Probabilistic Simulation of a Railway Timetable

Rebecca Haehn
RWTH Aachen University, LuFG THS, Germany
https://ths.rwth-aachen.de
haehn@cs.rwth-aachen.de

Erika Ábrahám
RWTH Aachen University, LuFG THS, Germany
abraham@cs.rwth-aachen.de

Nils Nießen
RWTH Aachen University, VIA, Germany
niessen@via.rwth-aachen.de

Abstract

Railway systems are often highly utilized, which makes them vulnerable to delay propagation. In order to minimize delays timetables are desired to be robust, a property that is often estimated by simulating the respective timetable for different deterministic delay values. To achieve an accurate estimation under consideration of uncertain delays many simulation runs need to be executed. Most established simulation systems additionally use microscopic models of the railway systems, which further increases the simulations running times and makes them applicable rather for small areas of interest for complexity reasons.

In this paper, we present a probabilistic, symbolic simulation algorithm for given timetables, this means we do not simulate individual executions, but all possible executions at once. We use a macroscopic model of the railway infrastructure as input. This way we consider the railway systems in less detail but are able to examine certain performance indicators for larger areas. For a given input model this simulation computes exact results. We implement the algorithm, examine its results, and discuss possible improvements of this approach.

2012 ACM Subject Classification Computing methodologies → Model development and analysis

Keywords and phrases Railway, Modeling, Scheduling, Probabilistic systems, Optimization

Digital Object Identifier 10.4230/OASIcs.ATMOS.2020.16

Funding Supported by the German Research Council (DFG) – Research Training Group 2236 (UnRAVeL).

Acknowledgements We are grateful to Deutsche Bahn for supporting us with data.

1 Introduction

Railway traffic has increased over the past years and is expected to increase further [11]. However, changes in the railway infrastructure to accommodate the increase in traffic are expensive and take a long time. Therefore, it is increasingly important to optimize the exploitation of the infrastructure capacity. As many passenger and freight trains as possible should be able to use the infrastructure, of course in compliance with the necessary safety requirements. At the same time the quality of service should still be satisfying. Unfortunately, with increasing traffic volume delay propagation is increasing, too. That is the case because the intervals between consecutive trains are smaller for a higher traffic volume, which makes it more likely that one train's delay impacts other trains' punctuality as well.

There are different approaches to still ensure an acceptable quality of service [12]. On the one hand, they aim at minimizing the *primary delays*, caused for example by malfunctions like signal or brake faults or by large numbers of passengers. On the other hand, there

20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020).
Editors: Dennis Huisman and Christos D. Zaroliagis; Article No. 16; pp. 16:1–16:14
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

are measures to reduce the *secondary delays*, resulting from conflicts with other trains and therefore indirectly from primary delays. These measures can be roughly divided into two categories, improving the *robustness* of the timetable (planning) [7, 14] and improving *train dispatching* (execution) [8, 9, 18]. In this paper we propose an approach to examine the robustness of a given timetable under consideration of probabilistic primary delays. By robustness we mean here that the trains in the timetable should be able to recover from small delays and that delay propagation in the timetable execution should be restricted, as defined in [5].

Simulation is a technique often used to assess railway timetable robustness. There is a variety of different simulation systems, varying for example in the level of details considered and the simulation type used. Some use *microscopic* models which describe railway systems in great detail, others are based on *macroscopic* models which are less detailed. For further information on railway models see [16]. Microscopic models are well suited to achieve precise simulation results for small areas of interest. However, for larger areas microscopic simulations are not feasible since the computations are too complex. For that use case macroscopic models are preferable.

The systems RailSys [4, 17] and OpenTrack [3, 15] use microscopic models and simulate all trains *synchronously* in discrete time steps, such that all train operations are simulated in a single run. In contrast, *asynchronous* simulation simulates train operations in a series of runs starting with the highest priority trains. The system LUKS [1, 13] also uses microscopic models but proceeds in an asynchronous/synchronous combination. Macroscopic models are used e.g. in MOSES/WiZug [19], which proceeds asynchronously and is used specifically for rail freight transportation.

The above mentioned systems have in common, that they implement *Monte Carlo* simulation. They consider primary delays as deterministic variables and conduct a large number of deterministic simulation runs with different random primary delay values. That is quite time consuming, because many different simulation runs have to be executed in order to achieve an accurate result. Another approach, presented in [6] and implemented in the software OnTime [2, 10], uses *analytical* procedures to compute delay propagation instead of Monte Carlo simulation. In that work the input timetables are modeled mesoscopic as activity graphs while the delays are represented as distribution functions. In contrast to this work, we focus more on the infrastructure utilization and therefore use a macroscopic model of the infrastructure network instead of an activity graph. Also, we discretise the random variables representing the delay. Another difference is that in [6] primary delay can occur at any time on a train's path, while we only consider primary delays at the beginning of each train ride for now. Also changes in the train sequence are considered more explicitly in [6]. In both approaches rerouting is neglected as an option to dispatch delayed trains.

