Online Data-Driven Adaptive Prediction of Train Event Times

Pavle Kecman and Rob M. P. Goverde

***Abstract*—This paper presents a microscopic model for accurate prediction of train event times based on a timed event graph with dynamic arc weights. The process times in the model are dynamically obtained using processed historical track occupation 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 networks. The accuracy of predictions is increased by incorporating the effects of predicted route conflicts on train running times due to braking and reacceleration. Moreover, the train runs with process times that continuously deviate from their estimates in a certain pattern are detected, and downstream process times are adaptively adjusted to minimize the expected prediction error. The tool has been tested and validated in a real-time environment using train describer log files.**

***Index Terms*—Data driven, graph model, prediction algorithm, rail transportation.**

I. Introduction

ACCURATE prediction of train positions in time and space is a basic requirement for effective route setting, traffic control, rescheduling, and passenger information [1]. However, in practice, only the last measured train delays are known in the traffic control centers, 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 using a parallel shift of the timetable. This method neglects the fact that some trains may (partially) recover from a delay using running time supplements, whereas others may get (more) delayed due to route conflicts.

Real-time models for traffic state prediction rely on the actual train positions, speeds, relative train orders, and routes. Real-time prediction models presented in this review can be classified according to their scope into 1) microscopic, which predict all signals passing and station event times, and 2) macroscopic, which focus only on station events such as departures and arrivals.

Manuscript received January 14, 2014; revised May 27, 2014; accepted July 31, 2014. Date of publication August 29, 2014; date of current version January 30, 2015. This work was supported by the Dutch Technology Foundation STW under Project 11025: "Model-Predictive Railway Traffic Management." The Associate Editor for this paper was B. De Schutter.

The authors are with the Department of Transport and Planning, Delft University of Technology, Delft, 2628 CN, The Netherlands (e-mail: p.kecman@tudelft.nl; r.m.p.goverde@tudelft.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2014.2347136

Berger *et al.* [2] created a stochastic-graph-based macroscopic model for delay prediction. The approach is suitable for online applications where updates about train positions are frequent. Running times are modeled as random variables with probability distributions conditional on departure time and train type. By using a set of waiting policies for passenger connections, delay propagation over the observed network can be predicted. The model has been applied on a large-scale case study in Germany for predicting delays over the entire network over an hour-long prediction horizon. Due to the low level of detail in modeling capacity constraints, it is unsuitable for applications in a real-time traffic control context.

Hansen *et al.* [3] presented a traffic state prediction approach that relies on the actual state of traffic and exploits it to derive estimates for train running and dwell times. A macroscopic model for prediction of train running times is calibrated from historical track occupation data. The robust estimates of minimum running times on the level of open track sections (line segment between two stations) are computed and used to predict subsequent arrival times. The prediction algorithm estimates the realization time of an event by computing the critical path through the macroscopic graph starting from the last realized event. The main contribution of this data-driven approach is that estimates of process times reflect phenomena of railway traffic such as dependence on delay, peak hours, weather, and rolling stock. In our paper, we extend this approach to the microscopic level in order to accurately model train behavior and interactions on open track sections and in stations. Moreover, the prediction algorithm has been refined to predict all event times within the prediction horizon in a single call.

The graph-based model presented by D'Ariano *et al.* [4] was designed for real-time rescheduling. Accurate prediction of future train positions was an important requirement in order to estimate the effect of proposed rescheduling actions. Temporal decomposition was used to apply the model for predictions over several hours. Although the model considers the microscopic capacity constraints of railway traffic on open track segments between stations, route setting and release principles in station areas were not considered. Moreover, running times are estimated based on theoretical values and dwell times based on minimum dwell times, which does not reflect the impact of delays, peak hours, and passenger volumes on process times.

Fukami and Yamamoto [5] presented a real-time prediction system for a high-speed line in Japan. The system follows positions of all trains on the network using train describer messages and estimates speeds of running trains by assuming constant speed over an insulated joint. Train trajectories until

the arrival to the next station are then simulated with respect to all microscopic operational constraints. However, predicted arrival delays at the next station are extrapolated to succeeding stations using the linear-shift method. That drastically reduces the complexity of the computationally demanding simulations but also makes the system less applicable for long corridors or complex networks.

An online prediction tool has been implemented in the Swiss traffic control system RCS-DISPO [6]. The main part of this tool is a model based on a directed acyclic graph. Nodes in the graph represent arrival and departure events at timetable points and signals, and arcs represent precedence relations between nodes corresponding to running, dwell, headway, and connection arcs. Arc weights are computed offline by solving the train motion equations using a detailed description of infrastructure and train characteristics. After each train position update, a critical-path algorithm derives time estimates of all other events in the graph. Prediction errors smaller than 1 min were obtained for events within a 10-min prediction horizon. However, this approach does not explicitly model train dynamics after route conflicts nor does it exploit the available information about the actually running trains.

This paper presents a real-time tool for continuous online prediction of train traffic using a detailed graph model that captures all scheduled events and signal passages and precedence relations between them, such as train runs and stops, connections, and minimum headways. Robust estimates for process times are dynamically derived from the preprocessed historical traffic data. The model is instantaneously updated 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. Moreover, trains with process times that continuously deviate from computed estimates in a certain pattern are detected, and downstream process times are adapted to minimize the expected prediction error. The model performance is demonstrated on the busy corridor of Leiden–Dordrecht in the Netherlands.

The next section describes the methodological framework of the prediction tool. Section III describes the microscopic traffic model, and Section IV describes the prediction algorithm. The results of an application on a real-world case study are given in Section V, and the summary and conclusions are given in Section VI.

II. Framework of the Online Prediction Tool

The methodological framework of the prediction tool is illustrated in Fig. 1. The parts of the tool presented in this paper are shown with shaded boxes. The traffic model is based on a directed acyclic graph with dynamic arc weights. The graph topology is built and updated based on the actual process plan (train orders, 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 traffic control for the duration of prediction horizon. The route plan for a train is given as a planned sequence of block sections in the train route that can be used to determine the necessary headway arcs for conflicting routes. Each change of the actual plans or information from the real-time operations, i.e., changing the relative order of trains, adding or canceling trains, modifying train routes, updating connections, and removing passed events, results in an update of the graph topology.

