Train describer records as a source of information for infrastructure monitoring, performance analysis and traffic management

Kecman, P.; Goverde, R. M P; Hansen, L. A.

DOI
10.1049/cp.2011.0607

Publication date
2011

Document Version
Final published version

Published in
5th IET Conference on Railway Condition Monitoring and Non-Destructive Testing, RCM 2011

TRAIN DESCRIBER RECORDS AS A SOURCE OF INFORMATION FOR INFRASTRUCTURE MONITORING, PERFORMANCE ANALYSIS AND TRAFFIC MANAGEMENT

P. Kecman*, R.M.P. Goverde*, I.A. Hansen*

* Delft University of Technology, Department of Transport and Planning,
Stevinweg 1, 2628 CN, Delft, the Netherlands
{p.kecman, r.m.p.goverde, i.a.hansen}@tudelft.nl

Keywords: train describers, timetables, route conflicts,

Abstract

Data records from train describer systems represent a valuable source of information for analysing system performance and assessing railway timetable quality. The aim of this paper is to introduce the Dutch train describer system and present algorithms developed for automatic identification of track blockages, route conflicts (including the identification of conflicting trains), accurate arrival and departure times/delays at stations, and realized train paths on track section level with the associated blocking times. Analysis of these realisation data can be used to identify incidents and disruptions, as well as to determine structural errors in the timetable design in order to increase the system performance.

1 Introduction

A railway timetable for passenger services is constructed and published once a year and in heavily utilized networks, such as the national network in the Netherlands, represents a result of a complex optimization process which takes into account all operational and capacity constraints. The outcome should be a set of *realisable* and *conflict-free* event times (departure and arrival times of all trains).

Variations in process times and disruptions in railway traffic are considered to be inevitable and therefore, actions are made in order to minimize their possible effect on the system in the stage of timetable construction. It is therefore crucial to have in mind the importance of robustness and resilience of the timetable, i.e., its ability to resist and adapt to minor disturbances. For that reason, running time supplements and buffer times are introduced in order to enable trains to make up for their delay and at least to some extent avoid affecting other trains and creating secondary delays. It is necessary to carefully analyse the effectiveness of these time reserves and update the actual timetable by implementing the eventual modifications, thus increasing the punctuality of the system.

Analysis of traffic realisation data is becoming increasingly popular lately among researchers from the field of railway operation. Daamen et al. [3] developed the software for automatic conflict identification based on train describer data, *TNV-conflict*. Its add-on for statistical analysis of train realisation data, *TNV-statistics*, was presented by Goverde & Meng [9]. Furthermore, mining of train delay data was used to determine systematic dependencies between delays in Switzerland [6], Germany [1] and to identify frequent delay patterns in Belgium [2].

There are several reasons for increasing interest in traffic realisation data analysis. First, infrastructure capacity is utilised extensively in western European countries. In such conditions, when capacity consumption is close to the level of congestion and saturation [12], delays propagate easily through the network and it is therefore necessary to determine the optimal values and allocation of time reserves in order to increase robustness and resilience of the system. In that context, in the process of timetable construction, feedback in form of performance analysis is essential.

Second, adoption and implementation of EC Directive 2001/14/EC [4], implies strictly regulated, transparent relations between all participants in the railway market. Punctuality norms and schedule violation penalties are imposed on infrastructure managers and train operating companies. Therefore, deriving accurate values of delays and partitioning them in primary and secondary delays is in interest of all parties.

The third reason has a more scientific importance. Namely, mathematical and simulation models of railway traffic use stochastic distribution of process times which reflect the variations caused by e.g. driving behaviour, passenger volumes, weather conditions, etc. It is however an important feature of the models themselves to capture the interactions of trains and the resulting conflicts and knock-on delays. Consequently, partitioning realised process times data to hindered and unhindered trains is of great importance [3].

The recent implementation of the new train describer system TROTS in the Netherlands and changes in data structure imply the necessity to modify the existing tools developed for conflict identification [3]. In this paper, we present a tool for automatic conflict identification and accurate reconstruction of train movements on the level of track sections. The tool is compatible with the current Dutch train describer system. Its output includes a list of route conflicts with the hindering and hindered train, and the signal of conflict. The tool also enables straightforward analysis and identification of systematic conflicts, primary delays and filtration of unhindered train runs.

