An online railway traffic prediction model

Pavle Kecman, Rob M.P. Goverde
Delft University of Technology, Department of Transport and Planning
Stevinweg 1, 2628 CN Delft, The Netherlands
e-mail: {p.kecman, r.m.p.goverde}@tudelft.nl

Abstract
Prediction of train positions in time and space is required for traffic control and passenger information. However, in practice only the last measured train delays are known and dispatchers must predict the arrival times of trains without adequate computer support. This paper presents a real-time tool for continuous online prediction of train traffic using a timed event graph that captures all scheduled events and precedence relations between them, such as train runs and stops, connections, and minimum headways. Robust estimates for the minimum process times are derived online by computing small percentiles (conditional on current delay where relevant) for conflict-free running times that are obtained by pre-processing historical train describer data. The graph is updated regularly when new information becomes available on train positions or traffic control decisions. The realization times of all events in the graph are predicted considering the usage of running time supplements and buffer times, as well as time loss due to route conflicts based on a conflict detection scheme within the prediction algorithm. The model is demonstrated on the busy corridor The Hague – Rotterdam in the Netherlands.

Keywords
Railway traffic, Train describers, Monitoring, Timed event graph, Prediction

1 Introduction

Real-time prediction of train positions in time and space is a basic requirement for effective route setting, traffic control, rescheduling, and passenger information. However, in practice only the last measured train delays are known in the traffic control centres and dispatchers must predict the arrival times of trains using experience only, without adequate computer support. This often results in simple extrapolation of the current delays for the expected arrival delays. Some railways use a linear shift of the timetable to extrapolate the current delays to the future. This method neglects the fact that some trains may (partially) recover from a delay using running time supplements, while others may get (more) delayed due to route conflicts. Better predictions could be obtained by microscopic simulation models but excessive computation times still prohibit online application of microscopic simulation to densely operated large-scale networks.

This paper presents a real-time model for continuous online prediction of train traffic using process mining, a method of analysing and extracting information about processes from event data records using a process model [23]. The method is applied to Dutch train describer log files.

Train describer systems keep track of train positions in discrete steps over its route, based on train numbers and messages received from elements of the signalling and interlocking systems (sections, switches and signals). One of the tasks of train describers is logging the incoming infrastructure element messages and the generated train number messages, resulting in chronologically ordered lists of infrastructure and train number messages.

We use track occupation data to determine the dependency of running and dwell times on departure and arrival delays, respectively. The data are classified separately for each train line, thus reflecting the different stopping patterns and rolling-stock types. These dependencies, together with the actual route plan, timetable and current positions of all trains, are used to predict the future running times (on block section level) and dwell times of all trains. Microscopic operational constraints are incorporated in predictions, therefore capturing all train interactions due to capacity or connection constraints. When an update of the train positions becomes available, the predictions are recomputed.

The predictive traffic model supports route setting and traffic control decisions and could be interactively used by signallers and traffic controllers. First, the model predicts route conflicts for a given actual route plan and train positions. This could be used by the signaller to pro-actively resolve the conflict by e.g. changing routes or the order of trains. The impact of any control decision can be checked by an update of the predictive model leading to new conflict and arrival time predictions. If a control decision leads to satisfying results it can be implemented in the actual process plan. If on the other hand a route conflict cannot be avoided, the signaller could give speed advice (or new target passage times) to the relevant train drivers so that the impact of the route conflict is minimal and energy can be saved by preventing unnecessary braking and re-acceleration. Second, the arrival time predictions could be used to check connections in the case of arrival delays. When a connection conflict is detected, the signaller may decide to secure or cancel a connection in advance. This way up-to-date passenger information can be provided, both at stations and in the delayed trains. Similarly, endangered logistic connections (crew or rolling-stock) can be predicted in advance.

The next section gives an overview of relevant approaches in current literature, Section 3 describes processing of train describer data, and Section 4 gives a detailed description of the model and its components. Section 5 presents the performance of the tool when applied in a simulated real-time environment. Finally, Section 6 summarises the presented model and gives guidelines for further research and improvements.

2 Literature review

This section gives an overview of recent experiences in the field of railway data mining and different approaches to railway traffic prediction in the literature.

A description of train describers and an overview of different systems used in Europe is given by Exer [7]. Train describer data recently became an important source of information for analysing railway traffic. Medeossi *et al.* [16] used track occupation data along with train event data recorded on-board to calibrate train motion equation parameters in the process of computing stochastic blocking times for individual trains. This model has recently been extended with stochastic estimates of initial delays and dwell times in order to build a simulation tool for an *a priori* evaluation of the impacts that new timetables or infrastructure improvements may have on railway operations [15].

Daamen *et al.* [4] and Goverde *et al.* [9] described the algorithms for automatic route

conflict identification based on data records of the Dutch train describer system TNV, which were implemented in the tool *TNV-Conflict*. The tool was further extended with an add-on, *TNV-Statistics* [10] for detailed statistical analysis of train realization data based on the output files of *TNV-Conflict*.

The TNV system was replaced by the new train describer system TROTS which contains an essential new approach to train number steps. Kecman & Goverde [13] presented a tool for recovering realised train paths and automatic conflict identification based on process mining of TROTS log files. Signal passage and section occupation data are stored in tabular output which is used in this paper for developing robust estimates of running and dwell times in the prediction model.

The prediction, model introduced in this paper, is based on an acyclic timed event graph. Goverde [8] introduced a macroscopic model for train delay propagation based on timed event graphs and max-plus algebra that allows application of fast algorithms for computation of delay propagation in a short time even for large networks. Due to the fixed structure of train orders and routes in the model, it is not directly suitable for real-time predictions that need to consider current conditions on the network and changes in the actual process plans.

Berger *et al.* [1] incorporated stochasticity in their graph-based macroscopic model for delay prediction. By using a set of waiting policies for passenger connections and assuming discrete distributions of process times, they are able to predict delay propagation over the network. In another approach, Büker & Seybold [2] modelled delays as random variables, described with suitable distribution functions, and applied analytical methods to compute delay propagation in a mesoscopic graph-based model. However, the large-scale character of the models does not allow precise modelling of train interactions and the resulting variability in running times.

