Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 37 (2019) 227–234

Transportation Research Procedia

www.elsevier.com/locate/procedia

21st EURO Working Group on Transportation Meeting, EWGT 2018, 17th – 19th September 2018, Braunschweig, Germany

Mining railway traffic control logs

Felix Mannhardt*, Andreas D. Landmark

SINTEF Digital, P.O. Box 4760 Torgarden, NO-7465 Trondheim, Norway

Abstract

Railway traffic is a set of interrelated processes that are centrally controlled. Despite optimized train schedules, train dispatchers still take ad-hoc decisions on the scheduling of trains in the context of unplanned events. Train orders are swapped, train crossings on single-tracks are moved, or trains are cancelled to minimize the disruption in the schedule. The actual scheduling of trains, as decided by dispatchers and observed through the movement of trains across stations, is then registered in railway traffic control logs. Using this data that contains information on the tacit knowledge of dispatchers can help to evaluate strategies for dealing with disruptions, which have not been subject to upfront planning. This paper proposes to use process mining methods, which are commonly applied in the context of business processes, to expose the hidden process of how the train traffic was actually dispatched. Different variants of dispatching are juxtaposed with the total delay in the railway system to visually explore the dispatching strategies taken. The technique has been implemented as a prototype and validated on a large dataset of real-life traffic in the Norwegian railway system.

Keywords: Traffic control logs, Process mining, Traffic disruptions, Railway

1. Introduction

Unlike most other transport modes, railway operation is a completely controlled traffic system. The location of every entity in the system is planned and controlled in order to maintain the safety and integrity of the system. This makes scheduling and traffic management key activities not only for safety, but also to provide a high level of service to the end-user. This is usually operationalised in punctuality and regularity. While safety and integrity is paramount, punctuality and regularity are central quality metrics both from the user- and the operator-perspectives. Punctual traffic may also be both a contributor to safe and secure execution of train schedules.

* Corresponding author. *E-mail address:* felix.mannhardt@sintef.no

10.1016/j.trpro.2018.12.187

There is a large body of research on providing (near-)optimal train schedules and operational real-time scheduling (Lamorgese and Mannino, 2015). However, the execution of the schedule is usually manually influenced for a variety of reasons, both in terms of causing variation in the execution, but also intentional improvisation for dealing with situations and events, such as avoiding dead-locks and other unwanted situations in traffic (Lium, 2013). Unplanned events and situations introduce limitations that did not exist at the time of planning which forces adjustments to plans (e.g., infrastructure and rolling stock malfunctions, passenger-related incidents, and inclement weather (Olsson and Haugland, 2004)). For the dispatcher, this means negotiating options such as moving crossings (for single-track traffic), swapping the order of trains, delaying or even cancelling trains to minimize consequences.

Railway traffic can be viewed as a set of interrelated processes and is often analysed as such. Mining empirical models of processes based on actual traffic patterns can inform schedulers on how dispatchers execute their plans when faced with real-life situations. Conversely, it may also provide feedback to dispatchers on the various heuristics used in negotiating variations in traffic. The automatic discovery and analysis of processes based on their actual execution — process mining — is an emerging research field in the context of workflow and business processes management (van der Aalst, 2016). Process mining typically considers processes in which sequences of work re-occur, e.g., an insurance claim process. Separate process instances are started for each incoming claim and each instance consists of a sequence of activities. Process mining re-discovers behavioural models of such processes based on execution data.

This paper proposes the application of process mining techniques on railway traffic control event logs together with performance indicators, e.g., the daily accumulated delay on a railway line that can be derived from the data. Our goal is to investigate the quality of ad-hoc decisions that are taken by railway dispatchers in the light of unplanned events. We evaluated several options of how to automatically discover a model of the decisions taken based on actual railway traffic data from Norway. Based on our study, we propose to take a station-centric view on the dispatch decision[1]. We consider the daily sequence of trains passing a station as one instance of the dispatching process in contrast to a process-view in which each train ride would be seen as one process instance, which was previously taken in the application of process mining to train data (Kecman and Goverde, 2012; Janssenswillen et al., 2017).