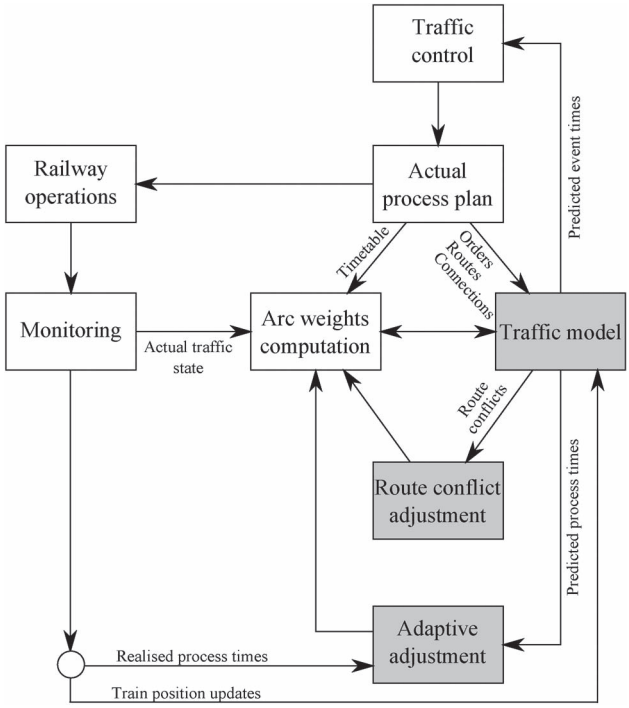

Fig. 1. Components of the online prediction tool.

Arc weights represent the estimated process times that are computed based on the actual (predicted) traffic state and processed historical data. The actual traffic state, comprising the current train positions and delays, is continuously provided by the monitoring component, and the future traffic state is obtained from the traffic model. The weight of an arc is time dependent and assigned in a dynamic way depending on the (estimated) starting time of the modeled process. By comparing the actually realized event times with the scheduled times, the actual delays are obtained. Similarly, we obtain predicted delays by subtracting the predicted event times from the scheduled ones. Arc weights are computed based on the predetermined dependence relations of process times on delays and peak hours [7].

This primary prediction loop is extended with an adaptive component that compares the actually realized process times of the running trains with the predicted values and adapts the running time until the next scheduled stop to minimize prediction error. The adaptive component of the prediction model enables online detection of the train runs with process times that continuously exceed or fall behind the computed estimates and adjusts the predictions of future train behavior accordingly.

Furthermore, the accuracy of predictions is increased by adjusting the running times over the approaching block for the hindered trains. Route conflict duration is predicted, and

the corresponding adjustment factor is retrieved from the predetermined dependence of running time increase on conflict duration. After every graph update, a prediction of event times of all reachable events is performed by traversing the graph model in topological order.

III. Microscopic Graph Model

This section presents the model, graph-building procedure, and computation of arc weights. The railway traffic represented as a discrete-event dynamic system is modeled with a timed event graph $G = (V, E)$, where V is a set of nodes, and E is a set of arcs [8]. Each event is modeled by a node. 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). The microscopic graph model needs to support rescheduling actions such as reordering, rerouting, revising services (canceling transfers or trains and adding extra trains), and retiming. Therefore, the graph G is constructed in the form of an adjacency list that is a suitable data structure that supports operations on dynamic sets [9].

A node $i \in V$ is described by $(n_i, infra_i, type_i, prev_i, next_i, t_i^{\mathrm{pred}}, t_i^{\mathrm{rec}})$, representing the train number, infrastructure element (signal or platform track), type, direct predecessors, direct successors, predicted realization time, and the recorded time (when available), respectively. Nodes that model scheduled events, i.e., arrivals and departures are also attributed with the scheduled event time t^{sch}. 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).

An arc models a precedence relation between events. Apart from modeling the running and dwelling processes related to a specific train, directed arcs are also used to model interactions between trains, namely, minimum headway and connection constraints. Arc $(i, j) \in E$ is described by $(i, j, w_{i,j}, type_{i,j})$ representing the tail event, the head event, the arc weight, and the arc type ('dwell,' 'run,' 'headway,' and 'connection').

A. Graph Construction

The graph is constructed based on the given train routes and relative train orders, timetable, and connection plan. A train route is represented as a sequence of track sections and signals. For events belonging to the same train, running arcs connect all signal passing events, as well as signal passing events with station events. Dwell arcs connect station events, i.e., an arrival event with a subsequent departure event.

A 'headway' arc separates the successive occupations of a block between two signals or a station route by different trains. If two train routes have a track section in common, a headway arc is constructed between the corresponding signal events to prevent the second train from passing the signal that protects the block (route) until it has been released by the first train.

A 'connection' arc models the fixed commercial constraints (passenger transfers) or logistic constraints (rolling-stock and crew connections) determined by the traffic control center. The arrival event of a feeder train is the tail event, and the departure event of a connecting train is the head event.

The graph can be constructed based on the train list (each train is described by the route and timetable), train orders, and the list of planned connections. Given $G = (V, E)$, the dynamics of railway traffic can be simulated with the following constraints:

$$t_j^{\mathrm{pred}} \geq t_i^{\mathrm{pred}} + w_{i,j}, \quad \forall i, j \in V, (i, j) \in E \tag{1}$$

$$t_i^{\mathrm{pred}} \geq t_i^{\mathrm{sch}}, \quad i \in \{V | type_i = \text{'departure'}\}. \tag{2}$$

Constraint (1) defines the precedence relation between the tail and the head event of an arc. Inequality (2) represents the timetable constraint for all departure events. The graph topology is continuously updated according to the rolling prediction horizon and traffic control decisions. 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 of train positions (signal passage and 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 realized event times), thus keeping the size of the graph stable within a certain time interval.

B. Computation of Arc Weights

In order to include the dependence of process times on delay, a dynamic time-dependent computation of arc weights is implemented in the prediction algorithm. Timed event graphs with uncertain process times were previously studied by Di Loreto *et al.* [10] by considering a finite interval for each variable process time. Schullerus *et al.* [11] proposed a solution to the parameter (process time) estimation from given measurements for timed event graphs represented as max-plus-linear systems.

The principal idea of time-dependent arc weights considered in this paper is that the running and dwell time of a train depends on the current delay and peak hours. In an earlier study, Kecman *et al.* [7] developed the predictive models for estimating process times based on historical track occupation data from the Dutch train describer system. The correlation between running and dwell times with actual delays is determined using least trimmed squares robust linear regression [12]. A separate time-series model was used to incorporate the effect of extended dwell times during peak hours.

Departure delay was used as the main explanatory variable for running time estimation. Delayed trains may run with full speed in order to use the running time supplements to reduce the delay. On the other hand, trains running on time or ahead of their schedule run in a lower speed, thus avoiding early arrivals and achieving energy-efficient driving [3].