A microscopic graph model, presented by D'Ariano *et al.* [5], considers the majority of operational constraints of railway traffic. Static arc weights require an iterative approach to recompute feasible speed profiles of trains based on train dynamics and detailed infrastructure data. Whereas the train interactions on open track are accurately modelled, detailed modelling of route setting principles in complex station areas is not considered. Corman *et al.* [3] performed a detailed analysis of necessary granularity in modelling train traffic in interlocking and station areas. In this work, we rely on traffic realization data to develop a fully data-driven model based on actually realized minimum headway times between conflicting routes.

The prediction model presented in this paper considers dynamic, online computation of arc weights. This concept of time-dependent graphs has been used in railway related applications mainly for supporting timetable queries [17]. While such applications are focused on developing fast algorithms for dynamic shortest path computation, in this paper we employ the depth-first search based prediction algorithm to obtain the predicted realization times of all events in the graph.

Microscopic simulation tools such as *OpenTrack* [18] or *RailSys* [20] are able to give accurate predictions of running times, and possible route conflicts and resulting delay propagation. Due to a high level of detail in modelling infrastructure and train dynamics, such models are not suitable for real-time applications on large and heavily utilised networks. Moreover, they do not incorporate peculiarities of train traffic such as variability of dwell times due to delays or peak hours, which may affect train traffic to a great extent.

Hansen *et al.* [11] presented a macroscopic model for prediction of train running times

using historical track occupation data. The dependencies of dwell times as well as running times on the level of open track sections (line segment between two stations) were captured and used to compute robust estimates of arrival and departure times. We extend this approach to the microscopic level in order to accurately model train behaviour and interactions on open track sections.

An on-line prediction tool similar to the one presented in this paper has been implemented in the Swiss traffic control system RCS-DISPO [6]. The main part of this tool is a microscopic model based on a directed acyclic graph with arc weights that are computed using train motion equations, with respect to detailed description of infrastructure and train dynamics. The tool is successfully applied to a large number of trains (between 900 and 2000). The authors concluded that the accuracy of predictions depend on the time interval between the moment of computation and predicted event. Prediction errors smaller than 1 minute were obtained for events within 20 minutes prediction horizon.

We attempt to improve this approach by computing the arc weights using historical data, thus incorporating the dependence of process times on current delays, with possible extensions that include influence of peak hours, weather, and rolling-stock type. By applying a fully data-driven approach in contrast to deterministic computation of process times, we aim to capture variability in running, dwell and headway times. The prediction window in our model is limited to 20-30 minutes, comprising the events of the trains currently running on the network with routes and operational plans determined by the traffic control centre. The graph structure of the model can conveniently be integrated with a network-wide delay propagation model [8], thus propagating the current conditions (or potential decisions by the traffic controllers) to subsequent periods in the timetable.

3 Processing of TROTS data

In the Dutch train describer system TROTS, the train steps are recorded on the level of track sections (a block section consists of one or more track sections), with a message when a new track section is occupied by a train and when a track section is released by a train. Moreover, train number step messages are coupled to track section messages.

The Dutch railway network is divided into multiple TROTS areas. Each area comprises one or more major station areas with complex topologies and 30 – 40 km of surrounding railway infrastructure. In order to reconstruct the train traffic over multiple TROTS areas it is necessary to merge the corresponding log files. TROTS log files are archived per day and area in large files of ASCII format of approximately 75 MB.

Infrastructure messages contain the following information: time stamp, event code, element type (section, signal, point), element name, and new state (‘occupied’/‘released’, ‘stop’/‘go’, ‘left’/‘right’). The train number step messages contain amongst others a time stamp, event code, train number, and a sequence of all occupied track sections. Each successive train number step message relates either to a new occupied track section at the front or to a released track section at the rear. The event code of a train number step corresponds to a section message with the same event code. This coding is used to match a message about a section occupation or release with a message of a train number step.

3.1 Process mining approach

Blocking time theory [12] provides the logic for building the process model from the log file. Signal passages are events that initiate processes such as blocking a part of the infrastructure and running over a block. Each complete train run can thus be represented as a graph, built on-line by sweeping through the file. Moreover, route conflicts can be identified simultaneously by determining the time difference between relevant events and verifying if the train separation principles are respected.

Due to the large size of TROTS log files, it is necessary to build an algorithm that sweeps through the file and visits every line only once, thus avoiding long computation times. An object-oriented approach is used to store the relevant data from the log files in infrastructure and train number objects which enables the algorithm to revisit the objects, and use and update the information therein [4].

Figure 1 shows the attributes of each block and train object. All objects are created and updated on the fly while sweeping through a TROTS log file. Static attributes in block objects ('Start signal', 'End signal' and 'Sections') are fixed when objects are created, using additional infrastructure files. They provide a unique description of a block or an interlocking route with the start and end signals, and comprised track sections.

Block
Start signal
End signal
Sections
Trains: {t_{occ},t_{rel},*series*}

Train
Number
Timetable
Sections:{name,t_{occ},t_{rel}}
Signals:{name,t_{pass}}

Figure 1: Objects and their attributes

Dynamic attribute 'Trains' in a block object contains a chronologically sorted list of trains that traversed the block. Information about the block occupation time, release time and train series is stored for each train.

A Train object is defined by a train number and the list of traversed sections and signals, that are updated with every message from the log file related to the train. A list of scheduled departure and arrival times at each station is given in the attribute 'Timetable'.

It is an important feature of the prediction model to capture the interactions of trains and the resulting conflicts and knock-on delays. Consequently, partitioning realized process times data to hindered and unhindered trains is of great importance. The process mining algorithm described by Kecman & Goverde [13], uses blocking time theory as the underlying process model and is thus able to identify all route conflicts. Realised hindered running times are filtered out and excluded from further analysis.

Furthermore, the process mining algorithm estimates the actually realized departure and arrival times of all trains with the largest estimation error smaller than 10 seconds. Therefore, all running (dwell) times can be analysed with respect to the last measured departure (arrival) delay, which will be exploited for computing running time and dwell time predictions in Section 4.2.

4 Online traffic prediction tool

The main components of the tool and the flow of data between them are depicted in Figure 2. The online prediction tool is based on a timed event graph with dynamic arc weights. The graph topology is built and updated based on the actual timetable, route and connection plan, and current positions of trains on the network. We assume that the actual route and connection plans are continuously provided by the traffic control for the next 30 min. Route plan for a train is given as a planned sequence of block sections in the train route. By using the 'Sections' attribute of the block objects (Figure 1), a route plan can be translated to the level of track sections and used to determine the necessary headway arcs for routes with common track sections. Each change of the actual plans or new information from the real-time operations, i.e. changing the relative order of trains, adding or cancelling trains, modifying train routes, updating connections and removing passed events, results in an update of the graph topology.