This paper is structured as follows. In Section 2, we explore related work. In Section 3, we present the proposed method along with the necessary process mining concepts and in Section 4 we show results of applying our method to data from the Norwegian railway network. Section 5 concludes the paper with an outlook of future work.

2. Related work

The application of process mining methods to historical railway traffic data, which records the process of trains moving in the railway system, seems a natural fit. Process mining could be a valuable tool that helps dispatchers in handling traffic and guide their actions based on historical information in the case of unplanned events. Despite this, only very little research has been conducted on taking such a process-mining view on railway systems.

Data from train describer system has been used to analyse the railway timetable quality and performance based on a process mining approach in The Netherlands by Kecman and Goverde (2012). Another application was the exploratory analysis of train re-routings in Belgium based on the complexity of discovered process models by Janssenswillen et al. (2017). Cule et al. (2011) used a frequent episode algorithm to analyse knock-on delays in the Belgian railway network. Similarly, to our work, Cule et al. also consider the crossing of trains at a single station, but only discover frequent episode instead of a full process model. Flier et al. (2009) detected systematic delay dependencies between individual trains in the context of finding knock-on delays. While a lot has been done on statistical analysis of train system performance (including minimization techniques for knock-on delays, increasing capacity utilisation under satisfactory punctuality, and so on), there are also some studies on the development and application of heuristics in train dispatching. As a good example, Corman et al. (2014) provide an extensive discussion of challenges between global coordination and local dispatching and the performance of various

[1] Note that with station we denote any kind of crossing or point in the railway system for which the time of passing trains are recorded.

heuristics to handle disturbances accurately. As they point out, further research is necessary to fully understand the challenge of handling disturbances, and amongst others disrupted traffic situations and alternative time tables.

3. Mining railway traffic control event logs

We define preliminaries required to describe our approach, then we present the method used.

3.1. Event logs

An event log $L = (E, \#)$ stores information about the events that occurred within a process. Each event occurs at a specific timestamp. Event logs can be seen as tables in which each row corresponds to the occurrence of an event. In this paper, we consider events logs that record the arrival and departure times of trains on stations in the railway network, i.e., railway traffic control event logs, which are commonly stored by the railway network operators. Each event in an event log is identified by a unique identifier $e \in E$ and writes a set of attributes that can be obtained through the attribute function $\#$. Typically, railway traffic control logs, such as shown in Table 1, at least write the following standard attributes for each event:

- train, an identifier of the train causing the event;
- time, the time of occurrence;
- day, the day of occurrence;
- station, the physical location of the event;
- type, a qualifier whether the event is about the arrival or departure of the train.

Process mining imposes two further requirements on event logs. Each event has to be assigned an activity identifier, i.e., what caused the recording of the event, and a case identifier, i.e., to which instance of the process the event belongs (van der Aalst, 2016). Given a set of activities A and a set of process cases C, we need to define a function $act: E \rightarrow A$, an activity classifier, and a function $case: E \rightarrow C$, a case classifier, to be able to use the event log for process mining. Often, it is clear what attributes define the activity and case notion for an event log, but it is also possible to define multiple different classifiers for a single event log (IEEE Computational Intelligence Society, 2016) to accommodate different process views on an event log.

