions is used:

<1,1> INITIALLY BLOCKED,

<1,1> UNBLOCKED at t_1^0.

The first command prevents Train 1 from starting by blocking <1,1>. The second command releases the blocking of <1,1> at t_1^0.

Sequencing: If it is required that Train 2 follows Train 1, then it suffices to add,

<2,1> INITIALLY BLOCKED,
<1,3> UNBLOCKES <2,1>.

Thus, Train 2 will start its journey when Train 1 has left <1,3>. On the other hand, if it is desired that Train 2 starts at $t = t_2^0$, then,

<2,1> INITIALLY BLOCKED,
<2,1> UNBLOCKED at t_2^0,

is used. Note that this automatically enforces some sequencing depending on which starting time is smaller (sooner).

Priorities: Suppose that Train 3 is desired to lead Train 1on M-F—ST-X. In other words, suppose that that Train 3 has priority over Train 1 at M-F, then this can be realized by

<1,5> INITIALLY BLOCKED,
<3,6> UNBLOCKS <1,5>,

with the following explanation. The first command means that Train 1 has to stop at the end of <1,5>, and the second command means that Train 1 resumes its journey when Train 3 has left <3,6>. More precisely, if the dynamics of the network is such that Train 1 has priority over Train 3 at M-F, then, because of these commands, Train 1 stops at the end of <1,5>, and resumes its journey when Train 3 has left <3,6>. On the other hand, if the dynamics of the network is such that Train 3 has left <3,6> before Train 1 reaches the braking point of <1,5>, then Train 1 will not stop in <1,5>.

Equivalently, this can be achieved by,

<1,6> INITIALLY BLOCKED,
<3,7> UNBLOCKS <1,6>,

with a similar explanation.

Correspondences: Given that Train 9 is required to wait for Train 2 at ST-X, and that 1 minute after Train 2 starts its journey it should move. This situation can represent the case that some passengers of Train 2 want to transfer to Train 2 at ST-X, and that they have 1 minute to catch it. This can be accomplished by,

<9,21> INITIALLY BLOCKED,
<2,9> UNBLOCKS <9,21> after $t_9^{MinWT} = 1\min$,

with the subsequent effect. The first command requires that Train 9 waits at ST-X. The second command allows it to resume its journey 1 minute after Train 2 has resumed its journey from ST-X.

However, the above will not take place in the following situation. Suppose Train 9 reaches the braking point of <9,21> (in order to stop at ST-X) more that one minute after Train 2 has left <2,9>.

This means that Train 9 will not stop and thus the passengers of Train 2 who are waiting for it cannot take it. Hence, the following rules are also needed:

<9,21> INITIALLY BLOCKED,
<9,21> UNBLOCKED after t_9^{MaxWT},

which make Train 9 stop at ST-X and resume its journey after the prescribed Maximum Waiting Time t_9^{MaxWT}. It is also possible to replace the second command with,

<9,21> UNBLOCKED at t_9^{LDT},

in which t_9^{LDT} denotes the prescribed Latest Departure Time.

Robustness: In the above correspondence it is clear that Train 9 should not wait unreasonably long for Train 2, if the latter does not arrive at ST-X at the planned time. Thus, to the above two sets of commands (i.e., four commands) we can add,

<9,21> UNBLOCKED after t_9^{MaxWT},

which allows Train 9 to resume its journey after a prescribed maximum waiting time t_9^{MaxWT}.

Finally, some issues are further clarified:

a) The number of intervals (i.e., their lengths, and also their corresponding maximal allowable velocities) depends on the network (including its environment). In practice the number of intervals may be much more than what we have shown in this example, see e.g. Example 7.2, Figures 12, 16-18.

b) By intuitive and innovative combinations of the four extended blocking and unblocking conditions mentioned in Section II, we can model any initialization, sequencing, priority, and correspondence with a desired degree of robustness and flexibility (in terms of time). It should also be noted that SDT and LDT are explicit times (like 9:05 pm), whereas MinWT and MaxWT are relative times (like 2 min).

c) In Section VI, Figure 5, the parameter β of a safety block, called its indicator function, was defined as the number of blockings of that safety block. From its definition it is clear that β is a function of time:

The initial value of all safety blocks $< i, j >$ are set to zero, $\beta^{i,j} = 0$. A blocking on $< i, j >$ increases $\beta^{i,j}$ by 1 and an unblocking of $< i, j >$ decreases $\beta^{i,j}$ by 1. For instance, consider the commands,

$< i, j >$ BLOCKS $< k, l >$,
$< i, j >$ UNBLOCKS $< k, l >$.

When Train I enters $< i, j >$ (which is a time instant – a dynamic condition), the first command makes $\beta^{k,l}++$, i.e., it increases the current value of $\beta^{k,l}$ by 1: $\beta^{k,l} = \beta^{k,l} + 1$. When Train i leaves $< i, j >$ (also a time instant – a dynamic condition), the second command makes $\beta^{k,l}--$, i.e., it decreases the current value of $\beta^{k,l}$ by 1: $\beta^{k,l} = \beta^{k,l} - 1$. Moreover,

$< i, j >$ INITIALLY BLOCKED,

sets $\beta^{i,j} = \beta^{i,j} + 1$. Besides,

$< i, j >$ UNBLOCKED at SDT,
$< i, j >$ UNBLOCKED at LDT,
$< i, j >$ UNBLOCKED after MaxWT,

result in $\beta^{i,j} --$ at SDT and LDT, and after MaxWT, respectively. On the other hand,

$< i, j >$ UNBLOCKS $< k, l >$ after MinWT,

renders $\beta^{k,l} --$ after MinWT.

Two specific examples are as follows. As the first example consider the initialization,

<1,1> INITIALLY BLOCKED,
<1,1> UNBLOCKED at t_1^0.

The first command sets $\beta^{1,1} = 0 + 1 = 1$, thus the train has to stop (or in this case remain stopped). The second command takes effect at t_1^0, with the result $\beta^{1,1} = 1 - 1 = 0$, and thus the train can move.

The second example is more complicated. Given the earlier-mentioned correspondence between Trains 2 and 9, where,

<9,21> INITIALLY BLOCKED,
<2,9> UNBLOCKS <9,21> after $t_9^{MinWT} = 1\min$,
<9,21> INITIALLY BLOCKED,
<9,21> UNBLOCKED after t_9^{MaxWT},
<9,21> UNBLOCKED after t_9^{MaxWT}.

The effect of these rules is as follows (supposing that at that time there in no other blocking/unblocking condition on <9,21>). The first and third commands make $\beta^{9,21} = 0 + 1 = 1$ and $\beta^{9,21} = 1 + 1 = 2$, respectively. Thus, Train 9 has to stop at the end of <9,21> when it reached there, since $\beta^{9,21} > 0$. However, after t_9^{MaxWT}, $\beta^{9,21} = 2 - 1 - 1 - 1 = -1$ if the second command has become active, and $\beta^{9,21} = 2 - 1 - 1 = 0$ if the second command has not become active. In either case $\beta^{9,21}$ is not greater than zero which means (subject to the following Remark) that Train 9 can resume its journey.