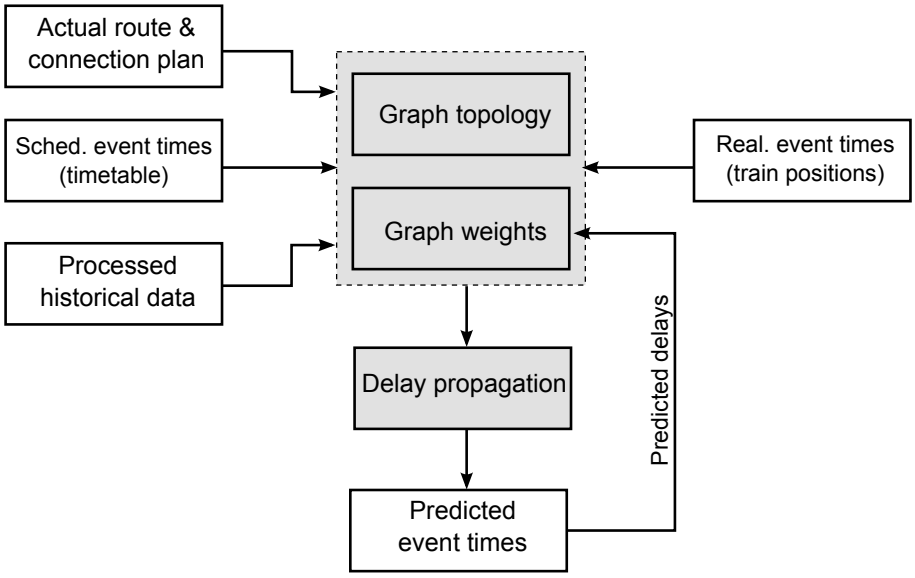

Figure 2: Components of the online prediction tool

Arc weights represent the minimum process times and they are computed based on the current train positions and delays, and processed historical data. The weight of an arc is time-dependent and assigned in a dynamic way depending on the (estimated) starting time of the modelled process. That way the dependence of running and dwell times on current (predicted) delays is incorporated in the model. After every graph update, a prediction of event times of all reachable events is performed.

In the following subsections, the three main components of the tool (shaded boxes in Figure 2 and the input data will be explained in detail.

4.1 Microscopic graph model

The railway traffic is modelled microscopically with a timed event graph (TEG). A TEG is a representation of a discrete-event dynamic system in which events are modeled by nodes and processes by arcs.

We distinguish between signal events (passing of a signal by a running train) and station events (arrival and departure to and from a platform track). Nodes are described by train number, infrastructure element (signal or platform track), type, predicted realisation time, and recorded realisation time (when available). Station event nodes (arrival and departure nodes) also include scheduled event times as attributes. By comparing the recorded (predicted) event times with the scheduled event times, the current (predicted) delay is obtained for a specific train and used to estimate the duration of its subsequent processes (dwell and running times). Scheduled departure times are also used to incorporate the timetable constraints (trains cannot depart before their scheduled departure times).

Apart from modelling the running and dwelling processes related to a specific train, directed arcs are also used to model interactions between trains, namely headway and connection constraints. Arcs are described by starting event, end event, type and weight. Types of the starting and end events determine the type of an arc. For events belonging to the same train, running arcs connect all signal passing events. Dwell arcs connect an arrival event with a subsequent departure event. An inbound running arc connects a signal event with a subsequent arrival event, whereas an outbound running arc connects a departure event with a subsequent signal event.

Connection arcs are introduced for modelling commercial constraints (passenger transfers), or logistic constraints (rolling-stock and crew connections). They connect the arrival event of a feeder train and the departure event of a connecting train in the same station.

Headway arcs separate the successive occupations of an infrastructure element by different trains. Typically, a signal changes to a permissive aspect as soon as all sections in a block (interlocking route of the approaching train) protected by the signal, have been released.

On open tracks and for interlocking routes with the same end signal, the critical section that constrains signal release is the section before the end signal of the block or route. This situation is typical for trains that run over the same block or station route or for trains with merging routes (routes that have different starting signals and the same end signal) in interlocking areas. An accurate train separation is ensured by adding a headway arc that constrains the realisation of a signal passing event of an approaching train until the protected block was cleared by the previous train. The head event is the start signal passing event of the approaching train, the tail event is the end signal passing event of the preceding train and the arc weight is equal to the clearing time of the preceding train increased by the setup and release time of the signalling system.

However, in interlocking areas, conflicting routes are often diverging (with the same starting signal and different end signals) or intersecting (different starting and different end signals). The 'sectional release' route setting principle [22] is designed to increase the capacity in station areas by allowing simultaneous multiple train movements with safety measures insured by the interlocking systems. A route can be set for an approaching train as soon as the last section, which the route has in common with the preceding route, has been released. Since all events in the model are signal passages or station events, the event of the critical section release has not been included. We model the train separation by adding a headway arc between passing events of signals that initiate running processes over the pro-

tected interlocking area. The head event and the tail event are the start signal passing events of the approaching and the preceding train, respectively, and the arc weight is computed as a small percentile of the headways between the same successive train runs from the historical track occupation data (see Section 4.2).

The concepts of train separation based on blocking time theory are implemented in the tool for the purpose of route conflict predictions (Section 4.3).

Figure 3 shows an illustrative example of a timed event graph for two trains. The planned route for each train can be described by the sequence of signals: S_1, S_2, S_3, S_4, and S_6 for train T_1 and S_1, S_2, S_3, S_5, S_6 for train T_2. Every signal passage is modelled as a node. Both trains have a scheduled stop at the station which is modelled with arrival and departure nodes (large nodes in the figure). Nodes belonging to one train run are connected by