Table 1. An example railway traffic control event log taken from the Norwegian railway network.

id	train	day	Time	schedule	station	type	direction
*e*1	407	2017-12-04	2017-12-04 06:30:43	2017-12-04 06:30:00	OPD	arrival	up
*e*2	407	2017-12-04	2017-12-04 06:47:52	2017-12-04 06:45:00	OPD	departure	up
*e*3	407	2017-12-04	2017-12-04 06:53:27	2017-12-04 06:51:30	FGH	arrival	up
*e*4	407	2017-12-04	2017-12-04 06:53:49	2017-12-04 06:52:00	FGH	departure	up
*e*5	407	2017-12-04	2017-12-04 07:02:49	2017-12-04 07:00:30	UBG	arrival	up
*e*6	407	2017-12-04	2017-12-04 07:03:25	2017-12-04 07:01:00	UBG	departure	up
*e*7	407	2017-12-04	2017-12-04 07:11:19	2017-12-04 07:08:00	BAK	arrival	up
*e*8	407	2017-12-04	2017-12-04 07:13:15	2017-12-04 07:09:00	BAK	departure	up
*e*9	42	2017-12-04	2017-12-04 09:30:03	2017-12-04 09:29:00	BAK	arrival	down
*e*10	42	2017-12-04	2017-12-04 09:31:54	2017-12-04 09:30:00	BAK	departure	down
*e*11	42	2017-12-04	2017-12-04 09:39:49	2017-12-04 09:38:30	UBG	arrival	down
*e*12	42	2017-12-04	2017-12-04 09:40:30	2017-12-04 09:39:00	UBG	departure	down
*e*13	42	2017-12-04	2017-12-04 09:49:19	2017-12-04 09:47:30	FGH	arrival	down
*e*14	42	2017-12-04	2017-12-04 09:49:50	2017-12-04 09:48:00	FGH	departure	down
*e*15	42	2017-12-04	2017-12-04 09:55:28	2017-12-04 09:55:00	OPD	arrival	down
*e*16	42	2017-12-04	2017-12-04 09:59:21	2017-12-04 09:58:00	OPD	departure	down
*e*17	41	2017-12-04	2017-12-04 12:45:13	2017-12-04 12:43:30	DRS	arrival	up
*e*18	41	2017-12-04	2017-12-04 12:45:37	2017-12-04 12:44:00	DRS	departure	up
	…	…	…	…	…	…	

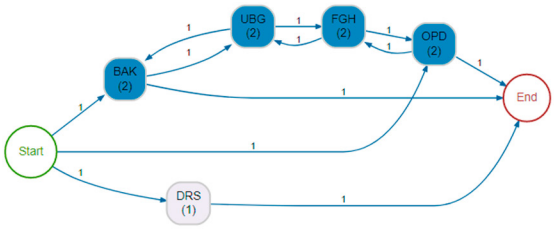


Figure 1. Process map discovered from the example railway traffic control log when using each train attribute as case classifier.

Given a case classifier, it is possible to obtain from the event log a set of execution traces T. We denote the sequence of events as trace to emphasize that the recorded events only indirectly refer to the process execution. Each trace $\sigma_c \in E^*$ is a sequence of events $\langle e_1, \dots, e_n \rangle$ such that:

1. events in the same trace map to the same case identifier $c \in C$,
2. each event appears exactly once in a trace, and
3. the events of a trace are totally ordered by their time of occurrence.

Formally, we define the set of trace for an event log $L = (E, \#)$ as:

$$T_L = \{\sigma_c \mid \sigma_c = \langle e_1, \dots, e_n \rangle \wedge \forall_{1 \le i \le j \le n} (\text{case}(e_i) = \text{case}(e_j) = c \wedge e_i \neq e_j \wedge \#(e_i, \text{time}) < \#(e_j, \text{time}))\}. \quad (1)$$

For example, based on the excerpt in Table 1 it is possible to form the following traces: $\sigma_{407} = \langle e_1, e_2, \dots, e_8 \rangle$, $\sigma_{42} = \langle e_9, e_{10}, \dots, e_{16} \rangle$, and $\sigma_{41} = \langle e_{17}, e_{18} \rangle$ when using the train attribute as case notion. Alternatively, when using the day attribute there would be only one trace: $\sigma_{2017-12-04} = \langle e_1, \dots, e_{18} \rangle$.