The novel contribution of this paper is a probabilistic simulation algorithm for given timetables using a macroscopic model of railway infrastructure and synchronous simulation. Our method allows to examine the timetables robustness by evaluating performance indicators such as the expected time of arrival for each train. Additionally, we can identify infrastructure elements that increase the expected delay and evaluate for each infrastructure element in the macroscopic network the expected utilization over time. A strong advantage of our approach is that, in contrast to Monte Carlo simulation, it provides *exact* results. We want to make clear that our approach differs fundamentally from Monte Carlo simulation in that no individual possible execution sequences are calculated, but rather a symbolic one, representing all possible execution sequences. Limitations of our approach are that we (1) discretise the random variables representing the delay, (2) only consider primary delays that occur at the start of each train ride and (3) neglect rerouting as a possibility to reduce delays. A short-term future work will aim at relaxing (2), as our approach is easily extensible to handle also general primary delays. Relaxing (1) would be possible using integration, but we would probably lose exactness. Relaxing (3) is currently not planned, as our focus lies on analysis. We implemented and evaluated the presented approach using existing German railway infrastructure networks and timetables.

We describe the model of railway systems that we use in Section 2 and present our probabilistic simulation approach in Section 3. We proceed in Section 4 with a detailed experimental evaluation and conclude the paper in Section 5.

2 Railway Systems

A *railway system* consists of an infrastructure network and a corresponding timetable. There are different ways for modeling railway systems. One decision to be made is the level of detail. *Microscopic* models describe railway systems in great detail, they contain for example all signals and switches. This has the advantage that calculations on such models are quite accurate. However, those models are very complex and get large even for small parts of a railway network. *Macroscopic* models are less detailed, they disregard signals and exact routes inside stations. In this paper we model railway systems on a macroscopic level.

Infrastructure Network

An *infrastructure network* is a directed graph $G = (V, E)$, with a set of vertices V that represent the *operation control points (OCPs)* and a set of directed edges $E \subseteq \{(v, u) \in V \times V \mid v \neq u\}$ representing the *tracks* between different OCPs that can be used in the corresponding direction. Each infrastructure element $x \in V \cup E$ is annotated with a *capacity* value $cap : V \cup E \to \mathbb{N}$. For vertices $v \in V$, $cap(v)$ is defined as the number of stopping points at the corresponding OCP or one if stopping is not possible there. $cap(v)$ the maximal number of trains not just dwelling at v but also passing through v. For edges $e \in E$, $cap(e)$ is the number of parallel tracks available in the given direction between the respective source and target OCPs. Currently we model bi-directional tracks (which can be used in both directions) with capacity c by two separate tracks, one in each direction and both having capacity c, thereby over-approximating the available resources[1]. We assume that all $cap(v)$ resp. $cap(e)$ tracks of a vertex v resp. edge e are equivalent in the sense that they could replace each other. In the following we also neglect whether stopping is not intended at some infrastructure elements. Despite the simplifying assumptions we made, we expect this model to reflect the real conditions to a sufficient extent.

Timetable

Time is modeled discretely in minutes within a predefined finite time horizon, yielding a time domain $\mathbb{T} = [t_{min}, t_{max}] \subset \mathbb{N}$. A (finite non-timed) *path* in an infrastructure network $G = (V, E)$ is a sequence $(v_1, v_2, \ldots, v_k)$ of nodes $v_i \in V$, $i \in \{1, \ldots, k\}$ connected through edges $(v_i, v_{i+1}) \in E$ for all $i \in \{1, \ldots, k-1\}$. A (finite) *timed path* $\pi = (v_1(a_1 \mapsto d_1), \ldots, v_k(a_k \mapsto d_k))$ in G is a path in G annotated with arrival and departure times $a_i, d_i \in \mathbb{T}$ for all $i \in \{1, \ldots, k\}$ and $a_1 \leq d_1 \leq a_2 \leq ... \leq d_k$. A *timetable* for G is a set of trains $\{train_1, \ldots, train_n\}$,

[1] We are currently working on an extension of our method to handle bi-directional tracks without such an over-approximation.