Remark 7.1: Every $\beta^{i,j} < 0$ is replaced by $\beta^{i,j} = 0$. Otherwise, for instance in the above situation ($\beta^{9,21} = -1$) suppose there is another blocking on <9,21> by another train. Then, $\beta^{9,21} = -1 + 1 = 0$, which means there is no blocking on Train 9 and it can move! This violates the safety concept and potentially results in a crash if not in off-the-schedule traffic. This problem, on the other hand, can be resolved by introducing some (small) *dummy* or *imaginary* safety blocks (just before the stations/merges) inside the simulation software. However, because it slows down the simulation (due to checking their entering and leaving) we do not further address this issue here.

d) In an actual network some routes are circled and some are commuted. In both cases the stations from/to which some consecutive trains start/go, have different platforms for them. For instance, suppose the three trains, 3, 4, and 5 Train leave/enter ST-B. In case they circle their route, when they

circle once and want to do it for a second time, they are treated as different trains with different numbers (and thus different exclusive safety blocks – physically the same as before but with indices whose first arguments are different, denoting the new train number) and probably different schedule. The latter means that probably some sequencing, priorities, or correspondences will be different. This in turn specifically means that a new set of simple and extended blocking and unblocking conditions are needed for the new trains. However, it is noteworthy that simple blocking and unblocking conditions will be the same up to the train index. More precisely, call the new trains e.g. Trains 374, 375, and 376, representing respectively Trains 3, 4, 5 on their second circle. Thus, instead of,

$k \in \{3,4,...\}$:
<3,k> BLOCKS <4,k-2>,
<3,k> UNBLOCKS <4,k-2>,

$k \in \{3,4,...\}$:
<4,k> BLOCKS <3,k-2>,
<4,k> UNBLOCKS <3,k-2>,

we will have,

$k \in \{3,4,...\}$:
<374,k> BLOCKS <375,k-2>,
<374,k> UNBLOCKS <375,k-2>,

$k \in \{3,4,...\}$:
<375,k> BLOCKS <374,k-2>,
<375,k> UNBLOCKS <374,k-2>,

If they commute their route, when they reach their destination and want to come back, with an analogous argument as that in the above, they will be treated as new trains (thus new exclusive safety blocks and with a new set of simple and extended blocking and unblocking conditions). A similar argument (with appropriate changes) applies to the new blocking and unblocking conditions as well.

In both cases, it is clear that with intuitive and elegant indexing the situations can become tangible and meaningful.

e) As it is observed, the safety concept requires that a minimum distance be always observed between consecutive trains, which depends on their positions and safety block placement. For instance, this minimum distance is $s_b^{1,9} - s_e^{1,7}$ if the leading train is in <1,9>, and $s_b^{3,5} - s_e^{3,3}$ if the leading train is in <3,5>, where $s_b^{i,j}$ and $s_e^{i,j}$ denote the positions of the beginning and end point of $< i, j >$.

This issue gives rise to a trade-off: the shorter this distance, the more the trains, i.e., the higher the flow and capacity of transportation, but the higher the number of safety blocks and the higher the implementation cost. As stated in Remark 2.5, this is currently under further investigation. It is clear that this is an optimization problem in which many factors are involved, among others the smoothness of ride and passengers comfort.

f) In view of the above issue, a *double measure* can be enforced for the safety concept by lengthening the minimum distance. This can be achieved by changing,

$k \in \{3,4,...\}$:
<1,k> BLOCKS <2,k-2>,
<1,k> UNBLOCKS <2,k-2>,

to,

$k \in \{5,6,...\}$:
<1,*k*> BLOCKS <2,*k*-4>,
<1,*k*> UNBLOCKS <2,*k*-4>,

with appropriate changes in other commands.

g) It is evident that whenever there is a priority, it (or equivalently, the network) must be *robustified* as in the case of a correspondence.

h) The situation in Figure 7 is explained as follows.
h1) The beginning points of the first and second safety blocks of trains with the same route – e.g. $s_b^{1,1}$, $s_b^{2,1}$ and $s_b^{1,2}$, $s_b^{2,2}$ – are not necessarily the same, since they have no effect.
h2) There is a correspondence between Trains 1 and 8.
h3) Train 3 is not allowed to leave <3,6> while Train 1 is in <1,9>. (Note that Train 3 has not yet reached the end point of <3,6>.)
h4) Both Train 2 and 4 are waiting and none of them is allowed to resume its journey as while as Train 3 is in <3,6>.
h5) A priority should determine either Train 2 or 4 to resume its journey first.
h6) Train 9 is not allowed to leave <9,19> while Train 8 is in <8,21>. (Note that Train 9 has not yet reached the end point of <9,19>.)

Some simulation results are presented in the following. More examples can be found in [5],[24].

Example 7.2 [5],[24]: This real example illustrates merge of two tracks like ST-A—M-F—ST-X and ST-B—M-F—ST-X of the preceding example, with a single train on each. The separate parts, i.e., ST-A—M-F and ST-B—M-F, have the same length about 13.5 km. The shared part, i.e., M-F—ST-B, is about 16.5 km. We present four scenarios.

1) There is no train scheduling. Train 1 starts at time 0 and Train 2 starts at time 90 sec.
2) There is no train scheduling. Train 2 starts at time 0 and Train 1 starts at time 90 sec.
3) There is a train scheduling, enforcing train 1 to pass the switch first. Train 1 starts at time 0 and Train 2 at time 90 sec.
4) There is a train scheduling, enforcing train 1 to pass the switch first. Train 2 starts at time 0 and Train 1 at time 90 sec.

The time-path diagrams are depicted in Figures 8-11, respectively.

Example 7.3 [5],[24]: This example is a demonstration of the computational effort. It has been carried out on a Pentium 2, 450 MHz, 384 MB RAM. The simulation consists of a single *but interactive* train over 80 km of a real piece of track called “22127-1013” of the German Rail network with a subdivision of 1275 safety blocks. The maximal allowable velocity over the whole path (consisting of many intervals) is shown in Figure 12. The uncontrollable part of the control input, u_{uc}, is illustrated in Figure 13. Figure 14 demonstrates the maximal braking force, while Figure 15 depicts the maximal control input.

The data of the Figures 12-15 as well as the specifications of this train and its relation with others are input to the pilot simulation program through an interface. Reading the interface file, the program then executes the following three functions:

a) Offline – Formation of the list of all trains’ data: taking 0.04 *s*. Note that this is an interactive train, i.e., there are some other trains which have interaction with this train and thus the safety concept has to be observed.

b) Offline – Computation of the switching points: taking 0.01 *s*. Some of the switching points are shown in Figure 17 which illustrates a cut of the maximal allowable velocities due to intervals, corresponding to the cut of the maximal allowable velocities given in Figure 16.
c) Online – Simulation of the network: taking about 0.24 *s*. A cut of the velocity-path diagram is depicted in Figure 18, while the whole time-path diagram is demonstrated in Figure 19.