3.2. Process maps

Process maps are used to understand the control-flow of processes by visualizing the ordering relations between a set of activities A. The activities of a process are represented as boxes and directed edges between activities represent a directly-follows relation for activities in the same process instance. A process map is a tuple (A_L, D_L) with A_L being the activities observed in event log L and D_L being the directly-follows relations between activities:

$$D_L = \{(a, b) \in A_L \times A_L \mid \exists_{e_i, e_j \in E} (\text{act}(e_i) = a \wedge \text{act}(e_j) = b \wedge \langle e_1, \dots, e_i, e_j, \dots, e_n \rangle \in T_L)\}. \quad (2)$$

The set of directly-follows relations can be computed in one pass through the event log in time linear to the number of events and quadratic to the number of activities (Leemans et al., 2018). In our use case there are only a limited number of distinct activities (e.g., trains or stations) and, thus, process maps can be computed efficiently. For example, in Figure 1 we build a process map of the example log by assuming a case notion based on the train attribute and an activity classifier based on the station. Also, we only keep the arrival events of the event log to build the process map. Thus, the edges of the process map indicate which stations in the railway network precede and follow each other based on observed trains. We add artificial start and end activities to each trace to ensure unique endpoints, which often improves the understandability of the process map (Mendling et al., 2012).

3.3. Mining method

As motivated, a choice regarding the case and activity classifiers needs to be made for the application of process mining. In the railway context, an obvious choice for the case would be to consider each train ride as a separate process instance. Like the example in Figure 1 cases are classified based on the train attribute and the day of travel. Choosing this case notion yields the well-known structure of the railway network as result. When adopting such case notion, it is natural to choose the station (i.e., location) of the train as activity. This might be useful when projecting

frequency or time statistics on the process map to uncover delays and traffic patterns. However, the railway network is well-known and, thus, discovering follows relations between stations yields little surprises.

We propose to use a different case notion that helps to uncover deviations in the actual dispatching of trains as decided on-the-spot by train dispatchers. We divide the data into process instances across the days of operation ($\forall_{e \in E}\ \mathrm{case}(e) = \#(e, \mathrm{day})$) and consider the identifiers of the actual trains passing a single station as activities ($\forall_{e \in E}\ \mathrm{act}(e) = \#(e, \mathrm{train})$). Since the traffic patterns are re-occurring each (working) day the model is expected to be stable across several days. Moreover, it may highlight differences in the choice made by dispatchers on the order in which trains pass the station. Using this case notion, we obtain Figure 2 when using an event log for 6 months in 2017 that contains all trains passing the station Fagerhaug (FGH) on weekdays. Having created a process map as in Figure 2, it is possible to project additional information on the edges and nodes of the process map. For example, the occurrence frequency of a specific train in process instances is projected on the activities by counting the number of events related to that activity. Similarly, the observation count of directly-following trains in the event log is shown on the edges. In our example, train 407 was observed in 84 of the cases and was normally followed (175 times) by train 42, but once it was followed by train 41 instead.

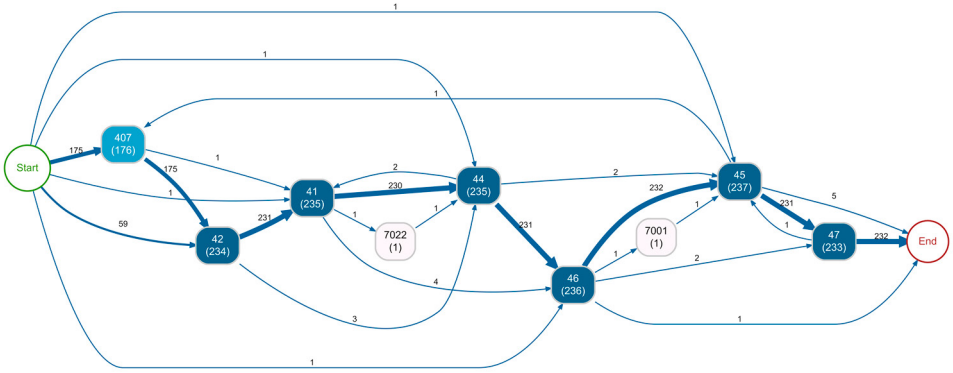


Figure 2. Process map uncovering the actual train dispatching for the station Fagerhaug during 6 months in 2017. The case classifier uses the day attribute to create cases and the train attribute is used by the activity classifier.