where each train $train_{id} = (type_{id}, \pi_{id})$ is specified by its type $type_{id}$ (in Germany e.g. ICE, RE) and a timed path $\pi_{id} = (v_{id,1}(a_{id,1} \mapsto d_{id,1}), \ldots, v_{id,k_{id}}(a_{id,k_{id}} \mapsto d_{id,k_{id}}))$ in G for $id \in \{1, \ldots, n\}$. Note that we do not explicitly specify the length of tracks and the speed of trains, but model them implicitly by specifying the time $a_{id,j+1} - d_{id,j}$ needed for train $train_{id}$ to pass the track $(v_{id,j}, v_{id,j+1})$. Let in the following $T = \{train_1, \ldots, train_n\}$ be an *executable* timetable, meaning that in the absence of uncertainties and delays, for each time point $t \in \mathbb{T}$ and each node $v \in V$, the number of trains that are in v at time t is at most $cap(v)$, i.e. $|\{id \in \{1, \ldots, n\} \mid \exists j \in \{1, \ldots, k_{id}\}.v_{id,j} = v \wedge a_{id,j} \leq t \leq d_{id,j}\}| \leq cap(v)$, and similarly $|\{id \in \{1, \ldots, n\} \mid \exists j \in \{1, \ldots, k_{id}\}.v_{id,j} = v \wedge v_{id,j+1} = v' \wedge d_{id,j} \leq t \leq a_{id,j+1}\}| \leq cap(e)$ for each edge $e = (v, v') \in E$.

3 Simulation

Simulation can be used to analyse complex real-world systems. It requires an executable model of the real-world system, typically described in terms of *states* and *events*, such that the execution of the model approximately imitates the real system's behaviour. In this paper, we simulate a railway timetable execution over time on a corresponding infrastructure network by virtually letting trains run through the network, where states encode trains being at certain stations and events encode trains moving through the infrastructure network.

There are different types of simulations that could be applied in our context. Without considering uncertainties, *deterministic* simulation is suitable to check whether a timetable is executable as planned (correctness). However, if we want to analyse timetable execution realistically, we need to consider uncertainties causing delays which might be further propagated due to capacity restrictions.

Suitable approaches for this are *stochastic* and *probabilistic* simulations, the best known is the Monte Carlo method. In these approaches, states and events may be uncertain, e.g. the input is not precisely known or some random behaviour occurs. Such uncertainties are modeled by stochastic (continuous) or probabilistic (discrete) distributions over the value domains. The Monte Carlo method executes a model several times with randomly generated values for uncertain model parameters and computes probability distributions to describe the observable system behaviour. Stochastic/probabilistic simulation can be used in our context to examine the robustness of timetables for different delay scenarios. A disadvantage of the Monte Carlo method is that for reliable results it needs a high number of runs.

3.1 Probabilistic Simulation

Due to the aforementioned restrictions of deterministic simulation approaches and the Monte Carlo method, in this paper we implement a *probabilistic* simulation. In contrast to Monte Carlo simulation, our approach executes a *single run* and computes all random outputs symbolically in an *exact* manner. Currently we only consider initial delays (i.e., delays for the departure times $d_{id,1}$) and propagated delays caused by them, but do not consider any further random events that affect the system. Especially, we assume that trains have no additional random delay while already on their way, but consider only the intermediate delays that are caused by initial delays. We explicitly represent these initial uncertainties by specifying inputs as discrete probability distributions over a certain domain of possible delay times, defined manually based on observations.

Since the inputs are uncertain, the outputs are uncertain as well. That means, the results of the analysis, based on inputs represented by probability distributions, are themselves probability distributions. To compute these probabilistic outputs, we use discrete-time simulation. That means we simulate discrete time steps iteratively, updating the state variable values at a finite number of points in time. In the following we describe this approach in detail. For technical reasons, we first need a slight extension of the input model introduced in Section 2. We first describe this extension and define the states and events that are used to describe the system, before we present the simulation algorithm in Section 3.2.

