Process mining of train describer event data and automatic conflict identification

P. Kecman & R. M. P. Goverde
Department of Transport and Planning,
Delft University of Technology, The Netherlands

Abstract

Data records from train describer systems are a valuable source of information for analysing railway operations performance and assessing railway timetable quality. This paper presents a process mining tool based on event data records from the Dutch train describer system TROTS, including algorithms developed for the automatic identification of route conflicts with conflicting trains, arrival and departure times/delays at stations, and train paths on track section and blocking time level. Visualisations of the time-distance diagrams and blocking time diagrams support the analysis of incidents, track obstructions, disruptions, and structural errors in the timetable design.
Keywords: train describers, realisation data, route conflict, delay, blocking time diagram, timetable quality, operations performance analysis.

1 Introduction

Improving the performance of railway infrastructure and train services is the core business of railway infrastructure managers and railway undertakings. Train delays decrease capacity, punctuality, reliability and safety, and should be prevented as much as possible.

Dense railway operations require feedback of operations data to improve planning and control. Typically, train delays at stations are monitored and registered online using train detection, train describers, and timetable databases, but the accuracy is insufficient for process improvements, and in particular, delays due to route conflicts are hard to recognize from delays at stations. Accurate data on the level of track sections and signal blocks are required to gain a better understanding of the realized train paths and conflicts between them.

WIT Transactions on The Built Environment, Vol 127, © 2012 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/CR120201

Train describer records are a main source of infrastructure event data such as occupations and releases of track sections and aspect changes of signals. These infrastructure events can be matched to train number events that are also stored in these files to recover the realized train paths on track section level. Moreover, the realized blocking time diagrams can be derived by adding a process model of the signalling logic. This process mining approach has been implemented for the log files of the Dutch train describer system TROTS. The resulting railway operations performance analysis tool recovers and visualizes the realized train paths, blocking times, and path conflicts, and thus provides essential information for the analysis of railway operations that can be used for fine-tuning the railway timetable and operational processes. The tool supports both tabular output for statistical analysis, such as discussed in Goverde and Meng [1], and visualisations of the realized time-distance and blocking time diagrams with high-lighted train path conflicts.

In earlier work, Daamen *et al*. [2] developed 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*. Goverde and Meng [1] developed the tool TNV-Statistics for a detailed statistical analysis of train realisation data based on the output files of TNV-Conflict. The TNV system was recently replaced by the new train describer system TROTS which contains an essential new approach to train number steps, and this came with a new format for the log files. In particular, train number steps are no longer given with respect to a route block to a next signal, but at section level. This means that a train number step no longer 'predicts' to which signal the train is heading, as was customary with TNV, and therefore we cannot just look ahead at the signal aspect of the signal at the end of a block to identify a conflict. Therefore, the algorithms described in Daamen *et al*. [2] had to be modified in a way described in the present paper. Moreover, several improvements have been implemented such as interpolating blocking times over non-logged signals so that route conflict identification is applicable over entire corridors, including 'dark territories' with aggregated track sections. The output of the new tool has the same format as that of TNV-Conflict by which TNV-Statistics is still applicable. Other approaches to train delay data mining include Conte [3] and Flier *et al*. [4] for determining systematic dependencies between delays in Germany and Switzerland, respectively, and Cule *et al*. [5] for identifying frequent delay patterns in Belgium.

The remainder of the paper is structured as follows. The Dutch train describer log files and their data structure are explained in Section 2. Section 3 explains the main algorithm and subroutines of the tool, followed by a case study with a description of the GUI in Section 4. Section 5 gives a brief summary and presents further application of train describer data in the framework of on-going research about model-predictive railway traffic management [6].

2 Train describer systems

Train describer systems keep track of train positions based on train numbers and messages received from elements of the signalling and interlocking systems (sections, switches and signals) [7]. One of the tasks of train describers is logging the generated train number messages and the incoming infrastructure element messages, resulting in chronologically ordered lists of infrastructure and train number messages.