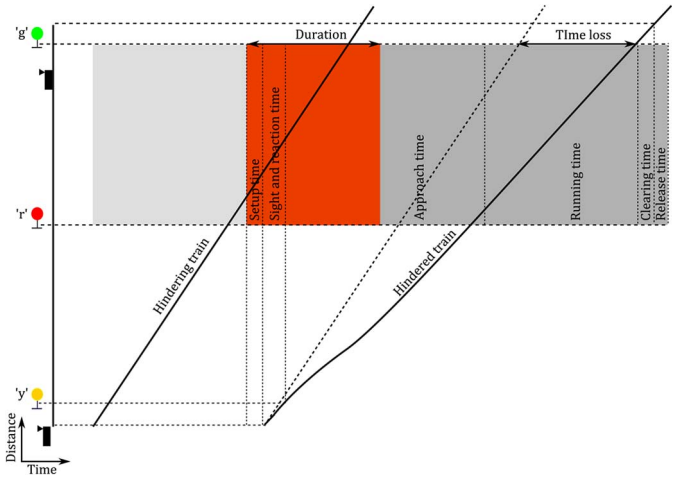

Fig. 2. Route conflict presented as an overlap of blocking times.

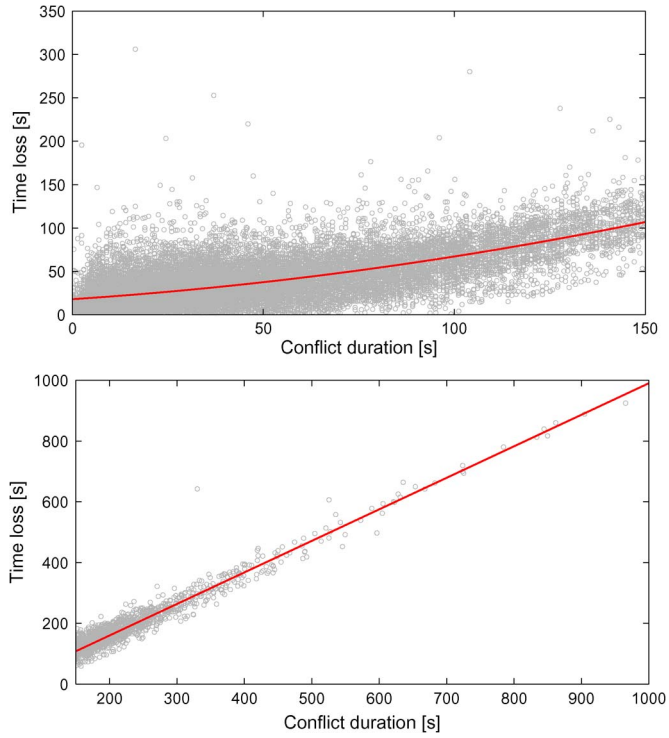

Fig. 3. Time loss dependence on conflict duration: quadratic fit for short $R^2 = 0.79$ (up) and linear fit for long conflicts $R^2 = 0.92$ (down).

Dwell times of trains in stations may depend on arrival delay. Since trains cannot depart from a station before the scheduled departure time, early trains have longer dwell times than scheduled in order to avoid early departures. On the other hand, trains with a positive arrival delay depart after a minimum dwell time in order to reduce the departure delay. Dependence of dwell time duration on peak hours was used to explain the dwell time variability of delayed trains.

Minimum headway and connection times are estimated with a small percentile of the values observed in historical data. They are considered to be independent of the actual delays. The minimum headway time is determined between all sequences of train lines observed in historical data.

For the purpose of obtaining the arc weights dynamically, each block section and station route in the database was attributed with regression coefficients for running time estimation that are separately computed for each train line and estimates of minimum headway times. Similarly, the regression coefficients, time-series parameters, and minimum connection times are stored for each station and train line pair. We define a mapping Ω that retrieves the weight of arc (i, j) depending on the current delay z_{n_i} of train n_i from the database W, i.e., $w_{i,j} = \Omega(W, z_{n_i}, (i, j))$.

C. Time Loss Due to Route Conflicts

The running time estimates are computed based on the free running times. Therefore, if a route conflict is detected, the arc weights that model the running processes of the hindered train over the affected blocks need to be adjusted to take into account braking (and possible waiting time in front of the signal), running at a lower speed, and reacceleration for every predicted route conflict [13].

Fig. 2 shows the blocking times and signal aspects that describe a route conflict that occurs when the hindered train arrives at the sight distance of the approach signal at the moment when the hindering train is still running over the block. A conflict-free train run is ensured if the preceding train has cleared the block and the switching (release and setup) time has passed by the time the next train reaches the sight distance of the approach signal. A route conflict has occurred when the train driver of the approaching train sees a yellow signal aspect indicating that he needs to be prepared to stop at the next signal. Typically, the running time over a block before the red signal aspect increases due to braking and possible waiting time in front of the red signal. Similarly, the running time over the subsequent blocks increases due to reacceleration.

We define the duration of a route conflict as the time difference between the passing time of a 'yellow' signal and the release time of the subsequent signal to a permissive aspect ('yellow' or 'green'). In the context of blocking time theory [14], a route conflict is defined as an overlap in blocking times indicated in red in Fig. 2.

The impact of a route conflict on the running time of the hindered train over the subsequent block depends on the conflict duration and the route and running time of the hindering train. The typical situation that occurs in practice when the two conflicting trains follow the same route is the 'conflict wave,' where the hindered train keeps passing signals that show the yellow aspect and is thus unable to reaccelerate to full speed [13]. We therefore consider the time loss due to reacceleration only after the hindered train passes a green aspect signal.

In order to estimate the effects of route conflicts on train running times, all route conflicts within 82 days of traffic on the busy corridor of Leiden–Dordrecht in the Netherlands were filtered out [15]. A quadratic robust fit was used to determine the correlation between conflict duration and the resulting time loss for short conflicts. Conflicts longer than 150 s were estimated with a linear fit (see Fig. 3). Time loss is obtained as the difference between the realized running time over a block

and the predicted conflict-free running time derived depending on the current train delay.

IV. Online Prediction of Event Times

This section describes the online prediction algorithm that traverses the graph in topological order. When an event happens, the corresponding node is selected as a root node. The algorithm then visits all reachable nodes in a graph with dynamic arc weights and predicts all event times within the prediction horizon. If route conflicts are predicted, the running times of affected trains are adjusted. Finally, the actual information from the running trains is used to smoothen the prediction error for future process times.