Input

In Section 2 we modeled railway systems as directed graphs $G = (V, E)$ with a corresponding timetable $T = \{train_1, \ldots, train_n\}$, $train_{id} = (type_{id}, \pi_{id})$ for $id \in \{1, \ldots, n\}$. For the simulation we need a slight extension of this model: we add two auxiliary vertices *source* and *target* to the set of vertices $V' = V \cup \{source, target\}$ and auxiliary edges from *source* to all vertices in V and from all vertices in V to *target*, resulting in $E' = E \cup \{(source, v), (v, target) \mid v \in V\}$ and $G' = (V', E')$. The capacities of the auxiliary infrastructure elements are arbitrarily large, so we extend cap to $cap' : V' \cup E' \to \mathbb{N}$ with $cap'(x) = cap(x)$ for all $x \in V \cup E$ and $cap'(x) = n$ for all other x. The idea is that virtually, all trains $train_{id}$ start in *source*, move to their initial node $v_{id,1}$, complete their original route in $v_{id,k_{id}}$ and move to *target* afterwards. The reason for this extension is that a node's capacities might be exceeded when a train would start in it, so in order to be able to cleanly insert starting trains we let them start in *source* at their planned starting time and move on to their initial node as soon as capacities allow. In addition, we avoid that trains block capacities once they have completed their routes; in practice they move on to another route at a time point specified by $d_{id,k_{id}}$, which we model by moving to *target*. Each timed path $\pi_{id} = (v_{id,1}(a_{id,1} \mapsto d_{id,1}), \ldots, v_{id,k_{id}}(a_{id,k_{id}} \mapsto d_{id,k_{id}}))$ in the timetable is extended accordingly to $\pi_{id}' = (v_{id,0}(a_{id,0} \mapsto d_{id,0}), v_{id,1}(a_{id,1} \mapsto d_{id,1}), \ldots, v_{id,k_{id}}(a_{id,k_{id}} \mapsto d_{id,k_{id}}), v_{id,k_{id}+1}(a_{id,k_{id}+1} \mapsto d_{id,k_{id}+1}))$ with $v_{id,0} = source$, $a_{id,0} = d_{id,0} = a_{id,1}$, $v_{id,k+1} = target$ and $a_{id,k+1} = d_{id,k+1} = d_{id,k}$. Let $T' = \{train_1', \ldots, train_n'\}$ with $train_{id}' = (type_{id}, \pi_{id}')$ for $id \in \{1, \ldots, n\}$.

States

Each train $train_{id}$ has the state set $S_{id} = S_{id}^V \cup S_{id}^E$ with $S_{id}^V = \{(v_{id,j}, epdt) \mid 0 \leq j \leq k_{id} + 1 \land epdt \in \mathbb{T} \land epdt \geq d_{id,j}\}$ and $S_{id}^E = \{((v_{id,j}, v_{id,j+1}), epdt) \mid 0 \leq j \leq k_{id} \land epdt \in \mathbb{T} \land epdt \geq a_{id,j+1}\}$. A state $(x, epdt) \in S_{id}$ encodes that train i is currently using the infrastructure element $x \in V' \cup E'$ and will not release it before the *earliest possible departure time epdt*. A train's movement is modeled by a sequence of random variables $(X_{id}^t)_{t \in \mathbb{T}}$ over its state set, where $P(X_{id}^t = s)$ for $t \in \mathbb{T}$ is the probability with which X_{id}^t has the value $s = (x, epdt)$ at time t. Initially at time t_{min}, each train $train_{id}$ is in node *source*, where the probability values $P(X_{id}^{t_{min}} = (source, epdt))$ are defined by an input distribution such that $P(X_{id}^{t_{min}} = (x, epdt)) > 0$ only for $x = source$ and $epdt \geq d_{id,0}$. Our aim is to compute the probabilities $P(X_{id}^t = s)$, $s \in S_{id}$ for all trains $id \in \{1, \ldots, n\}$ and time points $t \in \mathbb{T} \setminus \{t_{min}\}$.

In order to compute these probabilities we simulate the infrastructure behaviour and maintain for each infrastructure element $x \in V' \cup E'$ a set $at[x]$ of occupiers of type `Occupier` and a set $blocked[x]$ of blockers of type `Blocker` that currently use that infrastructure's capacities. The data type `Occupier`$= \{id, j, epdt, p\}$ is used to encode that with probability p the train $train_{id}$ resides at the given infrastructure element, which is the j-th vertex resp. edge[2] in its path, with earliest possible departure time $epdt$. The data type `Blocker`$= \{id, u, p\}$

[2] This information does not only reduce frequent searches of next steps in timetables but it is essential if timed paths in the timetable may visit an infrastructure element more than once.

encodes *blocking times*: with probability p, the train with id id has already left the respective infrastructure element but due to safety zones, it is still blocking the element until time u. Initially we set $at[source] = \{(id, 0, epdt, p) \mid P(X_{id}^{t_{min}} = (source, epdt)) = p > 0\}$, $at[x] = \emptyset$ for all $x \in V \setminus \{source\}$, and $blocked[x] = \emptyset$ for all $x \in V' \cup E'$.