Whereas deviations from the planned schedule, which generally coincides with the most frequent path, are of interest by themselves, railway traffic control event logs often contain more information that can be projected on a process map. One potentially interesting performance indicator for the actual dispatching of trains is the average overall delay in the railway system. We can compute the overall delay accumulated by those trains that pass the station in question from the event log. Here, we leverage that railway traffic control event logs typically contain both the scheduled and actual times for arrival and departure. Then, given the traces of an event log organized with a train-based case notion ($\forall_{e \in E}\ \mathrm{case}(e) = \#(e, \mathrm{train})$) the delay that a specific train accumulated during its trajectory can computed from its trace in the event log. Given trace $\sigma_x = \langle e_1, \ldots, e_n \rangle$ and a special start event e_0 with $\#(e_0, \mathrm{time}) = \#(e_0, \mathrm{schedule}) = 0$, we define:

$$\mathrm{delay}_{\sigma_x} = \sum_{1 \le i \le n} \max\left(0,\ \begin{array}{l} \max\big(0, \#(e_i, \mathrm{time}) - \#(e_i, \mathrm{schedule})\big) - \\ \max\big(0, \#(e_{i-1}, \mathrm{time}) - \#(e_{i-1}, \mathrm{schedule})\big) \end{array}\right) \quad (3)$$

Note that there are many ways to calculate the delay of a specific train. For example, we consider all delay accumulated and do not account for too early trains or catching up of trains. So, this definition is only one possible way that is used to illustrate our method.

By projecting this information on the edges of the process map, it is possible to quickly judge the co-occurrence of large system delays with alternative schedules / dispatch strategies. The process view helps to visualize the alternative dispatch strategies at a glance. It also allows the analyst to investigate the coordination/dispatching decisions taken on the day (or under certain disturbances) and discuss or evaluate their quality. This information

could also be useful input for planning or simulation activities. It is also possible to project any other measure that can be associated to the cases (day) or the activities (train). For example, it could make sense to look at the time between trains and the waiting times of trains at the station. Or, under specific circumstances, link them to more extrinsic factors such as passenger load, weather conditions, and other factors known to cause systemic disturbances (Olsson and Haugland, 2004).

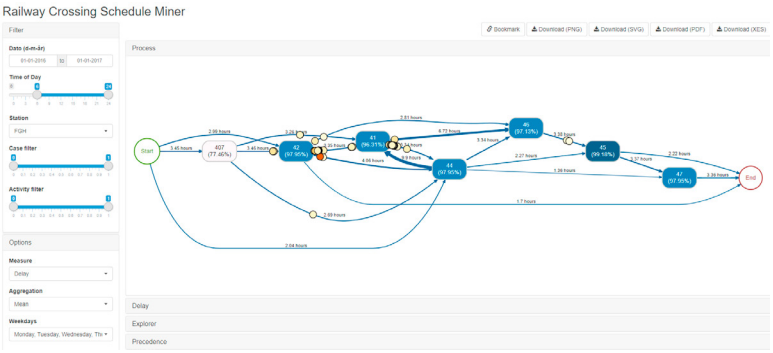
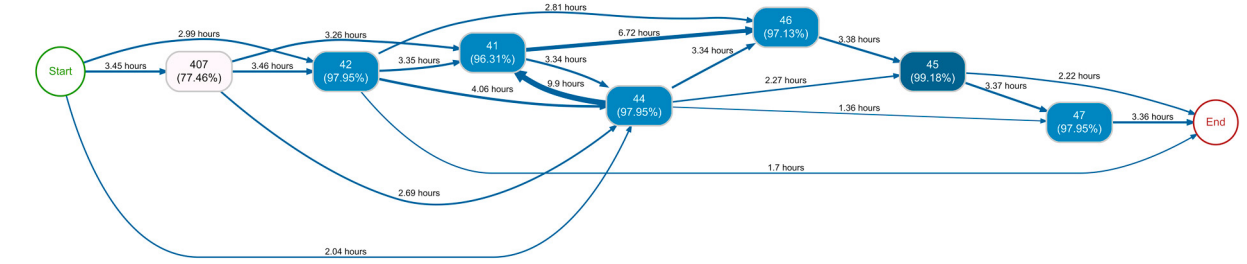