2.1 The Dutch train describer TROTS

In the Dutch train describer system TROTS, the train steps are recorded on the level of track sections, with both a message when a new track section is occupied by a train and when a track section is released by a train. Hence, train number step messages are coupled to track section messages.

The Dutch railway network has been divided into multiple TROTS areas which each comprise 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: timestamp, 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 timestamp, event code, train number, and a sequence of all occupied track sections. Each successive train number step message contains either a new occupied track section at the front or 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.

2.2 Shortcomings in TROTS log files

There are several issues in the TROTS log files that represent a potential source of inaccuracy and complicate performance analysis. The system architecture [8] reveals that infrastructure messages and train number step messages are generated by different components of the system which sometimes results in a significant difference between the timestamps of corresponding messages. Experiments show delays of up to seven seconds of the train number step messages. In order to avoid possible inconsistencies, the developed tool does not use the timestamps of the train number step messages but only the ones of the corresponding infrastructure element messages.

Furthermore, infrastructure messages of a signal aspect change to ‘stop’ cannot be coupled directly to any train number step or section occupation message. In order to overcome this, an additional input file is created in the form of a list of all signals together with the first section they protect by data mining the files in a pre-processing step. We use this input in the main algorithm to

identify the train number that caused the signal aspect change via the corresponding section that got occupied.

Other sources of inaccuracy are the automatic block signals on the open track which are not logged. Without intermediate logged signals, an open track between two stations looks as one block from the exit signal at the station of departure to the home signal at the station of arrival, and headway conflicts could not be identified. Moreover, open tracks may contain aggregated track sections which are occupied and released as one. We therefore defined an additional input file containing a list of automatic block signals on the open tracks together with the corresponding (aggregated) sections listed by individual sections and their lengths. If a non-logged signal is at the boundary of two (aggregated) sections then the occupation of the following (aggregated) section is used as stop aspect event time. Otherwise, three-aspect two-block signalling logic is simulated to estimate aspect changes of non-logged signals on aggregated sections. In this case, the event time of the stop aspect change is estimated by the occupation time of the corresponding protected section, which is derived as a fraction of the running time of the train over the aggregated section proportional to the ratio of the distance to the signal and the length of the aggregated section.

3 Automatic route conflict identification

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 online by sweeping through the log file. Also conflicts can be identified simultaneously by determining relationships between events. This section presents the main algorithm that passes through the TROTS log file once to reconstruct the movements of all trains that operated in the corresponding area whilst simultaneously deriving a list of all route conflicts that occurred.

The following input is used:

- TROTS log file
- Infrastructure lists (signals with protected sections, aggregated sections and lengths)
- Operational timetable
- List of platform track sections.

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

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 [2]. Each section, signal, and train in the log file is an object, attributed by a chronologically sorted list of activities. The algorithm visits all messages and for each signal, section of

train number message, the corresponding objects are updated with a time stamp and the train number (for section/signal objects) or infrastructure element id (for train objects).

The algorithm is implemented in Matlab and based on process mining, a concept of analysing processes based on event logs [9]. Blocking time theory [10] and the operational timetable provide the logic for building the process model.

3.1 The main algorithm

The main loop is initiated when the algorithm comes across a message reporting a section occupation. Figure 1 shows a flowchart of the main loop with embedded subroutines which will be explained in Section 3.2. After all objects have been updated, the first branching makes a distinction between sections protected by a signal (first section in a block) and others. 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.