An occupier with earliest possible departure time in the past represents a train waiting for free resources. Therefore, at a fixed time point $t \in \mathbb{T}$, for all train ids $id \in \{1, \ldots, n\}$ and infrastructure elements $x \in V' \cup E'$, all occupier entries $(id, j, epdt, p) \in at[x]$ with $epdt \leq t$ are equivalent in the sense that the train id is in x and is ready to depart if resources are available. Therefore, for each given train we *merge* all such entries and consolidate their probabilities. Technically, let I be the set of all entries $(id, j, epdt, p)$ in an occupier set $at[x]$ at time point t with the same id and $epdt \leq t$. Then we replace all these entries by a single entry $(id, j, t, \sum p_i)$. This merging strongly reduces the number of considered train states with non-zero probabilities and will have a major impact on efficiency.

Events

Next we define the updates of occupier and blocker sets from time point $t-1$ to t. Let $x \in V' \cup E'$ be an infrastructure element and $(id, j, epdt, p) \in at[x]$ at time point $t-1$. Let y be the infrastructure element that directly follows x in the path of $train_{id}$ i.e. $y = (v_{id,j}, v_{id,j+1}) \in E'$ if x is the node $v_{id,j} \in V'$ and $y = v_{id,j+1} \in V'$ otherwise (if x is the edge $(v_{id,j}, v_{id,j+1}) \in E'$). Then the train transitions to y with a certain probability $p_y \in [0,1] \subset \mathbb{R}$ and it remains in x one more time unit with the remaining probability $p_x = 1 - p_y$.

If $epdt > t$ then the earliest possible departure time lies in the future and the train id stays at x with probability $p_x = 1.0$ (and thus $p_y = 0$). Otherwise, the train transits to y iff there is free capacity not needed by higher-priority trains, i.e. p_y is the probability of free capacity and $p_x = 1 - p_y$. In this latter case, we compute p_y in the following two steps.

First, for $i = 1, \ldots, cap(y)$ we compute the probabilities p_i that at least i tracks are available in y at $t-1$. Let m be the number of different trains that use or block resources in y with positive probabilities at time point $t-1$. If $m < cap(y)$ then p_1 to $p_{cap(y)-m}$ are 1.0. For all remaining cases $cap(y) - m < i \leq cap(y)$ the probability p_i is the sum of the probabilities for exactly $k \in \{0, \ldots, cap(y) - i\}$ trains to use or block tracks at y. To compute these probabilities, we add for all k-combinations of the m trains the product of the probabilities with which they are at y or blocking y and the other $m - k$ trains are neither. The probability for a train id to be at y or block y is $\sum_{(id,.,.,p) \in at[y]} p + \sum_{(id,.,p) \in blocked[y]} p$. Note that this technically assumes that the random variables are statistically independent. However, due to the large number of interdependent variables the dependencies between the variables are disregarded here.

Second, we collect all trains that compete for resources in y in the current step and prioritize them. Here we use the data type `Request`$= \{x, id, j, epdt, p\}$, whose values encode the competitors $(id, j, epdt, p) \in at[x]$ which are willing to transit to y. Requests are then sorted by type, where requests with a smaller type (corresponding for example to long-distance passenger trains) have a higher priority. For requests with the same type those with an earlier planned arrival time have higher priority. Should that not be sufficient to define a clear order, the ids are used as conclusive criterion. Assuming we have an ordered set of requests $\{req_1, \ldots, req_{m'}\}$ to arrive in y at time t, where req_1 has the highest priority. Then req_1 arrives at y with probability p_1, req_r with probability p_r for $1 < r \leq \min\{cap(y), m'\}$ and all other requests, if there are any, with probability 0.0.

Listing 1 Probabilistic simulation

```
1  void simulate(Infrastructure G=(V,E), Timetable T) {
2    Occupier at[][];
3    Blocker blocked[][];
4    initialize();
5
6    for(Time t = t_min, t <= t_max, t++) {
7      for each v in V { simulate_vertex(v, t); }
8      for each e in E { simulate_edge(e, t); }
9    }
10 }
```

To model the train remaining in x, if $p_x > 0$ then we include $(id, j, \max\{epdt, t\}, p \cdot p_x)$ in $at[x]$ at time t. To model transit to y, if $p_y > 0$ then we include $(id, j', epdt', p \cdot p_y)$ in $at[y]$ at t, where the infrastructure element index j' of y in π'_{id} is j if $x \in V'$ and $j+1$ otherwise (as the j-th node is followed by the j-th edge and the j-th edge is followed by the $(j+1)$-st node).