A. Prediction Algorithm

A recursive depth-first search (DFS) [9] is chosen as the method of traversing the graph due to its low memory requirements, which is an important constraint for large graphs. For implementation purposes, an attribute $color$ is added to each arc with value 'white' to indicate that the arc has not been discovered, or 'black' if the arc has been discovered (appropriate weight has been assigned).

After each event realization, the reachable set of nodes is traversed, where the root node is the node that models the realized 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.

The weights of running and dwell arcs are determined online with every graph traversal using the functional dependence of a process time on the current train delay and peak hours (see Section III-B). During the algorithm execution, the predicted event times of a scheduled event will provide predicted delays of trains. Therefore, subsequent process time estimates are computed with respect to $\hat{z}$, which is a vector that contains the predicted delays for each train.

Finally, every first node in the planned route of a train, modeling the entrance time of train (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. After processing each event realization, the graph is updated by removing the realized event together with all incoming and outgoing arcs. An arc between node 0 and the next node in the event sequence of a train is added. The weight of the added arc is equal to the predicted realization time of the next event of the train. Moreover, every traffic control action also results in a graph update. The process times of each added train are initially calibrated with respect to the actual delays and expected entrance times.

When an update about the occurrence of event $i \in V$ arrives, a set of reachable nodes $\hat{V}$ is computed that comprises all nodes reachable from and including i. The recorded event time is set as $t_i^{\mathrm{pred}} \leftarrow t_i^{\mathrm{rec}}$, and the color attribute of each arc in the subgraph is set to 'white.' If $type_i \in \{\text{'departure'}, \text{'arrival'}\}$, the current delay value of the corresponding train is updated $z_{n_i} \leftarrow t_i^{\mathrm{rec}} - t_i^{\mathrm{sch}}$. The information is propagated through the graph, and predicted event times of all reachable events are computed according to Algorithm 1.

Algorithm 1 PredictEventTimes

```
1: Input: G, V̂, W, z, i
2: ẑ ← z
3: for all j ∈ next_i do
4:   w_{i,j} ← Ω(W, ẑ_{n_i}, (i, j))
5:   color_{i,j} = 'black'
6:   if color_{k,j} = 'black', ∀k ∈ prev_j ∩ V̂ then
7:     t_j^pred ← max_{k∈prev_j} (t_k^pred + w_{k,j})
8:     if type_j ∈ {arrival, departure} then
9:       if type_j = departure then
10:        t_j^pred ← max(t_j^pred, t_j^sch)
11:      ẑ_{n_j} ← t_j^pred − t_j^sch
12:    if t_j^pred ≤ T_hor then
13:      PredictEventTimes(G, V̂, W, ẑ, j) //recursive call
14: return G, ẑ
```

The main loop of the prediction algorithm is initiated in line 3. In line 4, the actual weight of an outgoing arc is computed using the procedure described in Section III-B. If all constraints on the event realization time are known, i.e., all direct predecessors within the subgraph were visited and all incoming arcs traversed (line 6), the predicted event time is computed in line 7. Otherwise, a new iteration of the main loop is initiated. The timetable constraint for departure events is included in line 10. For all scheduled events, the predicted delay vector is updated in line 11. Finally, if the predicted event time is within the prediction horizon T_{hor}, a recursive call of the algorithm is performed in line 13.

Note that the predicted event time may also depend on events that are not reachable from the realized event and, thus, do not belong to the subgraph. For that reason, it is required to explicitly define the set of reachable nodes $\hat{V}$. In such cases, the weight of (k, j), for $k \in prev_j \setminus \hat{V}$ in line 7 can be computed using the same procedure $w_{k,j} \leftarrow \Omega(W, \hat{z}_{n_k}, (k, j))$. The predicted event time of k is retrieved from G as a prediction during an earlier algorithm call.

Since Algorithm 1 represents a modified version of a DFS algorithm, its complexity can be determined in a similar way [9]. The modification restricts the generic DFS algorithm to follow the topological order of the graph. The prediction algorithm sweeps through the subgraph of reachable nodes $\hat{V}$, and it is called exactly once for each node (line 6). For an algorithm call for node $i \in \hat{V}$, the main loop (lines 3–13) is called $|next_i|$ times. Since $\sum_{i \in \hat{V}} next_i = \hat{E}$, where $\hat{E} = \{(j, k) \mid \{j, k\} \in \hat{V}\}$, the running time of Algorithm 1 is $O(|\hat{E}|)$.

B. Adjusting the Running Time Estimates Due to Route Conflicts

The prediction of route conflicts can be performed by extending the microscopic model with the principles of blocking time

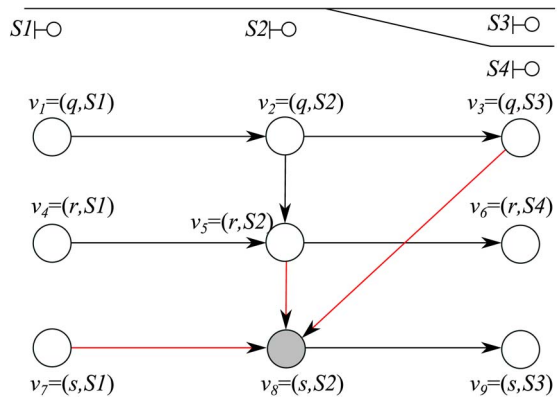

Fig. 4. Illustrative example of route conflict prediction.

theory. A route conflict can be identified by the overlapping blocking times (see Fig. 2). Since the running time estimates are computed based on the free-running times, arc weights that model the running processes over affected blocks need to be adjusted to take into account braking (and possible waiting time in front of the signal), running at a lower speed, and reacceleration for every predicted route conflict. Algorithm 1 is therefore extended with a running time adjustment procedure. For each predicted route conflict, the increase of running time of the hindered train is computed depending on the conflict duration.

In this paper, we consider a conventional three-aspect signaling system. The weight $w_{i,j}$ of running arc (i,j) needs to be adjusted if it is estimated that $Signal_i$ will be showing a 'yellow' aspect at the moment when the train arrives at the sighting distance of the signal. This moment is obtained by modifying the predicted signal passing time t_i^{pred} with a fixed value of 12 s for the sight and reaction time of the train driver. The signal aspect can be determined by comparison with the release time (switch to permissive aspect) of the following signal $Signal_j$:

$$t^{\text{rel}}_{Signal_j} = \max_k \left(t_k^{\text{pred}} + w_{k,j} \right), \quad \text{where } k \in \{prev_j \setminus i\}. \tag{3}$$

A route conflict has been predicted if $t_i^{\text{pred}} - 12 < t^{\text{rel}}_{Signal_j}$. The duration of the conflict (see Fig. 2) is computed as $d \leftarrow t^{\text{rel}}_{Signal_j} - (t_i^{\text{pred}} - 12)$, and the predicted time loss Δ as a function of d is determined from historical data as explained in Section III-C. After updating the running time estimate $w_{i,j} \leftarrow w_{i,j} + \Delta$, the predicted event time can be computed.