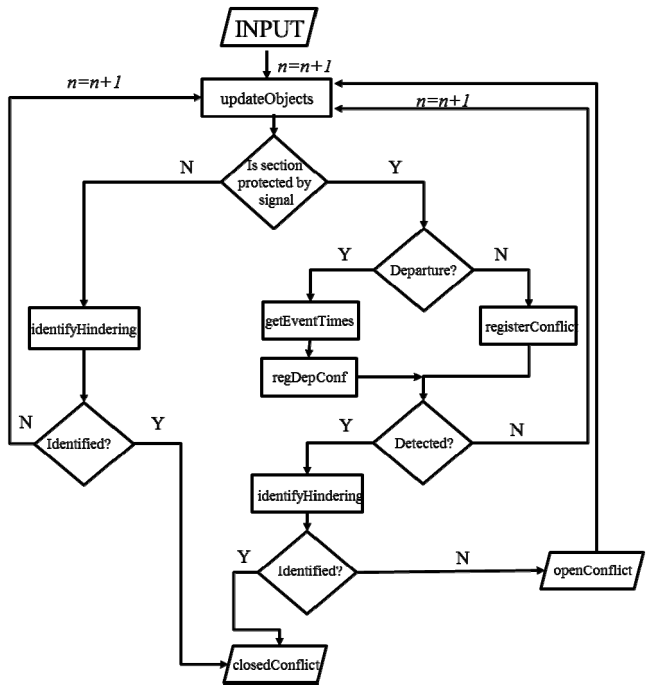

Figure 1: Main conflict identification loop.

3.2 Subroutines

This subsection gives a description of the subroutines that capture the main logic of the process mining method for automatic conflict registration. Figure 2 depicts a small part of an example network, which is used to illustrate the subroutines. The example network contains three signals (S1, S2, S3), six track sections (TS1-TS6) and a train that has just entered TS4.

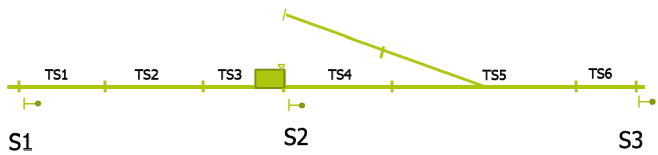

Figure 2: Illustrative example of a part of some railway network.

3.2.1 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 was at the sight distance of the signal protecting the block. When the train passes signal S2 (Figure 2), the subroutine compares the last release time (change to 'go' aspect) t_{rel}^{S2} of S2, with the time t_{sight}^{S1} that the train was at the sight distance of signal S1. For the latter time we subtract a constant sight and reaction time of 12 s from the passing time of S1 [10]. A conflict has occurred when the following condition has been met:

$$\textbf{if } t_{\text{rel}}^{S2} > t_{\text{sight}}^{S1} \rightarrow \textbf{\textit{conflict registered}} \tag{1}$$

3.2.2 Identify hindering train (*identifyHindering*)

As the hindered train progresses along the block protected by the signal of conflict (S2), *identifyHindering* compares the previous release time t_{rel}^{TSi} of each section belonging to the block (TS4, TS5, TS6 in Figure 4) with the time that the hindered train was at sight distance from the approach signal before the signal of conflict (S1, Figure 4). The train number that released the section for which (2) holds is the hindering train:

$$\textbf{if } t_{\text{rel}}^{TSi} > t_{\text{sight}}^{S1},\ i \in \{4,5,6\} \rightarrow \textbf{\textit{hindering train registered}} \tag{2}$$

3.2.3 Get event times (*getEventTimes*)

This routine derives the arrival and departure times from TROTS logfiles. When a train passes an exit signal after a scheduled stop, *getEventTimes* is initiated. It determines the period of standstill as the longest time gap between successive track section (occupied and release) messages of the relevant train. The time of the last section message before the standstill is set as the arrival time and the time of the first section message after the standstill is the departure time.

Note that the error of this arrival (departure) time estimate depends on the number of platform track sections and the distance between the stop location of the rear (front) of the train and the used section border.