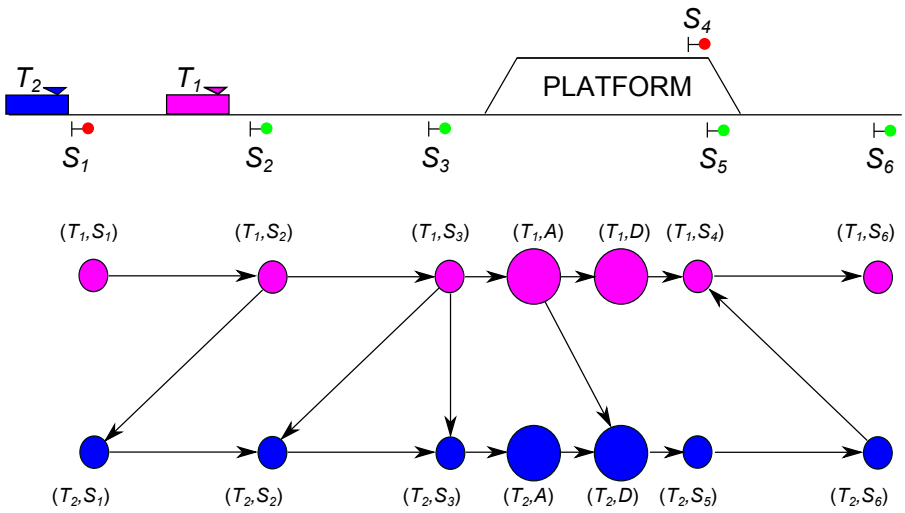

Figure 3: An illustrative example of a microscopic TEG

running and dwell arcs. Since the trains run over the same infrastructure, the necessary minimum headway times are ensured with headway arcs. The route between signals S_1 and S_3 is the same for both trains, thus requiring at least one block separation between trains, which is modelled with headway arcs $(T_1, S_2) \rightarrow (T_2, S_1)$ and $(T_1, S_3) \rightarrow (T_2, S_2)$. The sectional release principle between diverging inbound routes of two trains is enabled with the headway arc $(T_1, S_3) \rightarrow (T_2, S_3)$. Finally, train T_1 can leave the station when the block between S_5 and S_6 has been released by train T_2, which is modelled by the headway arc $(T_2, S_6) \rightarrow (T_1, S_4)$.

A planned connection is secured with the arc between the arrival event of T_1 and the departure event of T_2. Note that the direction of the headway arcs indicate the order of trains. In Figure 3 train T_2 overtakes train T_1 in the station.

The graph topology is continuously updated as the 30 min rolling horizon moves. Possible new trains, planned to operate within the actual horizon, are added to the graph with their planned route on the level of block sections. The necessary headway arcs are built per block between consecutive trains that use at least one same track section covered by the

block. With each update about the train positions (signal passage, departure or arrival of a train), the nodes describing events from the past and their incoming and outgoing arcs are removed from the graph (and stored with the realised event times), thus keeping the size of the graph stable within a certain time interval.

4.2 Computation of arc weights

Arc weights in timed event graphs are equal to the minimum process times of the modelled processes. In order to accurately estimate arc weights, we assume that delayed trains typically run in the full performance regime and have minimal dwell times aiming to use time supplements to (partially) recover from delays. Similarly, trains running on time or ahead of their schedule aim to run in a lower regime to avoid early arrivals and decrease energy consumption. In that context, a time-dependent, dynamic computation of arc weights [17] is added to the timed event graph presented in the previous section. The basic idea behind this approach is that running and dwell times depend on the previously noted delays [11].

Running and dwell arcs
In order to determine the dependence of running times on current delays, the train describer event data were processed with the process mining tool for recovering train paths and infrastructure utilization [13]. The tool provides the running times on the level of block sections classified by train line (trains of the same line operate with the same stopping pattern and usually with the same rolling-stock) and attributed by the current delay noted at the last departure or arrival event. Moreover, the actual arrival and departure times for each scheduled train stop are determined, thus enabling similar records for dwell times, and inbound (from passing the home signal to standstill at the platform) and outbound (from the platform to the exit signal) running times in stations. Running times of hindered trains were identified and filtered out from the data.

Robust regression with the least trimmed squares method [19], resisting 25% of outliers, is used to fit the data and compute the linear dependence of process times on delays. Each block section and station route is attributed with linear coefficients for each train line. We also include the 10^{th} percentile of a process time and use it as the absolute minimum process time to avoid infeasible predictions in case of large delays.

Figure 4 gives an example of the dependence of the running times of train line 2100 over the block between signals GV615 and DTA623, on the delay at the previous departure from The Hague HS. It is visible that in this case there is no clear linear dependence of running time on the departure delay. A possible interpretation is that due to intense capacity consumption, the timetable does not include sufficient running time supplements for trains to compensate for their delays on the corridor between The Hague and Rotterdam [11].

Figure 5 shows the dependence of the dwell time of the 2100 trains in The Hague HS station on arrival delay. The robust linear regression fit results in a coefficient of determination $R^2 = 0.94$. The horizontal line represents the minimum dwell time for passenger activities estimated as the 10^{th} percentile of all realized dwell times $p_{10} = 195$ seconds (the scheduled dwell time for 2100 trains in The Hague HS is 240 seconds). The 10^{th} percentile is included in the estimate in order to avoid unrealistic estimates in case of large delays.

The current data set, consists of 6 days of traffic on the corridor, which after filtering out hindered runs is reduced to about 200 records per train line. The analysis of running and dwell times on the Rotterdam – The Hague corridor showed a strong dependence of dwell

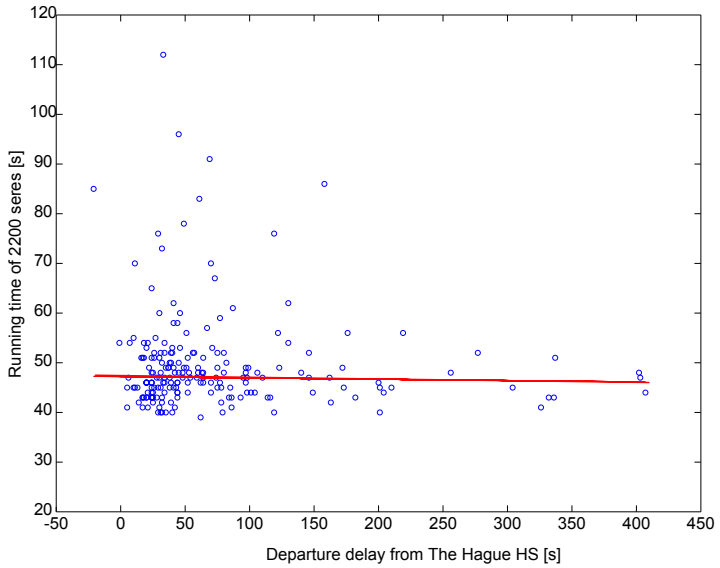

Figure 4: Dependence of running time between GV615 – DTA623 on departure delay from The Hague HS