The remainder of the paper is structured as follows: train describer logfiles and their data structure are explained in the next section. Section 3 explains the subroutines and the main algorithm followed by a description of the case study, and examples in Section 4. At the end, we give a brief summary and present further application of train describer data in the framework of the on-going research about model-predictive railway traffic management [10].

2 Train describer systems

Train describer systems keep track of train positions based on train numbers and messages received from signalling and interlocking systems (sections, switches and signals) [5]. Their logging is recorded in chronologically sorted lists of infrastructure and train number messages.

2.1 Train describers in the Netherlands

The Dutch train describer system TNV (*Train number following system* in Dutch) has since 2009 gradually been replaced by TROTS (*Train Observation and Tracking System*). Whereas in TNV the train number steps were recorded on the level of windows, comprising one or more blocks (train number transition message was recorded when the train passed the signal between two windows), in TROTS, the train steps are recorded on the level of track section (a message is recorded as the train occupies and releases each section in its route). Hence, infrastructure messages about the state change of sections are already coupled to the train number that has caused the state change, thus enabling more or less straightforward application of algorithms to replay a train run (and its interactions with other trains) in contrast to several levels of preparation that were necessary to adapt the raw TNV log files [3,7,8].

The Dutch railway network has been divided into several TROTS areas. Figure 1 shows the TROTS area Rotterdam which comprises a major station Rotterdam Centraal and corridors towards Dordrecht and Hoek van Holland. TROTS logfiles are being archived every day, per area, in large files of ASCII format of approximately 75 MB.

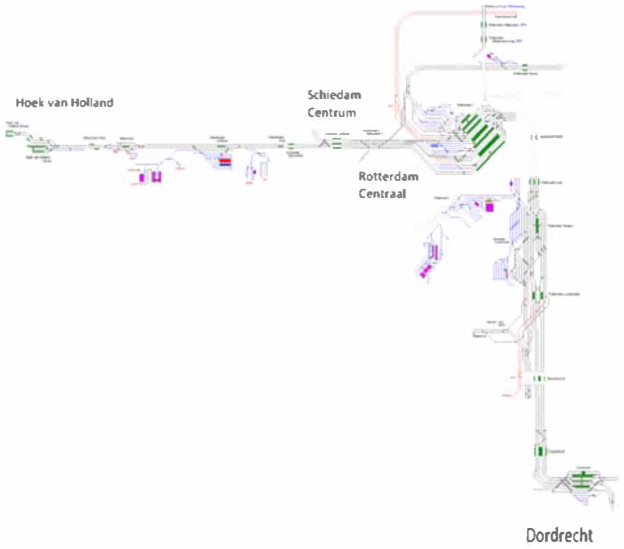

Figure 1 Rotterdam TROTS area [13]

Figure 2 shows an example of TROTS log messages within an interval of three seconds from infrastructure and trains in the area of Rotterdam. For the sake of simpler visualisation we show only parts of messages coming from sections, signals and train number steps and information therein, relevant for the algorithms presented in subsequent section.

2010-04-02_09:01:28	BM1119701	SECTIE	MSS$53BT	1
2010-04-02_09:01:28	BM1119701	ATWIJZIG	4120	
2010-04-02_09:01:28	BM1119702	SECTIE	RTD$170AT	0
2010-04-02_09:01:28	BM1119702	ATWIJZIG	2131	
2010-04-02_09:01:28	BM1119703	SECTIE	SDM$68AT	0
2010-04-02_09:01:28	BM1119703	ATWIJZIG	2122	
2010-04-02_09:01:28	BM1119704	SECTIE	KFHAZ$1414A/BT	0
2010-04-02_09:01:28	BM1119705	SECTIE	KFHAZ$1444BT	1
2010-04-02_09:01:28	BM1119704	ATWIJZIG	5029	
2010-04-02_09:01:28	BM1119705	ATWIJZIG	5024	
2010-04-02_09:01:29	BM1119706	SEIN	SDM$38	1
2010-04-02_09:01:29	BM1119707	SECTIE	SDM$A54AT	1
2010-04-02_09:01:29	BM1119708	SECTIE	WSPL$411AT	0
2010-04-02_09:01:29	BM1119707	ATWIJZIG	2122	
2010-04-02_09:01:29	BM1119709	SEIN	SDM$70	1
2010-04-02_09:01:29	BM1119708	ATWIJZIG	4027	
2010-04-02_09:01:30	BM1119710	SEIN	SDM$94	1
2010-04-02_09:01:30	BM1119711	SECTIE	SDM$712B-DT	0
2010-04-02_09:01:30	BM1119712	SECTIE	RTD$303AT	1
2010-04-02_09:01:30	BM1119711	ATWIJZIG	4131	
2010-04-02_09:01:30	BM1119712	ATWIJZIG	9318	