The *total* simulation time is 0.29 *s*. We have observed that the major part of the time spent on online simulation, which itself constitutes the major part of the total simulation time, is devoted to checking the other lists so as to observe the safety concept. This has motivated the introduction of the local and global step-sizes, as discussed earlier. (The results in this example have not yet used this step-size strategy.)

VIII. Concluding Remarks and Future Work

We conclude this paper by commenting on the contributions and future work, respectively in the proceeding subsections.

A. Concluding Remark

A constrained state-space model for train traffic in a network like that of the German railway has been developed. The model has been obtained by transforming or, more exactly, reducing the directed graph of the network to some parallel lists, along with a simple relevance between them representing the interaction and schedule. The model is thus a decentralized one, subject to some dynamic constraints called the simple and extended blocking and unblocking conditions representing the above-mentioned relevance. Based on this model, the theoretical essentials for time-optimal traffic have been given. The construction of an operating system for existing networks by the way of this model has been discussed. The simulation of the network has then been examined in full details.

It is clear that the proposed approach yields faster simulation than it could be obtained by a complete integration of the system and that it has been argued that it actually achieves faster-than-real-time simulation, but this can only be verified when the system is implemented in real operating centers. Nevertheless, the following statements can already be made about the simulation time of large networks. As a rule of thumb, on a sequential processor (i.e., a one-processor machine) the simulation time is proportional to: a) the number of interactions, b) the traveling distance, c) the number of interactive trains, and d) the number of active-interactive trains. In a typical large-scale network (as that in Example 7.3) with 2000 trains on the average traveling distance of 500 km, the number of interactive trains is about 300 and that of active-interactive ones about 100-200, i.e., on the average only 100-200 (of the 300 interactive) trains are interacting all over the time. Thus, the simulation time is expected to be on the order of $(100-200/1)\times(500/80)\times 0.29\,\mathrm{s} \approx 3-6\,\mathrm{min}$, and certainly *less than* $(300/1)\times(500/80)\times 0.29\,\mathrm{s} \approx 9\,\mathrm{min}$. Thus, on parallel processors the simulation time will be even shorter. On the other hand, we are aware that the simulation time is inversely proportional to: e) the local and global step-sizes, f) the accuracy tolerance ε in algorithms 4.1.1 and 4.3.1, g) the interpolation error in the Correction box in Figure 5, and h) the speed and RAM of the machine. This goes beyond the objective and scope of this paper and is thus not further followed here.

It should be highlighted that in the first glance a single train has no interaction with others and thus the above-given proportions (for the estimation of the simulation time of large-scale networks) seem unjustified. However, they are justified and are in fact correct if we note that Example 7.3 is about an interactive train and thus at every safety block the relation to other potential trains is checked in order to observe the safety concept, inherently tantamount to the inclusion of interaction. Remarkably, this is the advantage of transforming or rather reducing the graph of the network to some parallel lists.

To highlight the effectiveness of the advocated methodology, it is worth contrasting an actual traveling time with a simulation time. For instance, considering the maximal allowable velocity of 250 km, since it will not be over the whole paths, the above-described large-scale network takes certainly *much more than* $500/250 = 2\,\text{hrs}$ to travel (say $4\ \text{hrs}$), whereas the estimated simulation time is *less than* $9\,\text{min}$.

The advocated modeling, analysis, synthesis, and simulation methodology has the following distinctions: i) it is simple and circumvents the mathematical complexity of the problem, ii) all the features of the existing network are completely addressed by this model, iii) it is generic and can be used to establish an operating system for new networks, iv) some flexibility as well as some robustness can be introduced to the operation of the system, v) it would result in faster-than-real-time simulation, vi) it provides a quantitative estimate of the simulation time, and vii) it enables and facilitates real-time control and decision making in large-scale networks. These last three features – rendered by the proposed model – are the main contribution of this work, sharply distinguishing it from the existing methods by its uniqueness.

We close this part by a discussion about the efficiency of the approach. The natural definition of efficiency is,

$$\eta = \left(1 - \frac{\text{simulation time}}{\text{actual time}}\right) \times 100 \quad . \tag{13}$$

It is clear that e.g. the smaller the number of interactions, the smaller both the simulation time and the actual time. This is true for all the aforementioned factors a)-d) to which the simulation time is proportional. Consequently, it is not clear how a)-d) affect the efficiency, and this may be firmly characterized only when the system is implemented in dispatching centers. On the other hand, the abovementioned factors e)-h) affect the simulation time only, and not the actual time. Hence, because of the inverse relation, the larger any of these factors the higher the efficiency. A precise characterization of this can be obtained after the implementation of the system. Finally, we are aware that synchronization of parallel processors and other implementation-related matters may adversely affect the efficiency. This can also be firmly characterized only after the implementation of the system.

What is of both practical and theoretical significance is that – for a given network, schedule, and simulation settings (factors e)-h)) – the simulation time has an upper bound and the actual time has a lower bound and thus the efficiency has a lower bound. To manifest the contribution of the work it is worth evaluating this lower bound for the above-described network. On this network the lower bound of the efficiency is $96.25\,\%$. In practice, the efficiency is certainly higher than this, since: i) the simulation time will be less than $9\ \text{min}$, ii) the actual time may be more than $3\ \text{hrs}$ (for different reasons). With the simulation time of $3\text{–}6\ \text{min}$ and the same actual time the efficiency will be $97.50\text{–}98.75\,\%$. If the actual time is more, the efficiency will be higher. It should also be noted that on parallel processors all the above-given efficiencies will be higher.

B Future Work

An industrial simulation software is under development; it would greatly facilitate real-time control of the network by dispatchers, since they can observe the would-be result of their prescription beforehand. This software also has another usage, since it may be employed to introduce some additional trains into the existing network with no/minimum interaction with the existing ones.

In case of some deviation from the planned schedule, the existing real-time control of the network is in effect some ad hoc supervisory control, i.e., the feedback loop is *manually* closed by dispatchers using some heuristic tables, etc. A faster-than-real-time simulation software thus would

greatly facilitate their task, since they can observe the would-be result of their prescription beforehand. It is under investigation how to close the feedback loop theoretically. This is a sophisticated multi-rate hybrid output-feedback control problem exacerbated with uncertainty: the only measurements in the German railway network are some asynchronous uncertain position measurements; there is no velocity measurement. This is a major direction for future research.

Moreover, in the first step of modeling some simplifications are done. Consequently, the advocated model is an approximate one. To measure its validity/accuracy we need to have some real data to do a *model validation*, see any classic paper or textbook on identification e.g. [28]-[32]. However, we cannot do this at this stage, since we do not have all the real data (as stated in the previous paragraph). On the other hand, even if we measured and gathered real data, there still would remain the fact that we have *supposed* a bang-bang control by drivers. That is, we cannot guarantee that these data are related to an exact bang-bang control. Consequently, accuracy of the model cannot be firmly considered. Nevertheless, it is possible to attack this issue in a probabilistic framework; this is a sophisticated problem and is another major topic for future work.