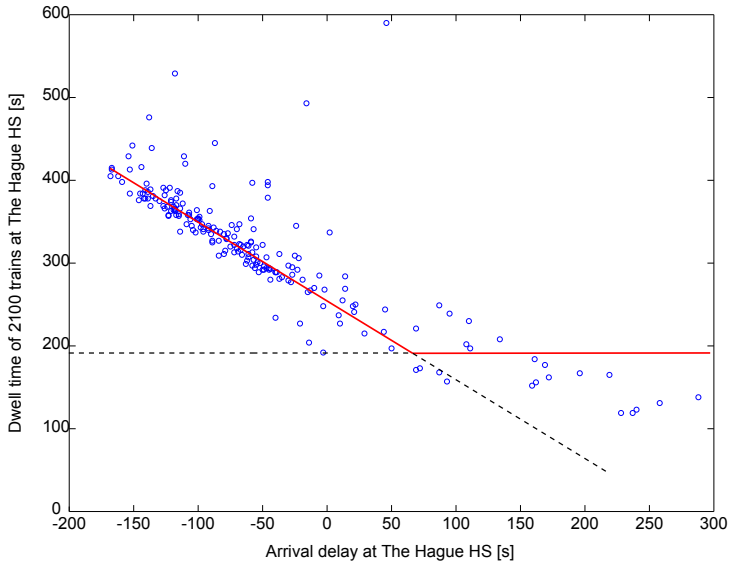

Figure 5: Dependence of dwell times in Den Hague HS on arrival delay

times on arrival delays. Running times show weaker dependence on departure delay from the last station with a scheduled stop. We expect to find stronger dependencies of running times when the model is applied on corridors with more running time supplements.

Headway arcs
The weights of headway arcs represent the minimum headway time between two trains on the same infrastructure element. Minimum headway time between successive block occupations (route settings) equals the sum of running time of the first train, clearing time, and setup and release time of the signalling system [12]. In this paper a constant value of 2 seconds is used for the setup and release time on open track and 12 seconds for route setting time in stations. Clearing time is estimated from the data as the 10^{th} percentile of the clearing times of a block by a specific train line.

In order to model the principle of sectional release using only signal passing events, the minimum headway time between two trains with diverging or intersecting routes is estimated from the data as the 10^{th} percentile of the time headways between train runs of the corresponding train lines from the historical track occupation data. By choosing a small percentile of the realised time headways, the impact of buffer times on minimum headway times estimates is excluded.

Connection arcs
The weight of a connection arc is equal to the minimum transfer time for passenger connections or the time needed to perform activities that enable planned rolling-stock and crew circulations, for logistic connections.

Connection times do not depend on the current delay of trains and the possible effect of delays on headway times was not considered in this work. Therefore, these values are computed offline and the corresponding arc weights are fixed.

4.3 Online prediction of event times

The pseudo code of the online depth-first search based algorithm for prediction of event times over graphs with dynamic arc weights is given in Algorithm 1. A recursive depth-first search is chosen as the method of traversing the graph, due to its low memory requirements, which is an important constraint for large graphs. After each event realisation, the reachable set of nodes is isolated, where the root node is the node that models the realised event. The prediction algorithm then updates the predicted event times of all events in the reachable set. Note that if a node is not reachable, the corresponding event time can in no way be affected by the new information. Therefore, it is not necessary to visit that node in the prediction process.

We model railway traffic with a graph $G = (V, E)$ where V is a set of nodes and E is a set of arcs. A node $v \in V$ is described by $(n(v)$, $\mathit{infra}(v)$, $\mathit{type}(v)$, $\mathit{in}(v)$, $\mathit{out}(v)$, $t_{pred}(v)$, $t_{rec}(v))$, representing the train number, infrastructure element (signal or platform track), type, the set of incoming arcs, the set of outgoing arcs, predicted realisation time and the recorded time (when available), respectively. The nodes that model scheduled events, i.e. arrivals and departures are also attributed with the scheduled event time $t_{sch}(v)$.

For implementation purposes, a vector $t_e(v, v_j)$, containing the earliest possible realisation time of v with respect to each direct predecessor $v_j, j = 1, ..., |\mathit{in}(v)|$, is added to every node. When a new train is added to the graph, the values of t_e are computed for each node

of the train in the topological order. The initial prediction of each event time can than be computed as the maximum of the earliest possible realization times over all incoming arcs $t_{pred}(v) \leftarrow \max(t_e(v, \cdot))$

An arc $e \in E$ is described by ($start(e)$, $end(e)$, $w(e)$, $type(e)$) representing the start event, the end event, the arc weight and the arc type (*dwell, run, headway, connection*). Note that, as explained in Section 4.2, headway and connection arcs have fixed predefined weights, stored in the data structure of processed historical data W. We define a mapping Ω that retrieves the predefined weight of an arc from W.

The weights of running and dwell arcs are determined online with every algorithm execution using the functional dependence of process time on the current train delay. For running and dwell arcs, the start and end events belong to the same train and the infrastructure resource (block or station route) is known. The arc weight is computed by $w(e) = f_{block,line}(z(n))$, where z is a vector that contains the value of last recorded delay for each train number n (note that delays are recorded only at scheduled arrival and departure events). During the algorithm execution, predicted event times of a scheduled event will give predicted delays of trains. Therefore, subsequent process time estimates are computed by $w(e) = f_{block,line}(\hat{z}(n))$, where $\hat{z}$ is the vector of predicted delays for every train number n. Function f is retrieved from W for the appropriate block (station route) and the appropriate train line.

For every *run* and *dwell* process, the 10^{th} percentile of process times for every train line, denoted by $p^{10}_{block,line}$, is computed based on historical data, in order to avoid infeasible estimates of process times for large delays and stored in W.

Finally, every first node in the planned route of a train $v_1(n)$, modelling the entrance time of train n (the first departure or the first event within the observed network), is connected to a dummy node 0 by an arc with weight that is equal to the expected entrance time.

When an update about the realisation of event $v_k \in V, k = 1, ..., |V|$ arrives, the subgraph $G_k = (V_k, E_k)$ is computed. Set V_k comprises all nodes reachable from v_k, and $E_k = \{(v_i, v_j) | \{v_i, v_j\} \in V_k\}$. We set $t_{pred}(v_k) \leftarrow t_{rec}(v_k)$. If $type(v_k) \in \{$*departure, arrival*$\}$, the current delay value of the corresponding train is updated $z(n(v_k)) \leftarrow t_{rec}(v_k) - t_{sch}(v_k)$. The information is further propagated through the graph and predicted event times of all reachable events are computed according to Algorithm 1.