3.2.4 Detect departure conflict (*registerDepartureConf*)

After the arrival and departure times have been derived, the *registerDepartureConf* 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 scheduled departure time (if the train had no arrival delay) or after the minimum dwell time has passed since the arrival time (if the train arrived with a delay).

$$\textbf{if } t_{\text{rel}}^{\text{exit}} > \max\left(t_{\text{arr}} + t_{\text{dwell}}^{\min}, t_{\text{dep}}^{\text{sched}}\right) \rightarrow \textbf{\textit{conflict registered}} \tag{3}$$

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

This section illustrates the application of the presented algorithm on one day of traffic (2nd April 2010) in the TROTS areas of The Hague and Rotterdam. The algorithm sweeps through the merged log files of the two areas and reconstructs the realized train paths of 2048 trains on the level of track sections. Moreover, all occupation times of 1396 track sections and all blocking times of 733 blocks are determined, as well as the aspect changes of 624 signals and the arrival and departure time estimates of all trains at 21 stations. Finally, 1011 route conflicts were registered.

4.1 Graphical user interface

In order to simplify the analysis of this output, a GUI was created (Figure 3). The left part of the GUI contains tabbed panels for loading data (top left panel), visualisation control (top right) and displaying results in tables (lower panel). The right part of the GUI is reserved for the visualisation of traffic in either time-distance or blocking time diagrams. The tabbed panel for loading data enables the user to either load the raw data and start the algorithm or load already processed data and display the results. In the lower tabbed panel the user can choose which results to display. In the tab *Trains* (Figure 4), a train line can be selected from the popup menu which enables selecting a train from the chosen line. We can then select the whole train path or a part of it by selecting a start and end station.

The results are then displayed in the tables on the left and the visualisation panel on the right. The selected part of the train route is visualized together with all other trains that operated on the selected corridor 15 min before and after the

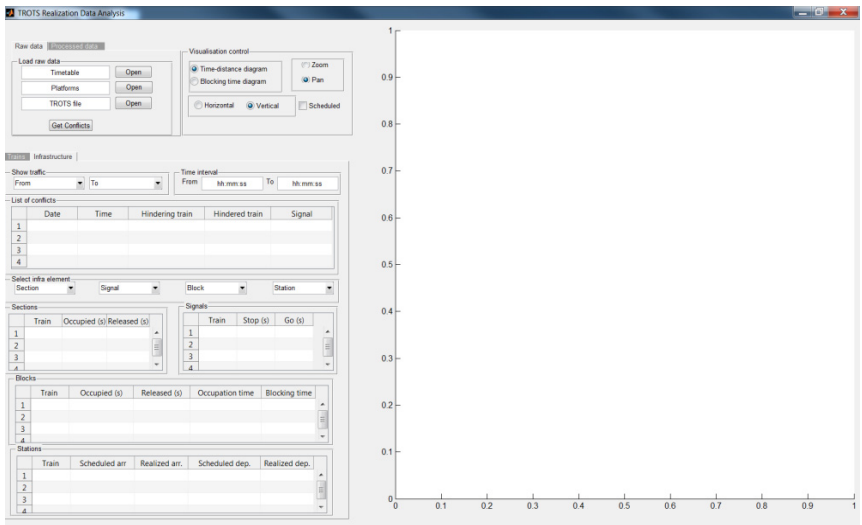

Figure 3: Graphical user interface.