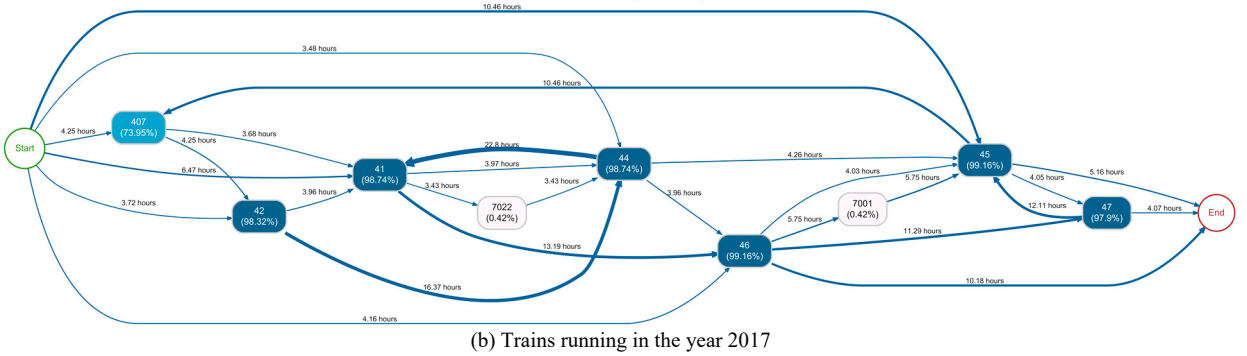
Figure 3. Interface of the web application that automatically mines process maps and overlays the total daily delay for trains crossing a station.

4. Results and discussion

The data source for this research is an online database of train movements for the Norwegian railway. It includes the scheduled and actual arrival and departure times of trains for all stations and junctions in the Norwegian infrastructure like the example in Table 1. Overall this database can be viewed as an event log with more than 120 million events, from which we extracted subsets of events of interest. Figure 3 shows a web-application that we developed based on the process mining library bupaR (Janssenswillen and Depaire, 2017), which can be used to explore the train traffic for all stations in the Norwegian railway network.



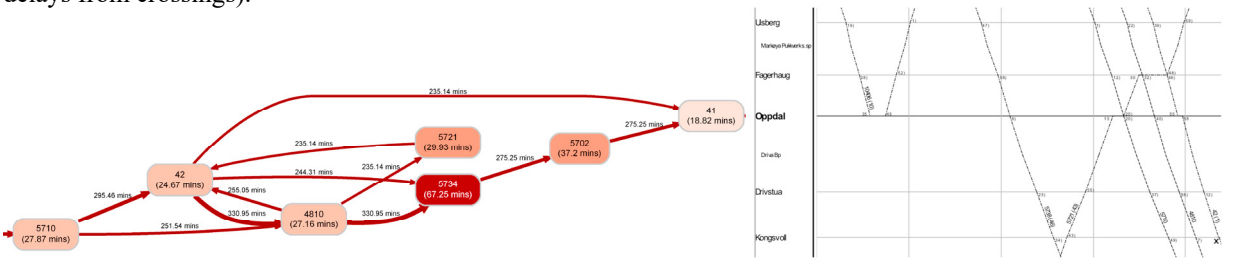
(a) Trains running in the year 2016



(b) Trains running in the year 2017

Figure 4. Process map for the railway traffic crossing station Fagerhaug on weekdays. We project the relative frequency of trains passing the station on the nodes of the process map and the average daily delay for all cases that follow an edge on the edges.