The main loop of the prediction algorithm is initiated in line 4. In lines 5–8, the actual weight of an outgoing arc is computed (or retrieved in case of headway and connection arcs). The earliest realization time of the corresponding successor is updated in line 10. If all constraints on event realization time are known (all direct predecessors were visited and all incoming arcs traversed) the predicted event time is computed in line 12. The timetable constraint for departure events is included in line 15. For all scheduled events, the predicted delay vector is updated in line 16. Finally, in line 17, a recursive call of the algorithm is performed.

The prediction algorithm sweeps through the subraph of reachable nodes G_k passing each arc only once. The running time of Algorithm 1 is therefore $O(E_k)$.

After each event realisation, the graph updated using the procedure 'UpdateGraph' (Algorithm 2). Events of a train occur in a given sequence that reflect a defined route plan. As the first event in a sequence is realised, the corresponding node is removed from the graph along with its incoming and outgoing arcs (lines 3–4) and an arc between node 0 and the next node in the event sequence of a train is added (line 5). The weight of the added arc is equal to the predicted realisation time of the next event of the train.

Algorithm 1 PREDICTEVENTTIMES

Input: G_k, W, z, v_k
Output: $G_k, \hat{z}$
$\hat{z} \leftarrow z$
for all $e \in out(v_k)$ **do**
  if *type*$(e) \in \{$*dwell,run*$\}$ **then**
    $w(e) \leftarrow \max[p^{10}_{block,line}, f_{block,line}(\hat{z}(n(v_i)))]$
  else
    $w(e) \leftarrow \Omega(W, e)$
  $v_j \leftarrow$*end*(e)
  $t_e(v_j, v_k) \leftarrow w(e) + t_{pred}(v_k)$ //update earliest time w.r.t. v_k
  if $|t_e(v_j, \cdot)| = |in(v_j)|$ **then**
    $t_{pred}(v_j) \leftarrow \max(t_e(v_j, \cdot)$ //if all direct predecessors of v_j wre visited
    if *type*$(v_j) \in \{$*arrival,departure*$\}$ **then**
      if *type*(v_j) = *departure* **then**
        $t_{pred}(v_j) \leftarrow \max(t_{pred}(v_j), t_{sch}(v_j))$
      $\hat{z}(n(v_j)) \leftarrow t_{pred}(v_j) - t_{sch}(v_j)$
    PredictEventTimes$(G_k, W, \hat{z}, v_j)$ //recursive call

Algorithm 2 UPDATEGRAPH

Input: $G = (V, E)$, v_k
Output: G
$V \leftarrow V \setminus v_k$ //remove realized node
$E \leftarrow E \setminus \{$*in*$(v_k)$,*out*$(v_k)\}$ //update arc list
$E \leftarrow E \cup (0, v_1(n), t_{pred}(v_1(n)))$

Algorithm 1 enables straightforward identification of connection conflicts. If the critical incoming arc (the arc that actively constraints the earliest realisation time of the predicted event) is a connection arc, then a connection conflict is identified. Note that a critical headway arc indicates a prediction of 'stop' signal aspect before the train.

Prediction of route conflicts as defined by blocking time theory [12] is simple after execution of Algorithm 1 since the estimates of all train speed dependent times (approaching, running, clearing) are known. After including the signal watching, setup and release time, taken as constant values for all trains, the blocking times are determined and a route conflict is identified by the overlapping blocking times.

5 Application of the prediction model

The performance of the model is illustrated on an example of the busy corridor between The Hague and Rotterdam in the Netherlands.

The training set of data for regression analysis consists of train describer log files for six days of traffic in two traffic control areas. While sweeping the files with the process mining and conflict identification tool [13], the dependencies of process times on current delays are computed, as well as the necessary percentiles to model the lower bounds of running, dwelling and headway times, as explained in Section 4.2. The predictions are performed on a separate test set consisting of track occupation data for one day of traffic.

For model validation (and example of application) we simulate the real-time environment by scanning the train describer log file from the separate test set, that contains the chronologically sorted infrastructure and train messages from the two TROTS areas (Rotterdam and The Hague). Traffic control input is included in the form of a list of trains described by the train number, timetable, route plan (block sections) and expected entrance time to the observed part of the network (or the first departure times if the train starts within the observed area).

The selected corridor (Figure 6) and train routes enable testing the model with all possible train interactions. Between station The Hague HS and the junction close to Rijswijk, the trains running towards Rotterdam use two parallel tracks thus leading to merging routes at the junction, where the two tracks merge into one. Diverging routes and corresponding headways are also included, as the inbound routes of local and intercity trains in Rotterdam (RTD) lead to different platforms.

An example of predictions is shown in Figure 7. The presented time-distance diagram shows the predicted train paths (local trains are given in magenta and intercity trains in blue). The realised train paths in space and time are presented with black lines. The prediction is performed at the departure of train ST5025 from The Hague HS (GV). Complete routes of the seven trains that enter the network within the 30 min prediction horizon are included in predictions. The average prediction error for 161 predicted events (including signal passages) is 19.33 s, while the maximum prediction error is 68.71 s. The maximum prediction error was produced for the passing time of a signal between stations Delft South (DTZ) and Schiedam (SDM) by train IC9216. The data set for that train line is significantly smaller than for other lines due to its frequency (it runs only once in an hour while other lines run every 30 min). We expect that the accuracy of predictions for that particular line will improve when a larger training data set is considered.