Last but not least, this work gives rise to the enticing question whether it is possible to apply the proposed methodology – i.e., transforming or, more exactly, reducing the directed graph of the network to some parallel lists – to other large-scale networks? Expert knowledge of large-scale networks in question is necessary to this end.

REFERENCES

[1] I. A. Ansis, A. V. Dmitruk, and N. P. Osmolovsky, "Solution of the problem of the energetically optimal control of the motion of a train by the maximum principle," *USSR Comput. Math. Phys.*, vol. 25, pp. 37-44, 1985 (in Russian).

[2] F. Arciniegas, *C++ XML*, IN: New Riders, 2002.

[3] L. A Baranov, Ed., *Microprocessing systems for train control*, Moscow: Transport, 1990.

[4] Y. Bavafa-Toosi, C. Blendinger, V. Mehrmann, H. Ohmori, A. Steinbrecher, and R. Unger, "Time-optimal train traffic in large networks based on a new model," *Preprints of the 10th IFAC/IFORS/IMACS/IFIP Symposium on Large Scale Systems: Theory and Applications*, Osaka, Japan, vol. 2, pp. 729-734, July 2004.

[5] C. Blendinger, V. Mehrmann, A. Steinbrecher, and R. Unger, "*Numerical simulation of train traffic in large networks via time-optimal control*," Technical Report No. 721, Department of Mathematics, Technical University of Berlin, Berlin, FRG, 2001.

[6] C. S. Chang and S. S. Sim, "Optimizing train movement through coast control using genetic algorithms," *IEE Proceedings-B Electric Power Applications*, vol. 144, no. 1, pp. 65-73, 1997.

[7] J. Cheng and P. Howlett, "Application of critical velocities to the minimization of fuel consumption in the control of trains," *Automatica*, vol. 28, no. 1, pp. 165-169, 1992.

[8] J. E. Cury, F .A. C. Gomide, and M. J. Mendes, "A methodology for generation of optimal schedules for an underground railway system," *IEEE Transactions on Automatic Control*, vol. AC-25, no. 2, pp. 217-222, 1980.

[9] S. E. Eddy, *XML in plain English*, NY: Hungry Minds, Inc., 1998.

[10] C. Grossmann and J. Terno, *Numerik der Optimierung*, Stuttgart-Leipzig: Teubner, 1997.

[11] P. Gruber and M. M. Bayoumi, "Suboptimal control strategies for multilocomotive powered trains," *IEEE Transactions on Automatic Control*, vol. AC-27, no. 3, pp. 536-546, 1982.

[12] R. F. Hartl, S. P. Sethi, and R. G. Vickson, "A survey on the maximum principles for optimal control problems with state constraints," *SIAM Review*, vol. 37, no. 2, pp. 181-218, 1995.

[13] H. H. Hoang, M. P. Polis, and A. Haurie, "Reducing energy consumption through trajectory optimization for a metro network," *IEEE Transactions on Automatic Control*, vol. AC-20, no. 5, pp. 590-595, 1975.

[14] L. M. Hocking, *Optimal control: An Introduction to the theory with applications*, Oxford: Clarendon Press, 1991.

[15] I. Horowitz, *Synthesis of feedback control systems*, NY: Academic Press, 1963.

[16] P. Howlett, “An optimal strategy for the control of a train,” *Journal of Australian Mathematics Society, Series B*, vol. 31, no. 4, pp. 454-471, 1990.
[17] P. Howlett and P. J. Pudney, *Energy-efficient train control*, Berlin: Springer-Verlag, 1995.
[18] S.-J. Huang, “Fuzzy control of automatic train operation system,” *International Journal of Modelling and Simulation*, vol. 12, no. 7, pp. 321-327, 1997.
[19] E. Khmelnitsky, “On an optimal control problem of train operation,” *IEEE Transactions on Automatic Control*, vol. 45, no. 7, pp. 1257-1266, 2000.
[20] P. J. McLane, L. E. Peppard, and K. K. Sundareswaran, “Decentralized feedback control for the brakeless operation of multilocomotive powered trains,” *IEEE Transactions on Automatic Control*, vol. 21, no.3, pp. 358-36, 1976.
[21] J. L. Meriam and L. G. Kraige, *Engineering mechanics: Statics and dynamics, 3rd*.ed. NY: John Wiley & Sons, 1992.
[22] J. Pachl, *Systemtechnik des Schienenverkehrs*, Stuttgart-Leipzig: Teubner, 1999 (in German).
[23] P. J. Pudney and P. Howlett, “Optimal train strategies for a train journey with speed limits,” *Journal of Australian Mathematics Society, Series B*, vol. 36, no. 1, pp. 38-49, 1994.
[24] R. Unger, “*Numerische Simulation von Zugfahrten unter Realbedingungen*,” Diplomarbeit, Department of Mathematics, Technical University of Chemnitz, Chemnitz, FRG, Mar. 2001 (in German).
[25] S. Seidel, “*Numerische Simulation und Optimierung von Zugfahrten in realen Bahnnetzen*,” Diplomarbeit, Department of Mathematics, Technical University of Berlin, Berlin, FRG, July 2004 (in German).
[26] S. N. Talukdar and R. L. Koo, “Multiobjective trajectory optimization for electric trains,” *IEEE Transactions on Automatic Control*, vol. AC-24, no. 6, pp. 888-893, 1979.
[27] V. Van Breusegem, G. Campion, and G. Bastin, “Traffic modeling and state feedback control for metro lines,” *IEEE Transactions on Automatic Control*, vol. 36, no. 7, pp. 770-784, 1991.
[28] P. C. Young, “Parameter estimation for continuous-time models: A survey,” *Automatica*, vol. 17, pp. 23-29, 1981.
[29] Unbehauen and G. P. Rao, *Identification of continuous-time systems*. Amsterdam: North-Holland, 1987.
[30] T. Soderstrom and P. Stoica, *System identification*. Helmel Hempstead, UK: Prentice-Hall, 1988.
[31] L. Ljung and S. Gunnarsson, “Adaptation and tracking in system identification – A survey,” *Automatica*, vol. 26, pp. 7-21, 1990.
[32] Johansson, *System modeling and identification*. NJ: Prentice-Hall, 1992.

BIOGRAPHIES

Yazdan Bavafa-Toosi received BEng and MEng degrees in electrical power and control engineering from Ferdowsi University of Mashhad and K.N. Toosi University of Technology, Iran, in 1997 and 2000, respectively. Between 2000 and 2004 he held various positions in many places, ranging from an English Teacher to a University Lecturer. He obtained his PhD degree in system design engineering from Keio University, Japan, in 2006. His primary research interests include systems and control theory and applications. Currently, he is self-employed.