As to the value $epdt'$ of the earliest possible departure time, let a_y, d_y be the planned arrival and departure times at y. If we enforce that trains spend at least the time planned in the timetable at each infrastructure element then we would have $epdt' = t + (d_y - a_y)$. However, in reality additional buffer times are included in the timetable to be able to make up for past delays if necessary. In this paper we assume that the driving times can be reduced by up to 5%. For the waiting times we make the assumption that for passenger trains they can be reduced to three minutes, while for freight trains they can be reduced to ten minutes. However, trains are not allowed to be earlier than planned. This means that for $y \in E'$ we have $epdt' = \max\{d_y, t + 0.95 \cdot (d_y - a_y)\}$, and for $y \in V'$ we have $epdt' = \max\{d_y, t + stop\}$ with $stop = \min\{(d_y - a_y), 3\}$ for passenger trains and $stop = \min\{(d_y - a_y), 10\}$ for freight trains (according to $type_{id}$).

To model blocking, for each infrastructure element x we start from the set $blocked[x]$ at time $t-1$, remove all entries (id, u, p) with $u < t$, and for each entry $(id, j, epdt, p)$ added to $at[y]$ for a request from x we also add an entry $(id, t+u, p)$ to $blocked[x]$, where u is a pre-defined blocking duration (in our experiments 2 minutes).

3.2 Algorithm

The main algorithm is shown in Listing 1. First, the required variables need to be initialized, then the actual simulation can be executed in lines 6-9. For each time step all vertices and edges are simulated, shown for the vertices in Listing 2. After unblocking the vertex, see line 2, the requests from all incoming edges are collected after combining the respective occupiers in lines 4-16. Next the probabilities with which the requests arrive are computed, see line 18. Afterwards the requests can actually be scheduled, as described in Section 3.1 and shown in Listing 3. To schedule a request with a certain probability p the corresponding occupiers probability is multiplied with $1-p$, see lines 16, 17, unless the probability would get below a certain threshold, in which case the occupier is deleted in line 7. This is done in the implementation to avoid time-consuming arithmetic computations. For exact results exact arithmetic would be necessary, however, the results are still accurate for a sufficiently small threshold. Next a new occupier is added for the current infrastructure element with the probability of the old occupier multiplied with p, see lines 18-20, respectively 8-10, while the previous infrastructure element is blocked with that probability. Under the assumption that the random variables are statistically independent this algorithm is correct

Listing 2 Vertex simulation

```
void simulate_vertex(Vertex v, Time t) {
  unblock(blocked[v], t); // delete all Blocker with u < t

  ordered_set<Request> req;
  for each e = (v',v) in E {
    // merge Occupiers (id,j,t1,p1), (id,j,t2,p2)..
    // ..to (id,j,t,p1+p2) when t1,t2 <= t
    merge(at[e], t);
    for each (id,j,epdt,p) in at[e] {
      if(epdt <= t) {
        Time arr = T[o.id][o.j+1].a; // planned arrival time at v..
        // ..needed for sorting
        req.insert(Request(v', arr, id, j, epdt, p));
      }
    }
  }

  double cap[] = capacities(blocked[v], at[v], v.capacity, req.size());

  int i = 0;
  for each r in req {
    ++i;
    if(i > v.capacity || cap[i] == 0.0) { break; }
    // train is no longer at incoming edge e with probability r.p*cap[i]
    Edge e = (r.prev, v);
    schedule_request(r, r.p*cap[i], t, at[e], blocked[e], at[v], v);
  }
}
```

Listing 3 Request processing

```
void schedule_request(Request r, double p, int t, Occupier atCurrent[],
    Blocker blocked[], Occupier atNext[], Vertex v) {
  Occupier o = Occupier(r.id, r.j, r.epdt, r.p);

  // r is scheduled with probability 1.0..
  // ..(or probability with which it stays gets too small)
  if(p == 1.0 || (r.p*(1-p) < 0.00001)) {
    atCurrent.erase(o);
    blocked.insert(Blocker(r.id, t + tb, r.p));
    Time epdt = std::max(T[r.id][r.j+1].d, t + stop);
    atNext.insert(Occupier(r.id, r.j+1, epdt, r.p));
  } else if(r.p*p < 0.00001) {
    return; // the probability that r arrives is too small
  } else {
    // r arrives with probability r.p*p..
    // ..and stays with probability r.p*(1-p)
    atCurrent.erase(o);
    atCurrent.insert(Occupier(r.id,r.j,r.epdt,r.p*(1-p)));
    blocked.insert(Blocker(r.trainID, t + tb, r.p*p));
    Time epdt = std::max(T[r.id][r.j+1].d, t + stop);
    atNext.insert(Occupier(r.id, r.j+1, epdt, r.p*p));
  }
}
```

Table 1 Railway systems - infrastructure network and timetable properties.