An example of route conflict prediction is given in Fig. 4. The graph shows three trains q, r, s (the trains are planned to pass signal $S2$ in that order) with their planned routes over the given subnetwork. Using the introduced notation, we denote by t_8^{pred} the time when train s passes signal $S2$. A route conflict can be identified by comparing the passing time of train s at signal $S1$, t_7^{pred} with the earliest possible release time of signal $S2$ due to minimum headway times after passing of trains q and r, $\max(t_3^{\text{pred}} + w_{3,8}, t_5^{\text{pred}} + w_{5,8})$. If train s passes signal $S1$ before release time of $S2$ the conflict is identified, the running time estimate of train s between signals $S1$ and $S2$ can be adjusted depending on the predicted conflict duration.

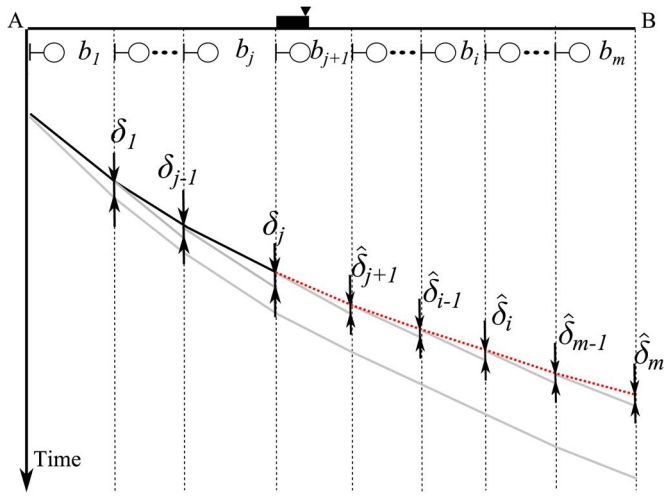

Fig. 5. Illustrative example of adaptive prediction.

C. Adaptive Adjustments of Running Time Predictions

In the presented prediction model, the estimated running times over block sections depend on departure delay from the last scheduled stop. In order to exploit the real-time information received since the last departure, an adaptive component has been developed that keeps track of the actually realized running times of a running train and adjusts the running times estimates until the next scheduled stop. A similar concept for detecting a deviation of a dynamic system from the nominal behavior based on an online observer was presented by Declerck and Guezzi [16].

A moving-average smoothing method is used to incorporate the prediction error observed during the train run into future predictions until the next stop. A schematic example of adaptive prediction is given in Fig. 5. The running train departed from station A and in the situation in the figure has just cleared the jth out of m blocks to station B where it is scheduled to stop. The gray solid line starting at station A represents the predicted running time of the train based on the actually registered departure delay. For the sake of clarity, for subsequent realized signal passages, only the predicted running time over the following block is shown.

The prediction error of the running time over block b_k is denoted by δ_k and computed after each observed signal passage. For subsequent blocks until some block $m' \in \{j+1, \ldots, m\}$, we derive the estimated prediction error $\hat{\delta}$ and adjust the estimates of running times over the remaining blocks by

$$\hat{\delta}_i = \frac{1}{l} \sum_{k=j-l}^{j} \delta_k, \quad \forall i = j+1, \ldots, m' \tag{4}$$

where l is a parameter $l \in \{1, \ldots, j-1\}$ that specifies the length of the moving average. Parameters l and m' are separately calibrated for each train type. The red dotted line in Fig. 5 denotes the adjusted prediction of running times to station B.

By applying this adaptive prediction strategy, the continuous delay sources of the conflict-free run of a single train (e.g., due to a particular driving style or defective rolling stock) as well as

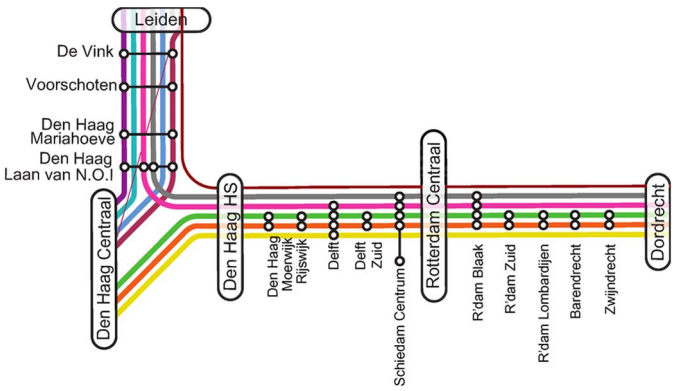

Fig. 6. Network and train lines for the case study.

temporary speed restrictions (due to infrastructure malfunctions or maintenance) will be possible to identify and include in predictions.

V. Application on a Case Study

A. Experimental Setup

In order to test the performance of the described algorithms, an experimental environment was set up that includes a static and a dynamic component. The static component consists of the database of historical track occupation data used for dynamic arc weight assignment and running time adjustments (see Section III-B).

The dynamic component of the experimental environment consist of the actual process plans for all trains within the prediction horizon and actual train event times. The actual route for each train is given on the level of track sections, which is crucial for accurate modeling of route conflicts and building the graph model. As the prediction horizon moves, new trains are added to the model, and passed events are removed.

The train describer log files contain chronologically ordered infrastructure and train step messages. We created a real-time environment for model validation by sweeping through the train describer file for one day of traffic. Every train step message (signal passage) represents the new information that is propagated through the graph using Algorithm 1. The actually realized that train event times are used to test the accuracy of predictions.

B. Description of the Case Study

The experimental setup was built for the busy corridor of Leiden–The Hague–Rotterdam–Dordrecht in the Netherlands. The 60-km-long corridor is (partially) traversed daily by approximately 300 trains per direction. Fig. 6 shows the schematic representation of the observed network along with the train lines and corresponding stopping pattern for the 2010 timetable, which was available for this study. The thin line illustrates a train line that runs once per hour, whereas the other lines operate twice per hour. The training set used to create the database for arc weight computation contains traffic realization data for 82 days.