Figure 2 Example of a TROTS log file (extract from the messages)

The first column contains timestamps for all messages. Unique message codes are listed in the second column, and information source (SECTIE for section, SEIN for signal and ATWIJZIG for train number step) in the third. Depending on the source, entries in the fourth column are section id, signal id or the train number. Only infrastructure messages have entries in the fifth column, which contains binary state change messages (0/1 – released/occupied for sections, stop/go for signals). Train messages also give the list of sections occupied by the train at the time of logging which is not shown in Figure 2.

Unique message codes are used to couple messages reporting a state change of a section to the messages reporting the activity of the train that caused the change (e.g. code BM1119701 for the first two rows in Figure 1 indicates that the train 4120 occupied the section MSS$53BT).

2.2 Shortcomings of the TROTS system for performance analysis

There are several issues in TROTS logfiles that represent a potential source of inaccuracy and complicate performance analysis.

The system architecture [11] reveals that infrastructure messages and train step messages are generated by different components of the system which sometimes results in a significant difference (up to 7 seconds) between the timestamps of corresponding messages. All examples in this paper rely solely on messages coming from infrastructure to avoid possible inconsistencies.

Furthermore, infrastructure messages reporting a signal change to a 'stop' aspect cannot be coupled directly to trains that caused the change, nor with the occupation of any sections protected by the signal. In order to overcome this, an additional input in form of a list of all signals and sections they protect is required (only the first section of the protected

block). We can thus identify the train that caused the signal aspect change via the protected section that got occupied.

Other sources of inaccuracy are the automatic block signals on the open track which are not logged and aggregated sections. If the train runs were reconstructed from the raw data, an open track between two stations would be treated as one block between two signals (exit signal at the station of departure and home signal at the station of arrival) and the train's progress would be only roughly estimated due to the aggregation of sections. Since no readable infrastructure database is available, this problem has been overcome by manually defining an additional input containing a list of open track signals and a list of aggregated sections, with individual sections and their lengths as attributes. Three-aspect two-blocks signalling logic has been simulated to estimate aspect changes of non-logged signals. In case such signal is located between two sections aggregated into one, the moment of its aspect change to 'stop' is determined by estimating the occupation time of the protected section, which is derived as a fraction of the occupation time of the aggregated sections proportional to the ratio of individual section length and the length of the aggregated sections.

3 Tool for automatic conflict identification

In this section we present the algorithm which sweeps through the TROTS logfile once, reconstructs the movements of all trains that operated in the corresponding area whilst simultaneously deriving the list of all route conflicts that occurred.

The following input is used:

1. TROTS logfile
2. Infrastructure lists (signals with sections they protect, aggregated sections and lengths)
3. Operational timetable
4. List of platform sections

The first two inputs were explained in the previous section and the latter two are necessary to handle route conflicts of departing trains and distinguish between long occupation times of platform sections in stations (due to scheduled stops) and other sections (due to e.g. infrastructure or vehicle failures).

The object oriented approach has turned out to be a convenient way of storing the relevant information from the logfiles thus enabling the algorithms to revisit the objects, and use and update the information therein [3]. Every section, signal and train that appears in the logfile is an object attributed by a chronologically sorted list of activities. As the algorithm comes across a message that reports a state change of an infrastructure element, the corresponding objects are updated with a time stamp and the train number (for section/signal object) or infrastructure element id (for train object).

3.1 The main algorithm

The main loop is initiated when the algorithm comes across a message reporting a section occupation. The flowchart of the main loop with embedded subroutines that will be explained in the next subsection is shown in Figure 3.

After all objects have been updated, the first level of branching makes a distinction between sections protected by a signal (sections on a block boundary) and those that are not.