Network	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T'_1\rvert$	$\lvert T'_2\rvert$	$\lvert T'_3\rvert$
N	2646	5622	436	566	1254
N_SO	5146	11028	858	1221	2521
W_M_SW	6635	14510	1509	1924	4474

(up to rounding).

4 Experimental Results

The algorithm defined in Subsection 3.2 was implemented in C++. Therefore, probabilities need to be represented. Theoretically probabilities are in the interval $[0, 1] \in \mathbb{R}$. Since computers can not easily represent reals and very small probabilities have no practical relevance we use high-precision floating point values. We restrict the precision of probabilities to 10^{-5}. That means we round probabilities smaller than that to be zero. Due to imprecision in the multiplication especially of small values, some intermediate results might be smaller than 0.0 or larger than 1.0 (by less than 10^{-5}). Those values are set to the respective limit.

For the experiments we used a computer with a 1.80 GHz $\times$ 8 Intel Core i7 CPU and 16 GB of RAM. To evaluate the algorithm we used real-world railway infrastructure networks. All of those have been generated from confidential infrastructure data in XML form, provided by DB Netz AG (German Railways). These data include many details that are not required for the infrastructure model used in this paper, therefore, we abstracted from the given data to extract the required input networks. Table 1 shows some properties of the networks: the second resp. third column lists the number of vertices resp. edges.

Time was modeled discretely in minutes, as mentioned in Section 2, with a time step size of one minute. In order to convert the time values in the given timetables consistently, we decided that a day is modeled as the time interval $[0, 1440]$ with 0 representing midnight. We considered different time intervals $\mathbb{T} = [t_{min}, t_{max}] \subset \mathbb{N}$ that are defined accordingly instead of starting with 0:

- $\mathcal{T}_1 = [480, 540]$, from 08:00 am to 09:00 am (1 hour)
- $\mathcal{T}_2 = [60, 360]$, from 01:00 am to 06:00 am (5 hours, during the night)
- $\mathcal{T}_3 = [720, 1020]$, from 12:00 am to 05:00 pm (5 hours, during the day)

For the infrastructure networks corresponding feasible timetables T_i extended with source and target for the considered time intervals were given. The remaining three columns in Table 1 show the number of trains contained in these timetables. This gives a rough idea about the utilization, despite the train lines containing varying numbers of stations. Like the infrastructure networks the timetables are based on the DB data. In order for the timetables to match the networks' level of detail we had to slightly modify the given timetable data as well.

In the following we first take a look at the computational efficiency and the running times of the algorithm, then we analyse and discuss the results.

4.1 Running Times of the Algorithm

In order to execute the implemented probabilistic simulation two more parameters need to be decided. The blocking time t_b between two consecutive trains is approximated with two minutes. For the required initial delay distributions we use a geometric distribution

Table 2 Running times (in *sec*) for different versions of the implementation.

Input	`basic`	`on-demand`	`min-prob`
N with its T_1	2,5	2,1	1,5
W_M_SW with its T_1	9,4	6,6	5,0
N with its T_2	11,1	8,6	4,9
W_M_SW with its T_2	45,0	32,8	14,7

and let each train depart with probability 0.8 at every time step. With the discretisation of time and the lower bound on the precision this results in the delays i with probability p_i for $(i, p_i) \in \{(0, 0.8); (1, 0.16); (2, 0.032); (3, 0.0064); (4, 0.00128); (5, 0.00026); (6, 0.00005); (7, 0.00001)\}$. Later we evaluate the impact of the initial delay distribution on the simulation results in more detail, however, for the examination of the running times we do not change them to get better comparability.

We examine the running times only for the networks N and W_M_SW with their corresponding timetables T_1 and T_2, because we are mainly interested in the impact of certain modifications on the running time. The running times for different versions of the implementation are shown in Table 2. The first version, here referred to as `basic`, is just a straight-forward implementation of the algorithm described in Subsection 3.2. The running times for that version on the first three inputs are essentially acceptable, however, we should keep in mind that these are relatively small inputs. And for even just slightly larger inputs, as for example network W_M_SW with its timetable T_2, the simulation already takes more than four times as long. We realized that in every time step for each infrastructure element is checked whether some state changes, but often (for vertices in over 61% of the time steps, for edges even in 89%) nothing changed. Therefore, for the second version `on-demand`, we took into consideration whether state changes for infrastructure elements might occur in any given time step. This shows some improvements, the version `on-demand` was 19% to 42% faster for all inputs.