TABLE I
Model Size for Different Prediction Horizons

	Prediction horizon [min]				
	120	60	30	20	10
Average no. events	1040	532	269	180	90
Average no. arcs	2288	1117	590	389	202

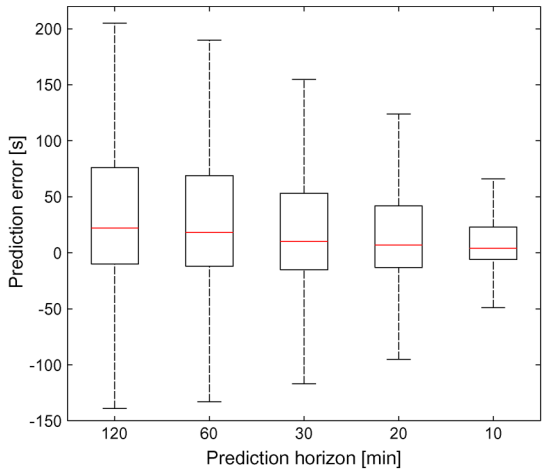

Fig. 7. Box plots of prediction error distributions for different prediction horizons.

The selected corridor and train routes enable testing the model with all possible train interactions—merging, diverging, and intersecting routes. The part of the corridor between Delft and Rotterdam Centraal as well as the branches toward Den Haag Centraal is a double-track line. The remaining part is a four-track line, where two tracks are dedicated for each direction.

C. Comprehensive Analysis

This section presents a comprehensive analysis of the prediction tool performance based on the application to one day of traffic on the corridor. The prediction algorithm is initiated, and the rolling horizon moves after receiving each of the 9776 messages that report the realization of signal and station events that occurred on the observed day. Table I shows the average number of events that are predicted in each algorithm execution, and the average number of arcs for prediction horizons of 2 h, 1 h, 30 min, 20 min, and 10 min. The average number of nodes and arcs is monotonically decreasing as shorter prediction horizons are considered.

Fig. 7 shows box plots of errors of event time predictions for each considered prediction horizon. The standard deviation of prediction error reduces with the decrease in horizon length. The median of the prediction error also follows a monotonically decreasing trend as a smaller prediction horizon is considered. Medians in each box plot show a slight positive bias, which can be explained by the fact that the process times are more often underestimated.

The accuracy of predictions is indicated by the mean absolute error (MAE). The prediction horizon of 120 min is divided into 10-s wide intervals. The absolute prediction error is computed as the absolute value of the difference between the actually realized event time and the predicted event time. MAE is

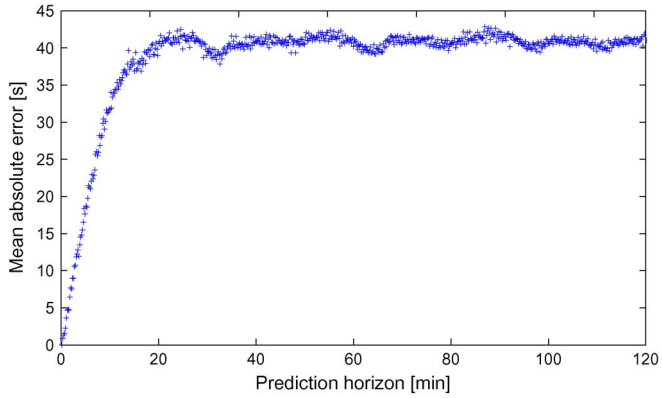

Fig. 8. Mean absolute prediction error depending on prediction horizon.

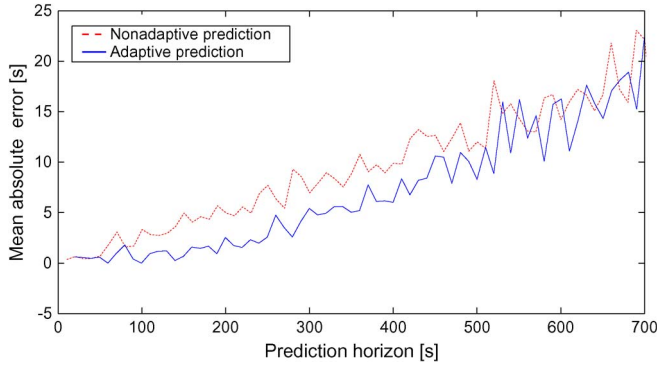

Fig. 9. MAE comparison for adaptive and nonadaptive prediction.

obtained in each interval by computing the mean value of all corresponding absolute prediction errors. The dependence of MAE on the length of the prediction horizon is shown in Fig. 8. The accuracy of predictions that are within a 30-min prediction horizon is significantly increased since more accurate information is available on events that have a direct impact on the realization time of an event. For prediction horizons shorter than 30 min, the MAE is monotonically decreasing with the horizon length. We obtain MAE smaller than 1 min for all prediction horizons. The 10-min horizon shows that in terms of average prediction error, our model outperforms the approach of predicting event times using train motion equations [6].

The benefit of adaptive prediction when applied to all observed train runs is shown in Fig. 9. Improvements are noticeable for prediction horizon of up to 10 min due to parameter l that defines the width of the moving average and m' that defines the smoothing horizon (see Section IV-C). Different combinations of parameter values were tested. Values $l = 3$ and $m' = 3$ for intercity trains and $l = j - 1$ and $m' = m$ for local trains showed the best performance in terms of MAE. Therefore, the running time estimates are adapted for the next three blocks for intercities and until the next scheduled stop for local trains. Similarly, the moving average is computed over last three block sections for intercity and over all blocks since the last departure for local trains.

The accuracy of route conflict predictions for different prediction horizons is strongly correlated with the MAE. For prediction horizons longer than 30 min, approximately 80% of route conflicts longer than 30 s are accurately predicted.

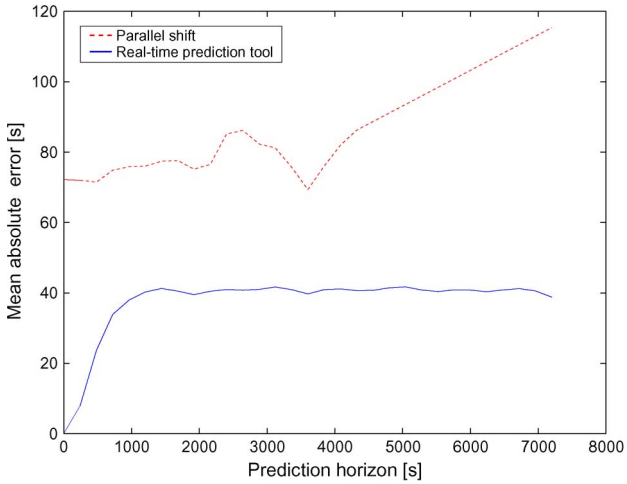

Fig. 10. MAE of scheduled event times for a linear shift strategy and the real-time prediction.