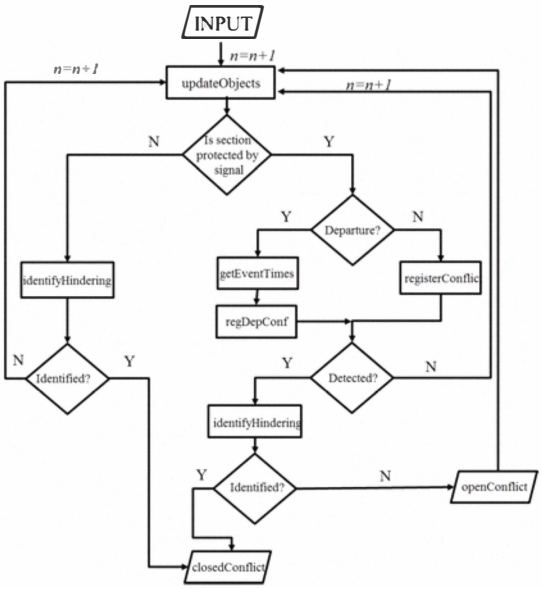

Figure 3 Main conflict identification loop

The second decision level initiates different subroutines depending on whether the train is departing from a station or not. Registered conflicts with the identified hindering train are being stored in the output *closedConflict*. On the other hand, registered conflicts with an unidentified hindering train, are stored in the list *openConflict* which is used to identify hindering trains as the train progresses along the protected block section after the conflict.

3.2 Subroutines

This subsection gives a description of subroutines that capture the main logic of the tool for automatic conflict registration. Figure 4 depicts a small part of the network with signals (S1, S2, S3), track sections (TS1-TS5) and the train that has just entered TS4, which is used to illustrate the subroutines.

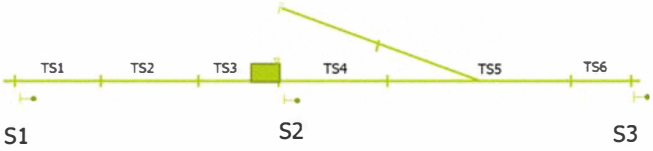

Figure 4. Illustrative example of the part of the network

Register route conflict (*registerConf*)
A route conflict occurs when a train movement is restricted by a stop signal because the protected block section is occupied by another train. The subroutine *registerConf* checks the aspect shown by the signal at the end of the block at the time when the train entered the block, using the "look back" approach. When the train passes signal S2 (Figure 4), the subroutine compares the last release time (change to 'go' aspect) of S2, t_{rel}^{S2} with the passage time of S1, t_{pass}^{S1}.

IF $t_{rel}^{S2} > t_{pass}^{S1}$ → CONFLICT REGISTERED (1)

Identify hindering train (*identifyHindering*)
As the hindered train progresses along the block section protected by the signal of conflict, *identifyHindering* compares the release time of each section belonging to the protected block (TS4, TS5, TS6 from Figure 4) with the time the hindered train passed the signal before the signal of conflict (S1, Figure 4). The train that released the section for which inequality (2) holds is the hindering train.

$$t_{rel}^{TSi} > t_{pass}^{S1}, \qquad i = 4,5,6. \tag{2}$$

Get event times (*getEventTimes*)
This routine derives the accurate arrival and departure times from TROTS logfiles. When a train occupies the section protected by the exit signal after a scheduled stop, *getEventTimes* is initiated. It determines a period of standstill as the longest time gap between successive infrastructure messages of a train. Then the time of the last message reported before the standstill is taken to be the arrival time and the time of the first message after the standstill as departure time.

Detect departure conflict (*registerDepartureConf*)
After accurate arrival and departure times have been derived, this subroutine checks whether the departing train was a victim in a route conflict. We assume here that the departing train was hindered if the exit signal was showing 'stop' at the moment of scheduled departure (if the train had no arrival delay) or after the minimum dwell time has passed since the arrival (if the train arrived with a delay).

$$\text{IF } t_{rel}^{exit} > \max\left(t_{ar} + t_{dwell}^{min}, t_{dep}^{sched}\right) \tag{3}$$
$$\rightarrow \text{CONFICT DETECTED}$$

This subroutine lists all candidates for outbound route conflicts. Extended dwell times in stations can not directly be explained by route conflicts. In order to exclude the trains that waited for a feeder train to realize a connection, or the ones that had extended dwell time for some other reason, additional information from signallers and dispatchers is necessary.