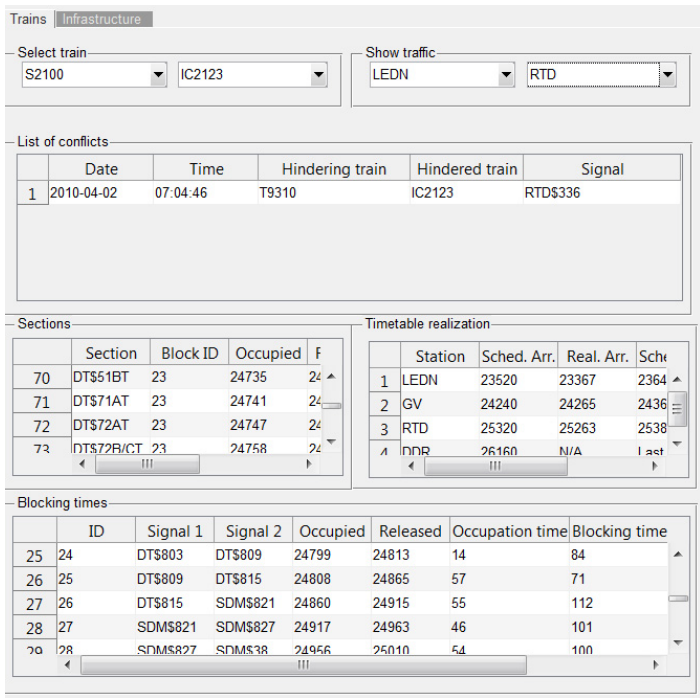

Figure 4: Train selection panel.

selected train. The tables are the list of conflicts in which the selected train participated, the running times on all sections, the blocking times, and actual arrival and departure times.

The panel *Infrastructure* (Figure 5) enables the user to choose the corridor and the time interval and get the corresponding list of conflicts, list of sections, signals, blocks, and stations that were utilized by trains on the corridor within the selected time interval. Selection of the infrastructure element from the corresponding popup menu displays all the state changes of that element with the associated train number and time instants (in seconds from midnight).

The visualisation control panel (upper right panel Figure 3) enables the user to switch between the blocking time diagram and time-distance diagram of traffic on the selected corridor and time interval. Also it is possible to turn on/off

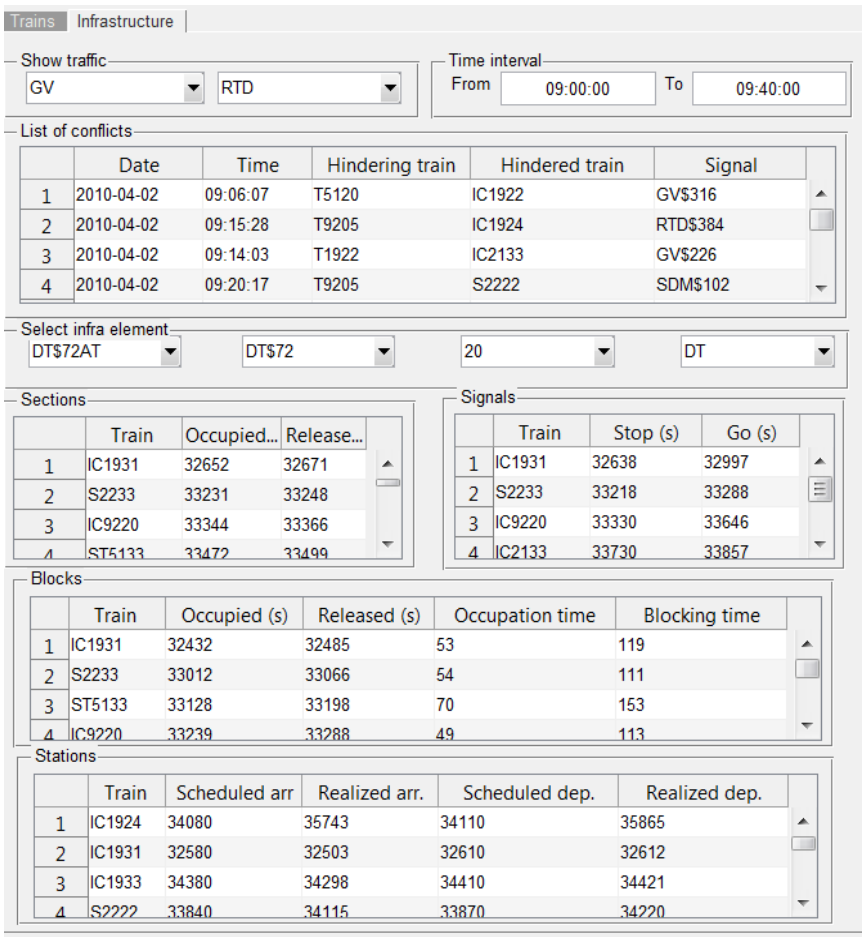

Figure 5: Infrastructure selection panel.

the zoom and pan tools and rotate the axis of the diagrams. Finally the selection of the checkbox *Scheduled* also visualises the scheduled train paths.