The major advantage of the presented model for traffic controllers is the ability to predict all route conflicts within the prediction horizon. We use the principle of overlapping

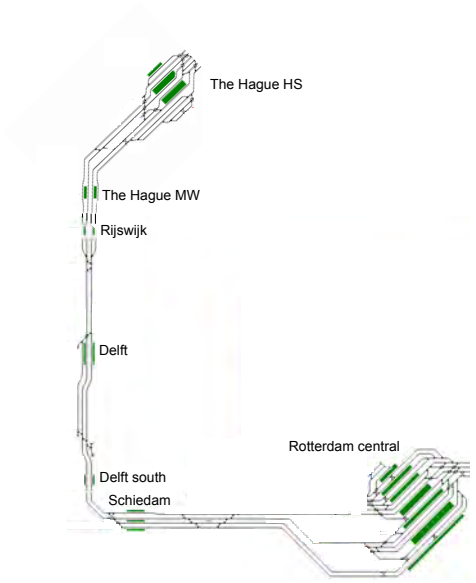

Figure 6: Schematic layout of the observed network [21]

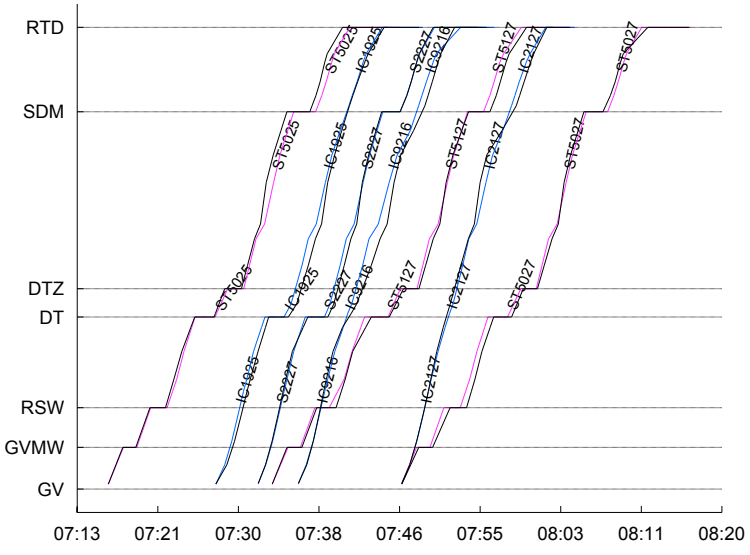

Figure 7: Time-distance diagram of predicted (at 7:13) and realised train paths

blocking times [12] to predict and visualise route conflicts.

Figures 8 and 9 show the predicted and realised blocking time diagram respectively. Local trains are presented in magenta and intercity trains in blue. Overlaps in blocking times that indicate route conflicts are given in red.

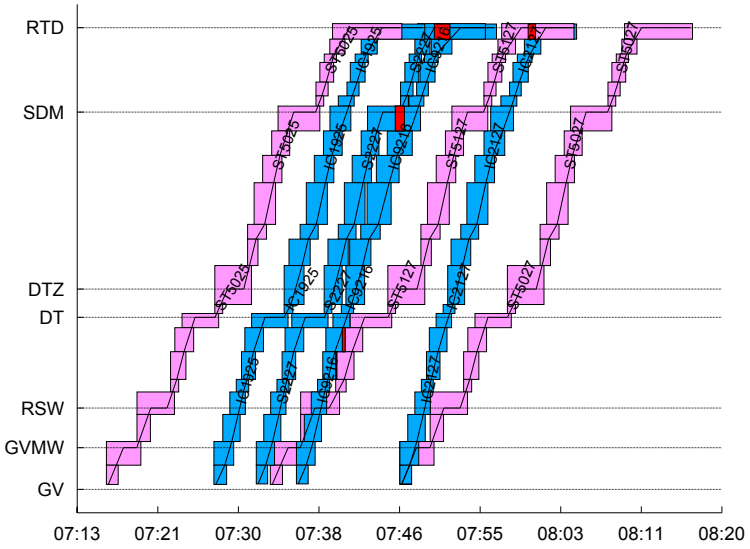

Figure 8: Blocking time diagram predicted at 7:13

The three out of the four major route conflicts that occurred, one in Schiedam (SDM) and two in Rotterdam (RTD), were predicted by the model. However, the very short running time of IC2127 between stations Delft South (DTZ) and Schiedam (Figure 7) caused a route conflict with the preceding ST5127 at SDM (Figure 9), which was not captured by the model (Figure 8).

Moreover, the chain of very short route conflicts at station Delft (DT) was not captured but only the conflict between IC9216 and ST5127. This example shows the morning peak hour which might cause longer dwell times in Delft and other short stops. Conditioning the dwell times on period of the day might improve these results. However, these inaccuracies of the current model did not affect the predictions of arrival times in Rotterdam.

Computation time of the online prediction model in this example is very short (less than one second). The model is therefore suitable for real-time applications with regular updates of current train positions. For larger examples that model dense traffic, it is expectable that more events can occur almost simultaneously, i.e., more than one update can arrive within one second. The presented event-driven prediction algorithm can easily be modified to a time-driven version where the prediction process is performed in regular time intervals based on the information that arrived within the interval.

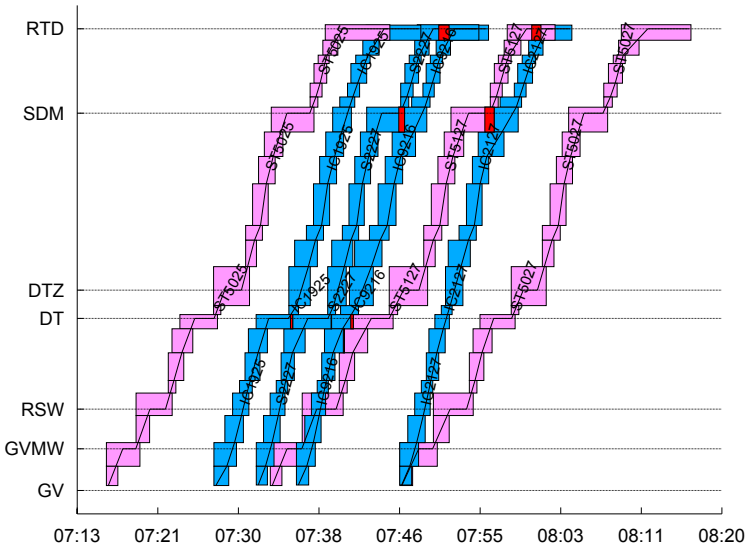

Figure 9: realised blocking time diagram

6 Summary and future work

This paper presents a microscopic model for accurate prediction of event times based on a timed event graph with dynamic arc weights. The process times in the model are obtained dynamically using processed historical train describer data, thus reflecting all phenomena of railway traffic captured by the train describer systems and preprocessing tools. The graph structure of the model allows applying fast algorithms to compute prediction of event times even for large and busy networks.