As shorter prediction horizons are considered, the accuracy of route conflicts prediction increases. For a 10-min horizon, 95% of route conflicts are accurately predicted.

The running time adjustment does not show a noticeable global effect when the MAE of all predictions of all events is considered. However, accuracy analysis performed on the isolated set of running times of hindered trains shows an increase in prediction accuracy of 8 s on average for a 30-min prediction horizon and 13 s for a 10-min prediction horizon.

Since the prediction algorithm is linear, the computational complexity, which depends on the size of the input graph, is not considered as a criterion for choosing the most appropriate prediction horizon. Even for the longest horizon, the prediction is computed in less than 1 s, including the computation of the reachable set.

Finally, Fig. 10 shows the comparison of prediction accuracy of the model presented in this paper with the conventional parallel shift strategy, which is typically applied for estimating train arrival and departure times. The analysis is performed for scheduled event times only. The benefits of real-time prediction are noticeable for every prediction horizon.

D. Example of Algorithm Execution

An example of predictions is shown in Fig. 11. The presented time–distance diagram shows the predicted train paths (local trains are given in magenta and intercity trains in blue). The realized train paths in space and time are presented with dashed lines. The prediction is performed at the departure of train ST5025 from Den Haag 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, whereas the maximum prediction error is 68.71 s.

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 blocking times [14] to predict and visualize route conflicts.

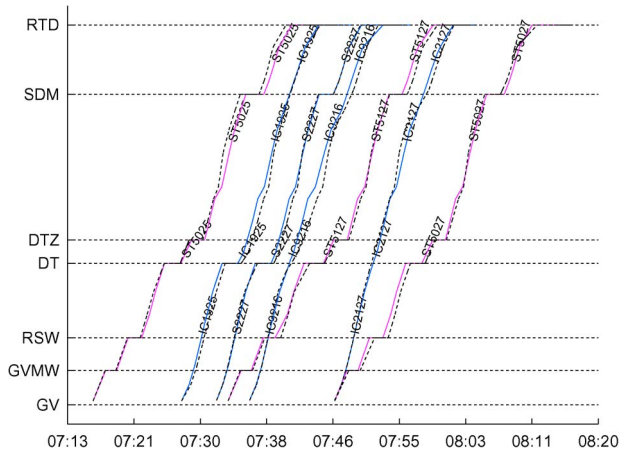

Fig. 11. Time–distance diagram of predicted (at 7:13) and realized train paths.

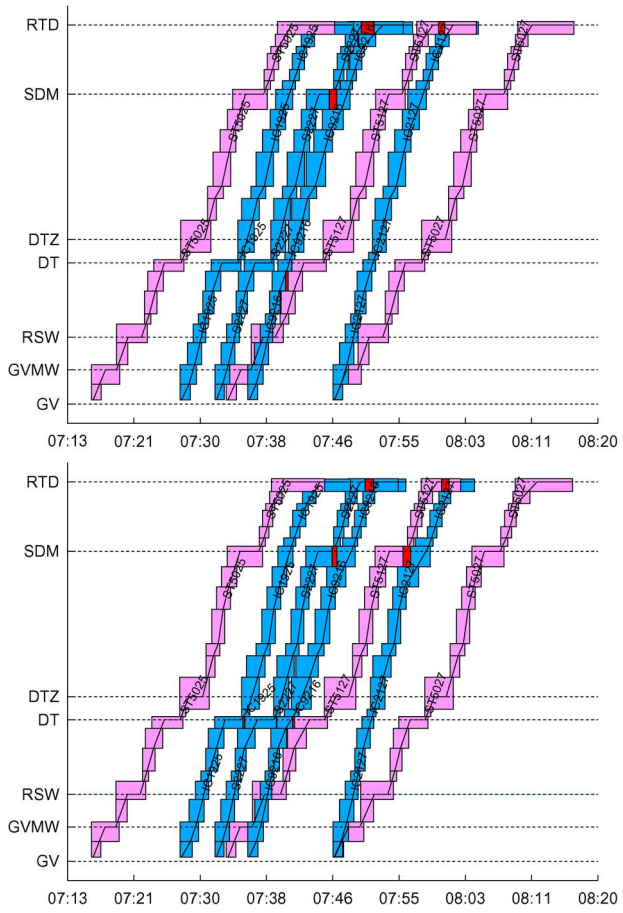

Fig. 12. Blocking time diagram predicted at 7:13 (up), realized blocking time diagram (down).

Fig. 12 shows the predicted (up) and realized (down) blocking time diagram. Local trains are presented in magenta and intercity trains in blue. Overlaps in blocking times that indicate route conflicts are given in red. The three out of the four major route conflicts that occurred, i.e., one in Schiedam (SDM) and two in Rotterdam (RTD), were predicted by the model. The fourth conflict that was not captured occurred more than 30 min after the moment of prediction.

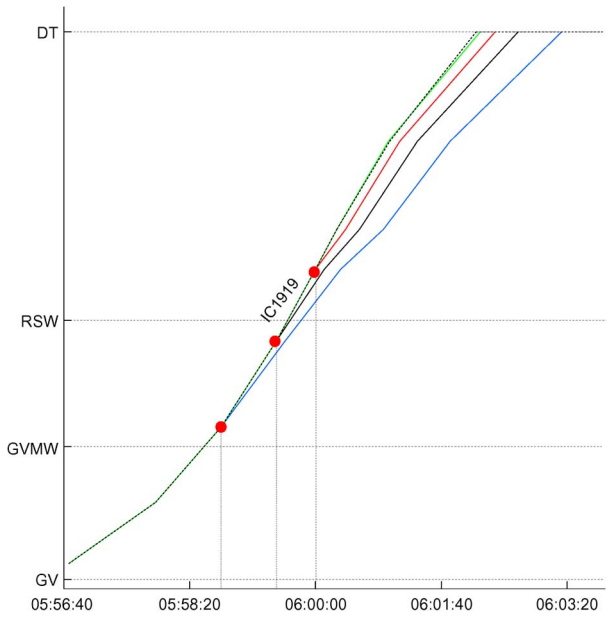

Fig. 13. Effects of adaptive prediction.