Figure 6 shows the time-distance diagram on the busy corridor between The Hague HS and Rotterdam in the Netherlands between 9:00 and 9:40 A.M. The number of tracks between the stations is indicated as well as the conflicts (red squares on the hindered train path at the location of the signal of conflict). Intercity trains are presented in blue colour and local trains in magenta.

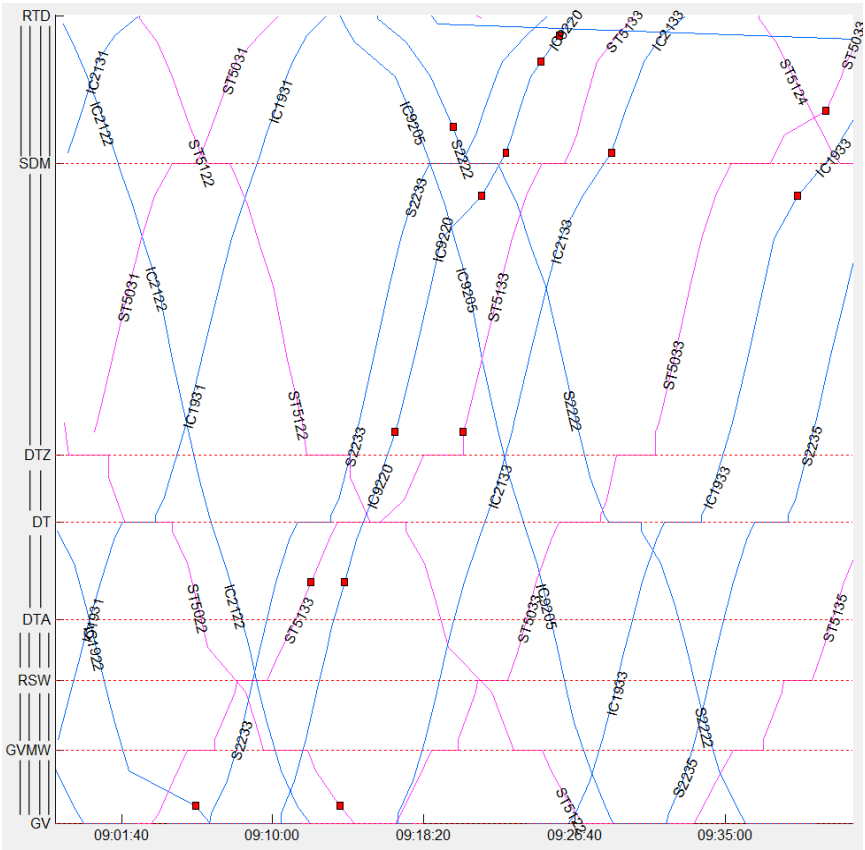

Figure 6: Time distance diagram.

Figure 7 displays the corresponding blocking time diagram for one direction that appears after selecting the appropriate radio button on the visualisation control panel. Overlaps in blocking times indicating conflicts are denoted by a red colour (or very dark in grayscale). Note that trains on parallel tracks of four-track lines may overtake each other. Blocking times that appear to be overlapping but are not shown in red (darker) are parallel processes without conflict.