Using this application, we conducted an analysis of the planned schedule and the actual dispatch of trains at Fagerhaug station (FGH). Figure 4 shows process maps of the actual dispatch and accumulated delay for passenger trains in the years 2016 and 2017, respectively. Together both event logs contained about 5800 events and it took less than 5 seconds to extract them and create the process maps. There are differences in the actual dispatch strategies. An interesting dispatching challenge on this stretch, is in-frequent and routinely out of sequence (due to late departure) freight trains. These trains will routinely have to be dispatched manually with crossings moved between junctions to minimize total delay and maintain punctuality for on-time trains (i.e. minimizing knock- on delays from crossings).



(a) Excerpt of a process model for Fagerhaug. Total delay annotated on edges. (b) Excerpt of the schedule for Fagerhaug

Figure 5. Example of an analytical case using the developed tool

Figure 5a shows an except in which we filtered out the very rarely and the very frequently occurring traces from the 2017 event log (between the 10th and 50th percentile) and included freight trains. Here, the amount of total delay for the involved trains (as shown on the edges) is differs in the various schedule alternatives. Simple heuristics such as pushing priority traffic through and leaving out-of-order trains to wait is not necessarily the optimal strategy for coordinating the larger traffic picture. What we can see here is that after letting train 5710 run through, the dispatcher is routinely having to deal with disturbances where the ordering of 42 or 4810 becomes an operational decision after the scheduled 5721 runs through with crossings at Oppdal and Fagerhaug (where it is supposed to defer to the higher prioritized 42). Under certain conditions the dispatcher re-prioritizes and runs 5721 further north before crossing with 42 when 42 is running behind. However, this tool allows for quick inspection of the consequences of the various decisions. Adding other performance metrics to the tool, allows for evaluating the various goals that the dispatcher has to optimize for.

4.1. Future work

In further work, there is both process-mining methodological work possible as well as more analytical angles on rail traffic. In the latter, the possibility to step through various performance metrics beyond what we have implemented is obvious. It will also be necessary to be able to order various predecessors that will provide hints to causality for the discovered process models. In our limited example, it is obvious for the trained reader that the crossing of 5738 and 5721 south of Figure 5b will be a common cause for some of the process variety.

From a methodological standpoint, automatic qualification of interesting process variations is an interesting avenue of research. This could be guided by further work into quantifying the various goals of the dispatcher (i.e. the balance-act of minimizing delay, prioritization, executable plan, etc.). Potentially it would be possible to discover schedule alternatives that mitigate common disturbances and enumerate and learn from these best-practices.

There are also some limitations to the used process mining method when trains cross stations in the opposite direction in overlapping times. Instead of discovering that both trains pass the station in parallel, the observed interleaved sequence ($A < B$ and $B < A$) are depicted. Detecting such parallelism from event logs is a central challenge in process mining and in future work, we plan to apply process mining algorithms based on heuristics (Weijters and Ribeiro, 2011; Mannhardt et al., 2017) to discover such crossings and visualize them in a more compact manner. Another possible research topic worthwhile pursing would be the automatic ranking of the good or bad schedule adaptations based on the discovered processes or the application of conformance checking methods

(van der Aalst et al., 2012; Leemans et al., 2018) that takes the process model of the planned schedule as input and diagnose differences.