Christoph Blendinger received his Ph.D. in 1996 at the University of Bonn, Germany, on a topic concerning theory and numerics of nonlinear partial differential equations. Since then he is working for the Deutsche Bahn Company (DB Systems, Frankfurt) and is engaged in developing a new generation of dispatching systems.

Volker Mehrmann received a Diploma in Mathematics in 1979, his Ph.D. in 1982 and his Habilitation in 1987 from the University of Bielefeld, Germany. He spent research years at Kent State University 1979-1980, University of Wisconsin 1984-1985, IBM Research Center Heidelberg 1988-1989. After spending two years 1990-1992 as visiting full professor at the RWTH Aachen he was a full professor at TU Chemnitz from 1993 to 2000. Since then he is a full professor at TU Berlin. His research interests

are in the areas of numerical mathematics/scientific computing, applied and numerical linear algebra, control theory as well as differential algebraic equations.

Andreas Steinbrecher received a Diploma as Dipl. Math. Tech. (Mathematical Engineering) in 1998 from the TU Chemnitz, Germany. After spending two years 1998-2000 as scientific assistant at TU Chemnitz he has been a scientific assistant at TU Berlin. He received his Ph.D. in 2006 from TU Berlin. His research interests are in the areas of numerical mathematics/scientific computing, numerical integration of differential algebraic equations, in particular, numerical simulation of multibody systems.

Roman Unger received his Diploma as Dipl. Math. Tech. (Mathematical Engineering) in 2001, from the TU Chemnitz with a thesis on Numerical Simulation of train traffic under real conditions. Since then he is a Ph.D. student at TU Chemnitz. His research interests are in numerical mathematics, in particular finite element methods for partial differential equations and scientific computing.

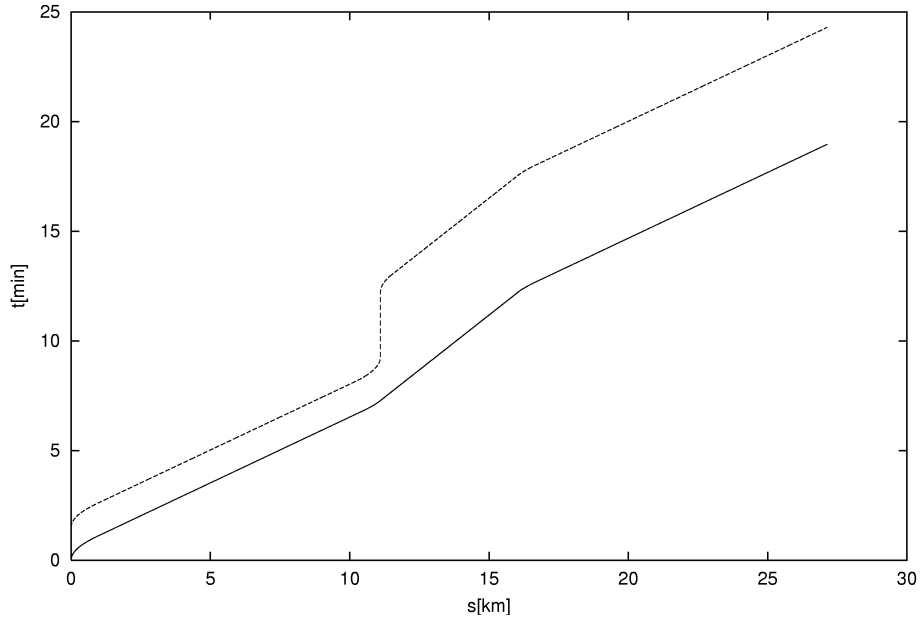

Fig. 8: Example 7.2, scenario 1, time-path diagram, train 1: solid, train 2: dotted

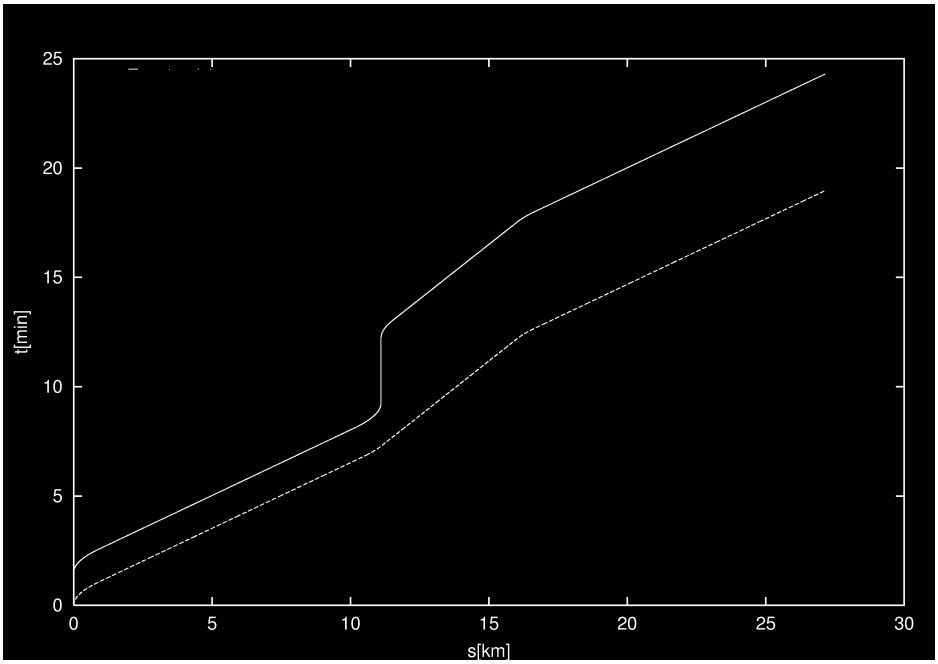

Fig. 9: Example 7.2, scenario 2, time-path diagram, train 1: solid, train 2: dotted

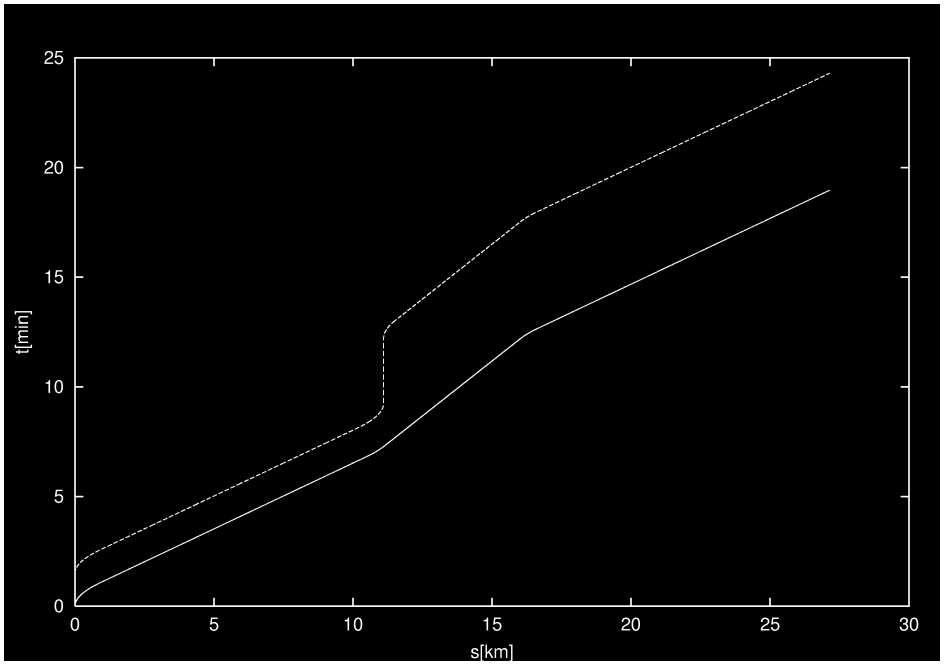

Fig. 10: Example 7.2, scenario 3, time-path diagram, train 1: solid, train 2: dotted