The main contribution of our approach is the dynamic estimation of process times for each train by using the predetermined functional dependence of process times on actual delays. Train interactions are modelled with high accuracy by including the main operational constraints and relying on actually realised corresponding minimum headway times (obtained from the historical data) rather than on theoretical values. The recursive depth-first search algorithm with dynamic arc weights gives predictions for all event times within the horizon.

The model has been applied in a case study on a busy corridor in the Netherlands in a simulated real-time environment, and produced accurate estimates for train traffic and route conflicts within 30 min. Application of the model to a wider area is possible either by enlarging the observed area or by coordinating multiple areas. Finally, the model structure enables straightforward application of the network-wide delay propagation algorithm [8] to estimate the further effect of current traffic conditions (or examined traffic control actions).

The further research will be focused on investigating the impact of other factors, such as period of the day and weather, on train running and dwell times. Factors with strong

impact on process times will be included in the prediction procedure in order to improve the accuracy.

The predictive model provides effective decision support to signallers and traffic control and contributes to a better utilisation of railway infrastructure, improved reliability of train services, and more reliable and dynamic passenger information. The developed model will be embedded in a closed-loop model-predictive railway traffic control framework where online optimization algorithms will automatically resolve detected conflicts and propose control decisions to traffic controllers together with the predicted conflicts [14]. This way an intelligent railway traffic management system will be obtained that pro-actively monitors the railway traffic and supports traffic controllers with decisions that optimize the traffic on a network level, beyond the traditional local control areas.

Acknowledgments
This paper is a result of the research project funded by the Dutch Technology Foundation STW: "Model-Predictive Railway Traffic Management" (project no. 11025).

References

[1] Berger, A., Gebhardt, A., Müller-Hannemann, M., and Ostrowski, M. Stochastic Delay Prediction in Large Train Networks. In Caprara, A. and Kontogiannis, S. (eds.), *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, pp. 100–111, Dagstuhl, 2011.

[2] Büker, T. and Seybold, B. Stochastic modelling of delay propagation in large networks. *Journal of Rail Transport Planning & Management*, 2(1-2):34–50, 2012.

[3] Corman, F., Goverde, R.M.P., and D'Ariano, A. Rescheduling dense train traffic over complex station interlocking areas. In Ahuja, R. K., Möhring, R. H., and Zaroliagis, C. D. (eds.), *Robust and Online Large-Scale Optimization*, vol. 5868 of *Lecture Notes in Computer Science*, pp. 369–386. Springer, Berlin, 2009.

[4] Daamen, W., Goverde, R.M.P., and Hansen, I.A. Non-Discriminatory Automatic Registration of Knock-On Train Delays. *Networks and Spatial Economics*, 9(1):47–61, 2008.

[5] D'Ariano, A., Pranzo, M., and Hansen, I.A. Conflict Resolution and Train Speed Coordination for Solving Real-Time Timetable Perturbations. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):208–222, 2007.

[6] Dolder, U., Krista, M., and Voelcker, M. RCS – Rail Control System – Realtime train run simulation and conflict detection on a net wide scale based on updated train positions. In *Proceedings of the 3rd International Seminar on Railway Operations Modelling and Analyisis (RailZurich2009)*, pp. 1–15, Zurich, 2009.

[7] Exer, A. *European Railway Signalling*, chapter Rail Traffic Management, pp. 311–343. Institution of Railway Signal Engineers, A&C Black, London, 1995.

[8] Goverde, R.M.P. A delay propagation algorithm for large-scale railway traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(3):269–287, 2010.

[9] Goverde, R.M.P. , Daamen, W., and Hansen, I.A. Automatic identification of route conflict occurrences and their consequences. In Allan, J., Arias, E., Brebbia C.A., Goodman C.J., Rumsey A.F., Sciutto, G., and Tomii, N. (eds.), *Computers in Railways XI*, pp. 473–482, Southampton, 2008. WIT Press.

[10] Goverde, R.M.P. and Meng, L. Advanced monitoring and management information of railway operations. *Journal of Rail Transport Planning & Management*, 1(2):69–79, 2011.

[11] Hansen, I.A., Goverde, R.M.P., and Van der Meer, D.J. Online train delay recognition and running time prediction. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 1783–1788, Madeira, 2010.

[12] Hansen, I.A. and Pachl, J. (eds.). *Railway Timetable & Traffic - Analysis, Modelling, Simulation*. Eurailpress, Hamburg, 2008.

[13] Kecman, P. and Goverde, R.M.P. Process mining of train describer event data and automatic conflict identification. In Brebbia, C.A., Tomii, N., and Mera, J.M. (eds.), *Computers in Railways XIII, WIT Transactions on The Built Environment*, vol. 127, pp. 227–238, Southampton, 2012. WIT Press.

[14] Kecman, P., Goverde, R.M.P., and van den Boom, T.J.J. A model-predictive control framework for railway traffic management. In *Proceedings of the 4th International Seminar on Railway Operations Modelling and Analyisis (RailRome2011)*, pp. 1–15, Rome, 2011.

[15] Longo, G. and Medeossi, G. An approach for calibrating and validating the simulation of complex rail networks. In *TRB 92nd Annual Meeting*, pp. 1–19, Washington, 2013.

[16] Medeossi, G., Longo, G., and de Fabris, S. A method for using stochastic blocking times to improve timetable planning. *Journal of Rail Transport Planning & Management*, 1(1):1–13, 2011.

[17] Nachtigall, K. Time depending shortest-path problems with applications to railway networks. *European Journal of Operational Research*, 83(1):154–166, 1995.

[18] Nash, A. and Huerlimann, D. Railroad simulation using OpenTrack. In Allan, J., C.A. Brebbia, R.J. Hill, Sciutto, G., and Sone, S. (eds.), *Computers in Railways IX*, pp. 45–54, Southampton, 2004. WIT Press.

[19] Rousseeuw, P.J. and Driessen, K. Computing LTS Regression for Large Data Sets. *Data Mining and Knowledge Discovery*, 12(1):29–45, 2006.

[20] Siefer, T. and Radtke, A. Evaluation of delay propagation. In *Proceedings of 7th World Congress on Railway Research*, Montreal, 2006.

[21] Sporenplan. `www.sporenplan.nl`, 2013.

[22] Theeg, G. and Vlasenko, S. (eds.). *Railway Signalling & Interlocking: International Compendium*. Eurailpress, Hamburg, 2009.

[23] Van der Aalst, W.M.P. *Process mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, Berlin, 2011.