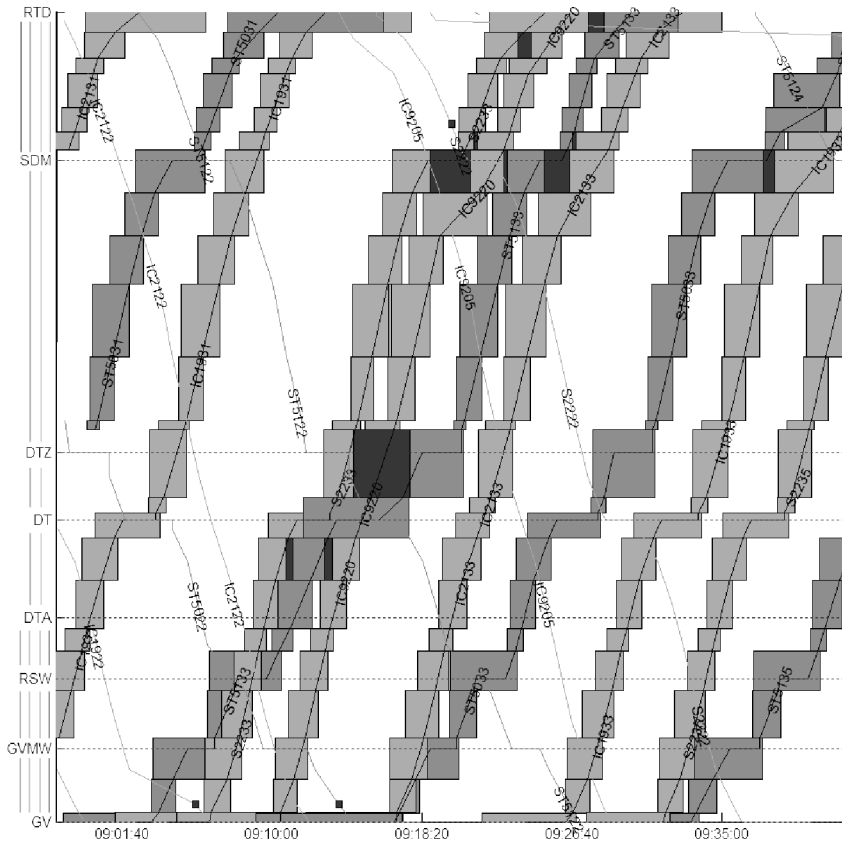

Figure 7: Blocking time diagram.

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.

Further developments are mainly directed towards automatic analysis by providing useful statistical indicators for structural flaws in the timetable, as well as detecting severe disruptions and identifying primary delays, see also Goverde and Meng [1].

Another stream of research within 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 [6]. We aim at exploiting advanced statistical and

machine learning methods to capture complex dependencies between process times in heavily utilized railway networks. The developed tool presented in this paper is the basis for this on-going work.

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] Goverde, R.M.P. and Meng, L., Advanced monitoring and management information of railway operations. *Journal of Rail Transport Planning and Management,* article in press, 2012.
[2] 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), pp. 47–61, 2009.
[3] Conte, C., *Identifying Dependencies among Delays*, PhD thesis, Georg-August Universität Göttingen, 2007.
[4] Flier, H., Gelashvili, R., Graffagnino, T. and Nunkesser, M., Mining railway delay dependencies in large-scale real-world delay data. In: R.K. Ahuja, R.H. Möhrung and C.D. Zaroliaglis (Eds.), *Robust and Online Large-Scale Optimization*, Lecture Notes in Computer Science, vol. 5868, Springer: Berlin, pp. 354–368, 2009.
[5] Cule, B., Goethals, B., Tassenoy, S. and Verboven, S., Mining train delays. *Proceedings of the 4th International Seminar on Railway Operations Modelling and Analysis (RailRome 2011),* Rome, Italy, 2011.
[6] Kecman, P., Goverde, R.M.P. and Van den Boom, T.J.J., 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.
[7] Exer, A., Rail traffic management. In: C. Bailey (Ed.), *European Railway Signalling*, IRSE, A and C Black: London, pp. 311–342, 1995.
[8] ProRail, *TROTS protocol – interface design description (in Dutch),* Utrecht, 2008.
[9] Van der Aalst, W.M.P., *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Springer: Heidelberg, 2011.
[10] Hansen, I.A. and Pachl, J. (Eds.), *Railway Timetable and Traffic: Analysis, Modelling, Simulation*, Eurailpress: Hamburg, 2008.