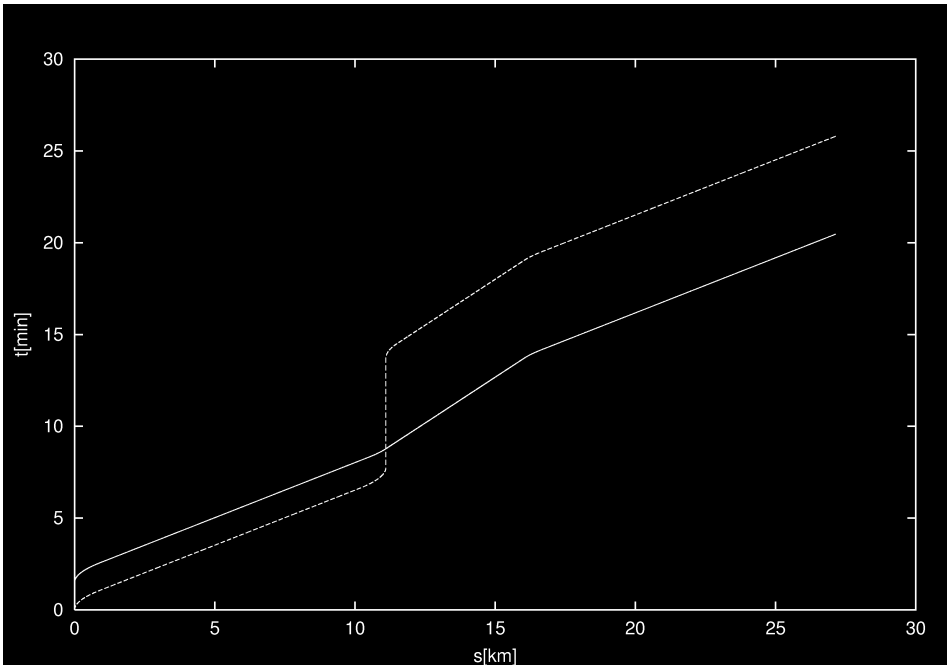

Fig. 11: Example 7.2, scenario 4, time-path diagram, train 1: solid, train 2: dotted

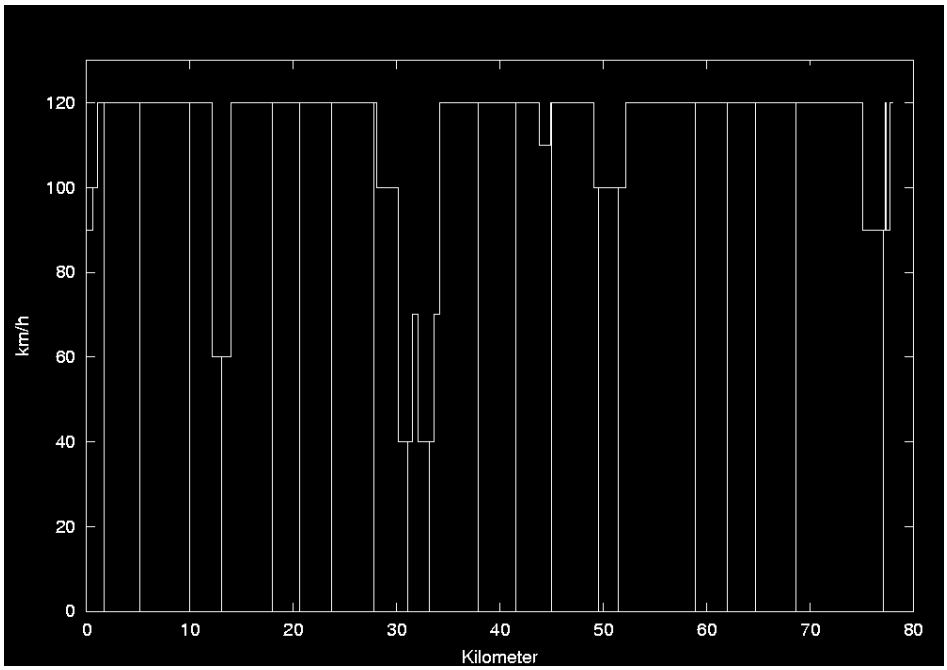

Fig. 12: Example 7.3, the given $v_{\max}$ (similar to Figure 3.a)