4 Case study

The algorithm presented in the previous section has been applied to the dataset of seven days in April 2010 in the Rotterdam area (Figure 1). Approximately 300 route conflicts per day were identified.

We will show two examples emphasizing the usefulness of the tool to: assist the analyst in unravelling complex conflict chains due to severe disruptions in the first (*acute* conflicts) and register systematic conflicts pointing the structural errors in timetable design (*chronic* conflicts) in the second.

4.1 Example 1 – acute conflict

This example was registered on the 2nd of April 2010 and it has been chosen because all subroutines explained in the previous section had to be applied to register this chain of conflicts.

Train number 5124 was registered as both a hindered train and a hindering train in a chain of three conflicts that were identified within a short time period in the interlocking area of Rotterdam Centraal towards Schiedam Centrum (Figure 5). Table 1 shows the route description of the four trains involved in these conflicts.

Table 1 Route description of trains involved in disruption

Train nr.	Event	Sched. time	Route		
			From signal/platf. Track	To signal/platf. track	Colour
1924	Dep.	9:17	9	114	Red
5124	Dep.	9:32	8	114	Red/blue
2133	Arr.	9:32	116	6	Green
5033	Arr.	9:42	116	6	Green

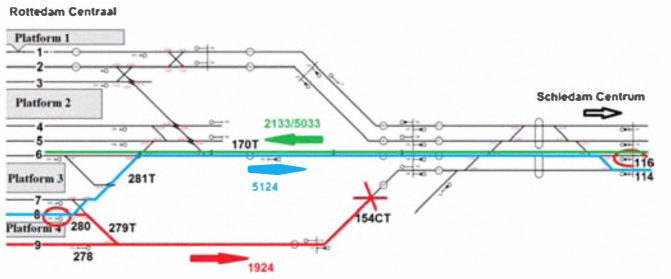

Figure 5 Example of route conflicts between Rotterdam and Schiedam [13]

The first registered conflict involved train 5124 as the hindered train and 2133 as hindering train at the exit signal 280 from platform track 8 (left red circle in Figure 5). The second conflict identified the hindrance of train 5033 by train 5124 at signal 116 (right red circle in Figure 5).

This situation pointed to the place to zoom in and analyse the realization data of the trains involved. The performed analysis revealed that train 1924, which departed on time, suffered a disruption of 25 minutes for an unknown reason on section 154CT, thus blocking the planned outbound route of train 5124 (red line Fig. 5 from exit signal 280 to home signal 114). The new outbound route for 5124 (blue line in Fig. 5) could have been set only after 2133 had arrived (green line Fig. 5). Furthermore, train 5033 could proceed along its planned inbound route (green line Fig. 5) only after 5124 had changed tracks and returned to its planned route before home signal 116.

This example also showed that the analysis of registered conflicts have been simplified by organizing and sorting the realization data.

4.2 Example 2 – chronic conflict

This example shows structural delays, i.e. conflicts that occur frequently between trains of two train lines.

We focus on train line series 9200 Brussels – Amsterdam and 1900 Venlo – The Hague. Routes of the northbound trains of the two series merge before station Dordrecht. Interlocking route and platform sections in station Dordrecht for the trains of both series are protected by signal 1136.

After Dordrecht, trains of both series proceed towards Rotterdam using different tracks of the four-track railway line

but their routes merge again after home signal 384 at Rotterdam Centraal, where both train series have the same planned platform track. According to the hourly pattern of the 2010 timetable, the scheduled time between the departure of the 9200 trains and the arrival of the 1900 trains is four minutes in both stations Dordrecht and Rotterdam Centraal.

After applying the automatic conflict registration tool on TROTS log archives for period 2-8 April 2010, it was determined that signals 1136 in Dordrecht and 384 in Rotterdam Centraal are the top two signals judged by the number of route conflicts identified (Table 2). Conflicts are more or less equally distributed over the observed seven days. Table 2 shows that a significant share of conflicts on both observed signals are between the 9200 and 1900 trains.