5. Conclusion

There are multiple possible applications of process mining on railway traffic control event logs. Previous approaches focused on using the train trajectory as process instance. We take a station-centric case notion, in which the activities of the process correspond to trains passing a specific station and each day of traffic is a process instance. Taking this view, we discover process maps based on the sequence in which trains cross a particular station. Under the assumption of a stable train schedule, the process discovered in this manner should be a sequence of trains. However, dispatcher often need to change the order of trains based on traffic disruptions and unplanned events. Thus, the actual traffic, as observed in the event log, often deviates from the plan and a process map with multiple possible schedules is discovered. We applied the method on two years of railway traffic control event logs from the Norwegian railway network. Based on the obtained process visualizations, we analysed how the different schedule adaptations are related to performance indicators such as the total delay accumulated per day for the trains that are part of the visualization.

Acknowledgements

This work was in part supported by Bane NOR.

References

van der Aalst, W.M.P., 2016. Process Mining - Data Science in Action, Second Edition. Springer. doi:10.1007/978-3-662-49851-4.

van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F., 2012. Replaying history on process models for conformance checking and performance analysis. WIREs Data Min Knowl Discovery 2, 182–192. doi:10.1002/widm.1045.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2014. Dispatching and coordination in multi-area railway traffic management. Computers & Operations Research 44, 146–160. doi: 10.1016/j.cor.2013.11.011.

Cule, B., Goethals, B., Tassenoy, S., Verboven, S., 2011. Mining train delays, in: Advances in Intelligent Data Analysis X. Springer Berlin Heidelberg, pp. 113–124. doi:10.1007/978-3-642-24800-9_13.

Flier, H., Gelashvili, R., Graffagnino, T., Nunkesser, M., 2009. Mining railway delay dependencies in large-scale real-world delay data, in: Robust and Online Large-Scale Optimization. Springer. volume 5868 of LNCS, pp. 354–368. doi:10.1007/978-3-642-05465-5_15.

IEEE Computational Intelligence Society, 2016. IEEE standard for extensible event stream (XES) for achieving interoperability in event logs and event streams. doi:10.1109/IEEESTD.2016.7740858. IEEE Std 1849-2016.

Janssenswillen, G., Depaire, B., 2017. bupar: Business process analysis in r, in: BPM 2017 Demos, CEUR-WS.org.

Janssenswillen, G., Depaire, B., Verboven, S., 2017. Detecting train reroutings with process mining. EURO Journal on Transportation and Logistics doi:10.1007/s13676-017-0105-8.

Kecman, P., Goverde, R.M.P., 2012. Process mining of train describer event data and automatic conflict identification, in: Computers in Railways XIII, WIT Press. doi:10.2495/cr120201.

Lamorgese, L., Mannino, C., 2015. An exact decomposition approach for the real-time train dispatching problem. Operations Research 63, pp. 48–64. doi:10.1287/opre.2014.1327.

Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P., 2018. Scalable process discovery and conformance checking. Softw Syst Model 17, pp. 599–631. doi:10.1007/s10270-016-0545-x.

Lium, A.G., 2013. Hvordan gjennomfres togledelse i Norge? Technical Report SINTEF A23665. SINTEF.

Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P., 2017. Data-driven process discovery - revealing conditional infrequent behavior from event logs, in: CAiSE 2017, pp. 545–560. doi:10.1007/978-3-319-59536-8_34.

Mendling, J., Snchez-Gonzlez, L., Garc, F., Rosa, M., 2012. Thresholds for error probability measures of business process models. J Syst Softw 85, pp. 1188–1197. doi:10.1016/j.jss.2012.01.017.

Olsson, N.O., Haugland, H., 2004. Influencing factors on train punctuality results from some norwegian studies. Transport policy 11, 387–397.

Weijters, A.J.M.M., Ribeiro, J.T.S., 2011. Flexible heuristics miner (FHM), in: CIDM 2011, IEEE. IEEE. pp. 310–317. doi:10.1109/cidm.2011.5949453.