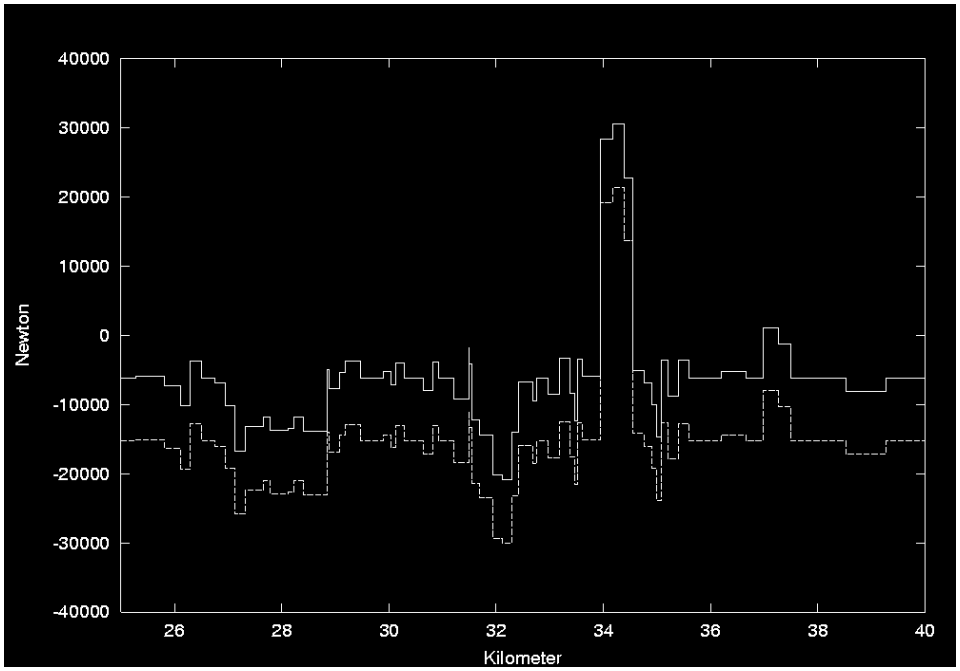

Fig. 13: Example 7.3, uncontrollable part of the control input u_{uc}

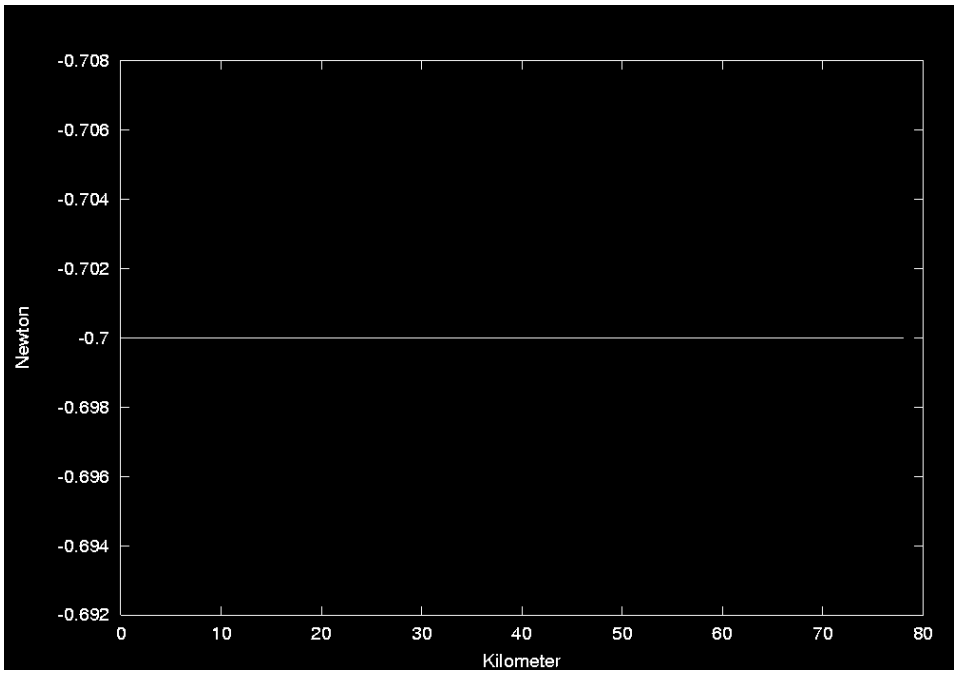

Fig. 14: Example 7.3, full braking force

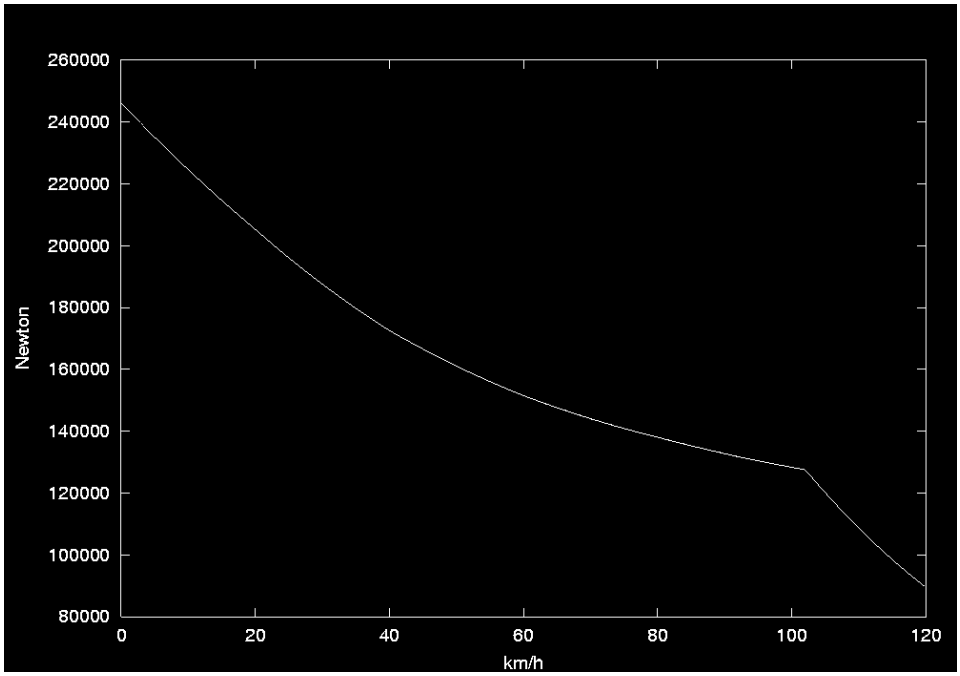

Fig. 15: Example 7.3, maximal thrust $u_{\max}(v(t)) = P_{\max}/v(t)$

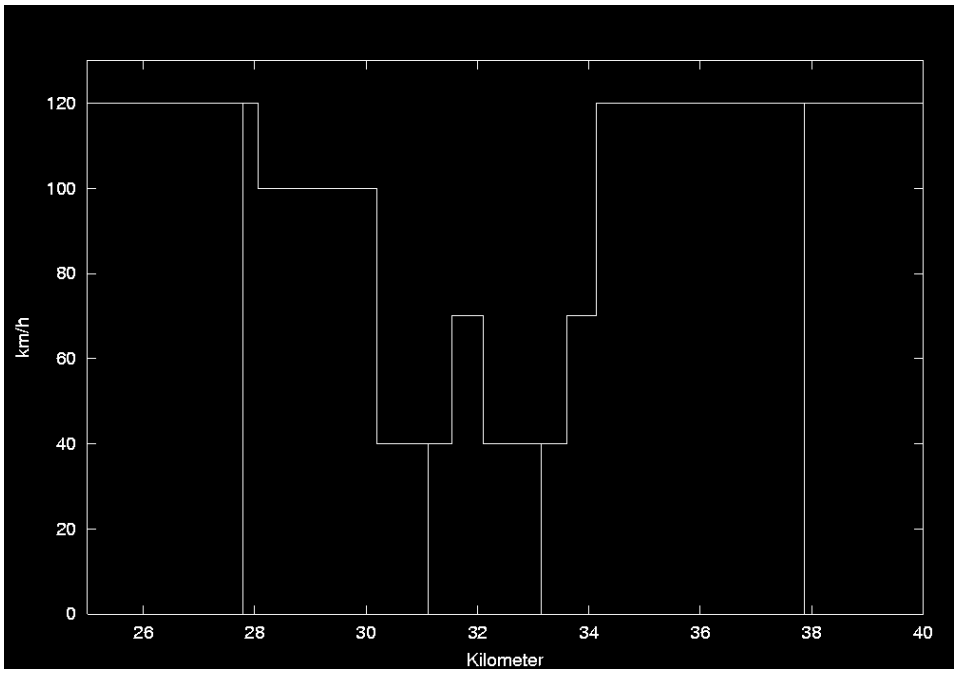