An example of adaptive prediction that minimizes the prediction error for running trains is shown in Fig. 13. The example considers intercity train IC1919 that does not have scheduled stops between The Hague HS (GV) and Delft (DT). The first prediction, derived at the moment of departure from The Hague HS, is represented by the blue line. After three corrections of running time estimates resulting in predictions shown in black, red, and green, the prediction error of less than 1 s was achieved (the final running time estimate given in green practically overlaps the realized running time given in magenta). The predicted arrival time error is monotonically decreasing as the train progresses toward station Delft. Therefore, the propagation of prediction error to connected events of other trains is reduced, thus affecting the overall performance of the model.

VI. Conclusion and Outlook

This paper has presented 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 were dynamically obtained 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 a method for dynamic estimation of process times for each train by using the predetermined functional dependence of process times on actual delays. Train interactions are modeled with high accuracy by including the main operational constraints and relying on actually realized corresponding minimum headway times (obtained from the historical data) rather than on theoretical values. The recursive DFS algorithm with time-dependent arc weights traverses the graph in topological order and predicts 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 [17] to estimate the further effect of current traffic conditions (or examined traffic control actions).

The performance analysis has been conducted using raw data from train describer log files. Although the data are very detailed and of high quality, various errors in logging of event times still occurred, which may affect the accuracy of subsequent predictions. In future research, robustness of the model to noise and errors in the data need to be increased in order to provide more stable and accurate predictions.

Finally, although the computation time is less than 1 s and memory requirements are low even for the longest considered prediction horizon of 2 h, a possible algorithmic improvement may be necessary to speed up the computation for network-wide instances. One direction is to coordinate predictions among multiple areas. Alternatively, the data structure of the graph can be modified in order to avoid the explicit computation of the reachable set for every graph update, which may be inefficient for large graphs.

REFERENCES

[1] H. Wang, F. Schmid, L. Chen, C. Roberts, and T. Xu, "A topology-based model for railway train control systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 819–827, Jun. 2013.

[2] A. Berger, A. Gebhardt, M. Müller-Hannemann, and M. Ostrowski, "Stochastic delay prediction in large train networks," in *Proc. 11th Workshop Algorithmic Approaches Transp. Model., Optim., Syst.*, A. Caprara and S. Kontogiannis, Eds., 2011, pp. 100–111.

[3] I. A. Hansen, R. M. P. Goverde, and D. J. Van der Meer, "Online train delay recognition and running time prediction," in *Proc. 13th Int. IEEE Conf. ITSC*, Funchal, Portugal, 2010, pp. 1783–1788.

[4] A. D'Ariano, M. Pranzo, and I. A. Hansen, "Conflict resolution and train speed coordination for solving real-time timetable perturbations," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 208–222, Jun. 2007.

[5] K. Fukami and H. Yamamoto, "A new diagram forecasting system for the Tokaido-Sanyo Shinkansen," in *Proc. World Congr. Railway Res.*, Köln, Germany, 2001, pp. 1–6.

[6] U. Dolder, M. Krista, and M. Voelcker, "RCS—Rail control system—Realtime train run simulation and conflict detection on a net wide scale based on updated train positions," in *Proc. 3rd Int. Semin. Railway Oper. Model. Anal. RailZurich*, Zurich, Switzerland, 2009, pp. 1–15.

[7] P. Kecman and R. M. P. Goverde, "Calibration of a data-driven railway traffic prediction model," in *Proc. 3rd Int. Conf. Models Technol. Intell. Transp. Syst.*, T. Albrecht, B. Jaekel, and M. Lehnert, Eds., 2013, pp. 459–469.

[8] R. M. P. Goverde, "Railway timetable stability analysis using maxplus system theory," *Transp. Res. Part B, Methodological*, vol. 41, no. 2, pp. 179–201, 2007.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[10] M. Di Loreto, S. Gaubert, R. Katz, and J. Loiseau, "Duality between invariant spaces for max-plus linear discrete event systems," *SIAM J. Control Optim.*, vol. 48, no. 8, pp. 5606–5628, 2010.

[11] G. Schullerus, V. Krebs, B. De Schutter, and T. van den Boom, "Input signal design for identification of max-plus-linear systems," *Automatica*, vol. 42, no. 6, pp. 937–943, 2006.

[12] P. J. Rousseeuw and K. Driessen, "Computing LTS regression for large data sets," *Data Mining Knowl. Discovery*, vol. 12, no. 1, pp. 29–45, 2006.

[13] R. M. P. Goverde and L. Meng, "Advanced monitoring and management information of railway operations," *J. Rail Transp. Planning Manag.*, vol. 1, no. 2, pp. 69–79, 2011.

[14] I. A. Hansen and J. Pachl, Eds., *Railway Timetable and Traffic—Analysis, Modelling, Simulation*. Hamburg, Germany: Eurailpress, 2008.

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

[16] P. Declerck and A. Guezzi, "State estimation and detection of changes in time interval models," *Discrete Event Dyn. Syst.*, vol. 24, no. 1, pp. 53–68, 2014.

[17] R. M. P. Goverde, "A delay propagation algorithm for large-scale railway traffic networks," *Transp. Res. Part C, Emerging Technol.*, vol. 18, no. 3, pp. 269–287, 2010.

Pavle Kecman received the B.S. and M.S. degrees in transportation engineering from University of Belgrade, Belgrade, Serbia, in 2008. He is currently working toward the Ph.D. degree in the Department of Transport and Planning, Delft University of Technology, Delft, The Netherlands, where he is working on a research project on model-predictive railway traffic management.

His research interests include railway traffic and transportation and application of data mining and operations research to transportation-related problems.

Rob M. P. Goverde received the M.Sc. degree in mathematics from Utrecht University, Utrecht, The Netherlands, in 1993 and the MTD degree in mathematical modeling and decision support and the Ph.D. degree in railway operations from Delft University of Technology, Delft, The Netherlands, in 1996 and 2005, respectively.

Since 1996 he has been with the Department of Transport and Planning, Delft University of Technology, successively as a Ph.D. student, a Postdoctoral Researcher, an Assistant Professor, and currently an Associate Professor. He is the author of more than 75 journal papers, book chapters, and peer-reviewed conference papers. His research interests include railway timetable planning and analysis, performance analysis of railway operations, railway traffic management, signaling and safety systems, train control, max-plus algebra, operations research, and discrete event dynamic systems.

Dr. Goverde is a Board Member of the International Association of Railway Operations Research, a member of the Institution of Railway Signal Engineers, a member of the International Federation of Accountants and the Society for Industrial and Applied Mathematics, and a Permanent Program Committee Member of the IEEE International Conference on Intelligent Rail Transportation. He was a member of the International Program Committee of the IEEE Conference on Intelligent Transportation Systems 2013.