Table 2 Distribution of all conflicts and share of conflicts between 9200 and 1900 trains for the dataset of seven days

Date	DDR1136			RTD384		
	All conflicts	9200 /1900	%	All conflicts	9200/ 1900	%
02.04	10	5	50.1	7	4	57.1
03.04	9	8	88.9	15	9	60.0
04.04	9	5	55.6	15	7	46.7
05.04	11	9	81.8	13	9	69.2
06.04	13	8	61.5	12	5	41.7
07.04	11	3	27.3	13	4	30.8
08.04	14	5	35.7	7	3	42.9
TOTAL	77	43	55.8	82	41	50.0

This result indicates that closer attention is needed to investigate the possibility of modifying the timetable by retiming or rerouting the trains of the two series, thus increasing the robustness against frequent delays of the 9200 trains. Since the main purpose of this paper is to present the tool for automatic conflict identification, such analysis is out of our scope.

5 Summary and outlook

In this paper we presented a tool for automatic conflict identification based on train describer data and illustrated its usefulness for identifying systematic delay dependencies and analysing delays during incidents and severe disruptions. The tool is compatible with the Dutch train describer system TROTS. Applicability for other train describer systems strongly depends on their data structure.

Application of the tool on a real life case study indicated the necessity for further developments, mainly in the direction of automatic analysis by providing useful statistical indicators for structural errors in the timetable, as well as detecting severe disruptions and identifying primary delays.

Further research, which includes mining and analysis of train realisation data, focuses on deriving accurate predictions of process times within the monitoring and short-term prediction component of a model-predictive controller for railway traffic management [10]. We aim at exploiting advanced statistical and machine learning methods to capture complex dependencies between process times in heavily utilized railway networks.

Acknowledgement

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] C. Conte, *Identifying dependencies among delays*, PhD thesis, Georg-August Universität Göttingen, 2007.

[2] B. Cule, B. Goethals, S. Tassenoy, S. Verboven, "Mining train delays", *Proceedings of the 4th International Seminar on Railway Operations Modelling and Analysis (RailRome 2011)*, Rome, Italy, 2011.

[3] W. Daamen, R.M.P. Goverde, I.A. Hansen, "Non-Discriminatory Automatic Registration of Knock-On Train Delays", *Networks and Spatial Economics*, vol. 9, no. 1, pp. 47-61, 2009.

[4] EC, "European Commission Directive 2001/14/EC of the European parliament and of the council of 26 February 2001 on the allocation of railway infrastructure capacity and the levying of charges for the use of railway infrastructure and safety certification", *Off. J. Eur. Communities* L75, pp. 29–46, 2001.

[5] A. Exer, "Rail traffic management", in: C. Bailey (ed), *European Railway Signalling*, IRSE, A&C Black, London, pp. 311-342, 1995.

[6] H. Flier, R. Gelashvili, T. Graffagnino, M. Nunkesser, "Mining Railway Delay Dependencies in Large-Scale Real-World Delay Data", in: Ahuja, R.K., *et al.* (eds.), *Robust and Online Large-Scale Optimization, Lecture Notes in Computer Science*, vol. 5868, pp. 354–368. Springer, Berlin, 2009)

[7] R.M.P. Goverde, *Punctuality of Railway Operations and Timetable Stability Analysis*, PhD thesis, TRAIL Thesis Series, no. T2005/10, Delft, 2005.

[8] R.M.P. Goverde, I.A. Hansen, "TNV-Prepare: Analysis of Dutch Railway Operations Based on Train detection Data"', In: Allan, J., Hill, R.J., Brebbia, C.A., Sciutto, G., Sone, S. (eds.), *Computers in Railways VII*, pp. 779-788, WIT Press, Southampton, 2000.

[9] R.M.P. Goverde, L. Meng, "Advanced Monitoring and Management Information of Railway Operations" *Proceedings of the 4th International Seminar on Railway Operations Modelling and Analysis (RailRome 2011)*, Rome, Italy, 2011.

[10] P. Kecman, R.M.P. Goverde, T.J.J. Van den Boom, "A model-predictive control framework for railway traffic management". *Proceedings of the 4th International Seminar on Railway Operations Modelling and Analysis (RailRome 2011)*, Rome, Italy, 2011.

[11] ProRail, *TROTS protocol – interface design description (in Dutch)*, Utrecht, (2008)

[12] UIC, *Capacity*, Leaflet Code 406 R, Union International des Chemins de Fer (UIC), 2004.

[13] www.sporenplan.nl