Fig. 16: Example 7.3, cut of the given $v_{\max}$

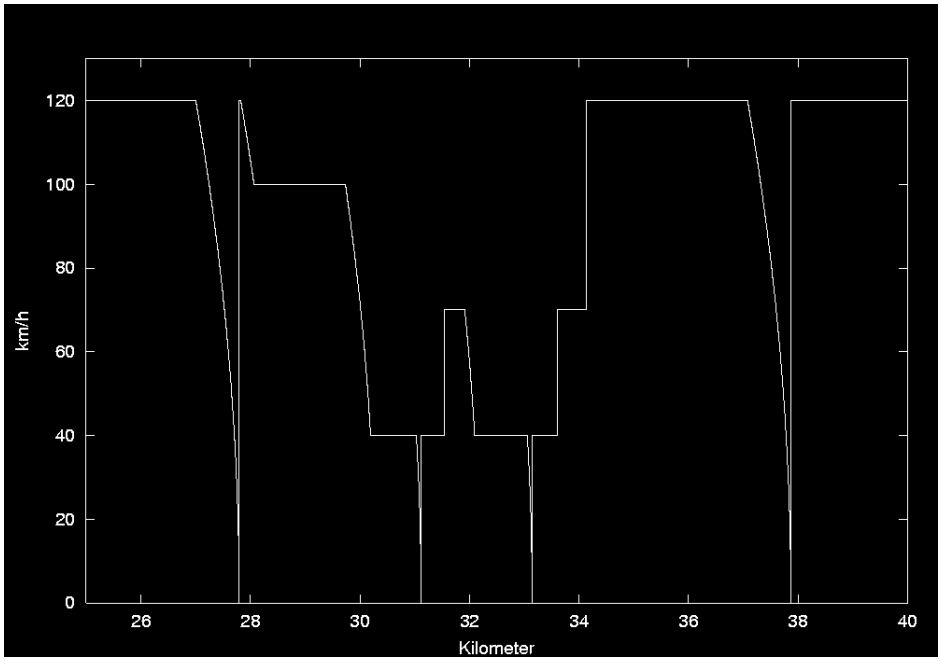

Fig. 17: Example 7.3, cut of $v_{\max}$ due to intervals (similar to Figure 3.b)

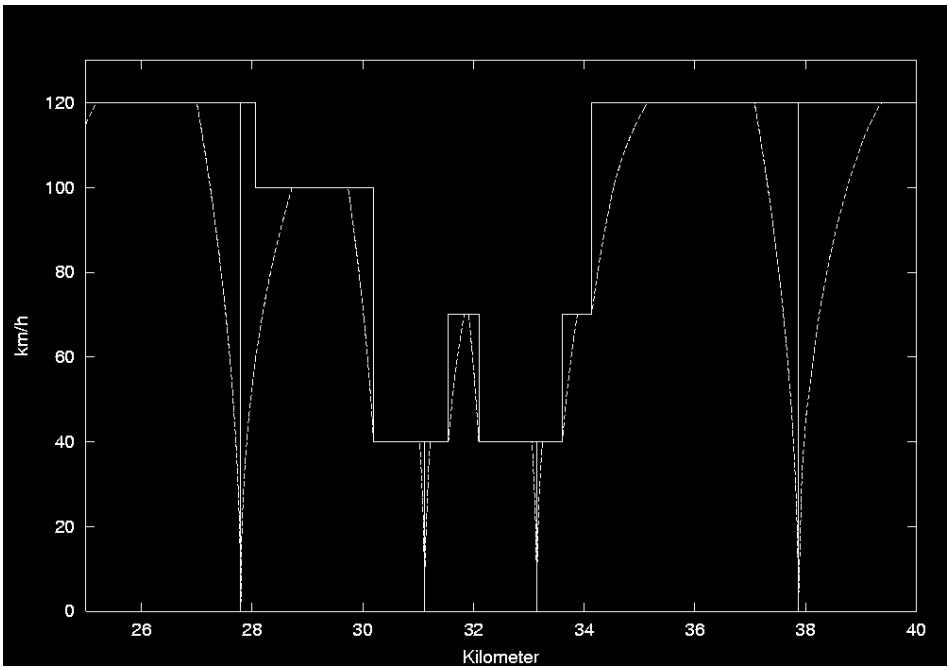

Fig. 18: Example 7.3, cut of the realized velocity-path diagram

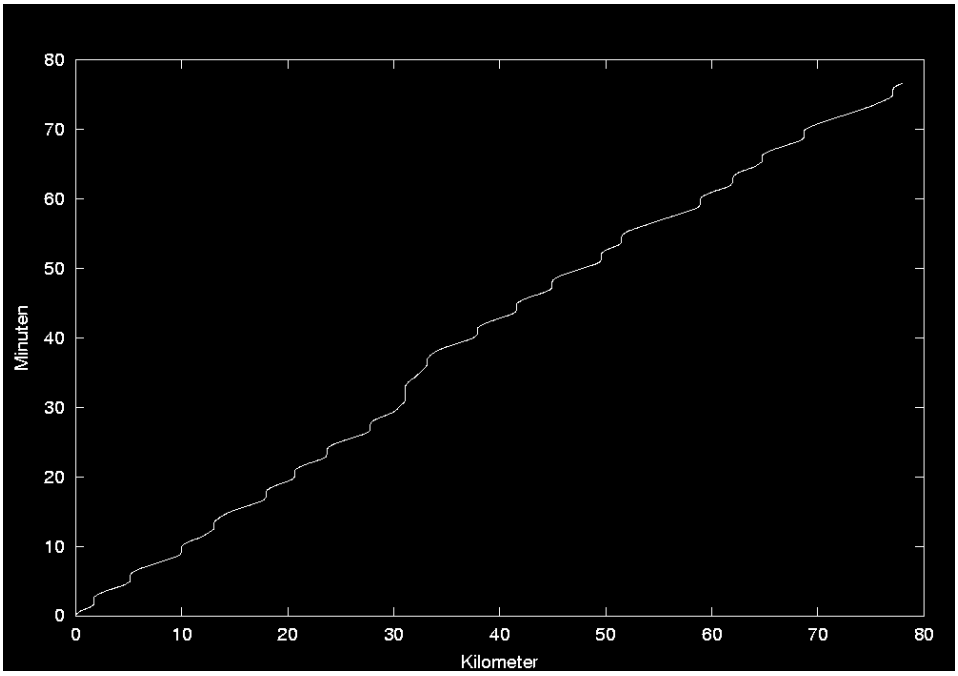

Fig. 19: Example 7.3, the realized time-path diagram