Next we examined whether some parts of the algorithm could be parallelized. Unfortunately, most sub-procedures work on common data. However, the computations of the capacity probabilities for the vertices and for the edges respectively might be performed in parallel. We will exploit this in future work.

Finally, we took a look at the number of `Occupiers`. Despite the possibility to combine some `Occupiers` this number is increasing over time. That is due to the fact that when a train can arrive at its next infrastructure element with a probability $p < 1.0$ often an additional `Occupier` is added to the simulation. Another effect of this is, that some `Occupiers`' probabilities become (maybe negligibly) small. In the last version `min-prob` we therefore additionally restricted the probabilities to values larger or equal to 0.01. This reduced the number of `Occupiers`, as shown in Figure 1 and further reduced the running times, however, it should also be considered how this impacts the result. When that level of accuracy is sufficient this is useful to reduce the running times.

4.2 Evaluation

As mentioned before not just the running times but mainly the actual results of the simulation are of interest. In this section we examine the impact of the initial delay distribution on the simulation. Therefore we consider geometric distributions with different success probabilities p that a train departs. The following distributions are used as initial delay distributions:

- $p = 0.9$: 0, 0.9; 1, 0.09; 2, 0.009; 3, 0.0009; 4, 0.00009; 5, 0.00001

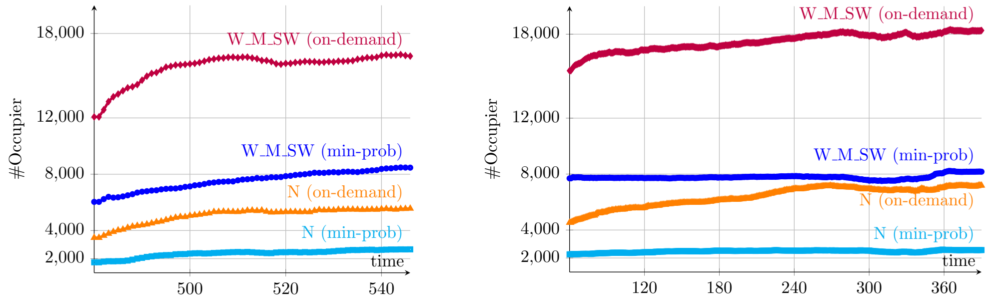

Figure 1 Number of `Occupiers` for input N and W_M_SW with T_1 (left) and T_2 (right), red and orange represent the version `on-demand`, cyan and blue version `min-prob`.

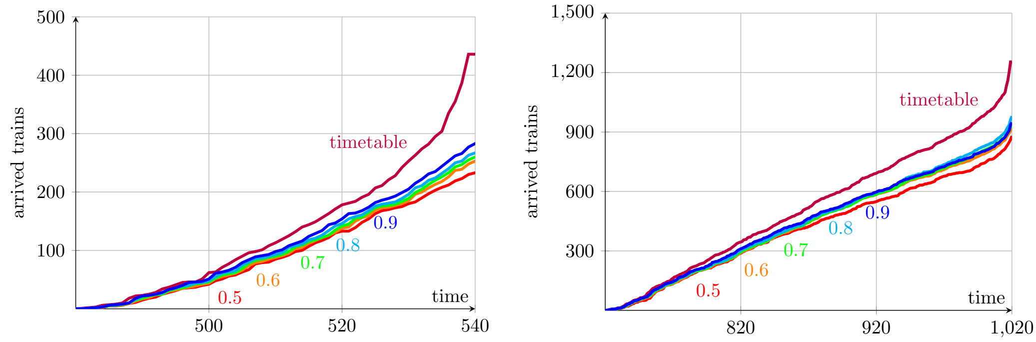

Figure 2 Number of trains that arrived at their target for infrastructure N, on the left with timetable T_1, on the right with timetable T_3.

- $p = 0.8$: 0, 0.8; 1, 0.16; 2, 0.032; 3, 0.0064; 4, 0.00128; 5, 0.00026; 6, 0.00005; 7, 0.00001
- $p = 0.7$: 0, 0.7; 1, 0.21; 2, 0.063; 3, 0.0189; 4, 0.00567; 5, 0.0017; 6, 0.00051; 7, 0.00015; 8, 0.00005; 9, 0.00002
- $p = 0.6$: 0, 0.6; 1, 0.24; 2, 0.096; 3, 0.0384; 4, 0.01536; 5, 0.00614; 6, 0.00246; 7, 0.00098; 8, 0.00039; 9, 0.00016; 10, 0.00006; 11, 0.00003; 12, 0.00002
- $p = 0.5$: 0, 0.5; 1, 0.25; 2, 0.125; 3, 0.0625; 4, 0.03125; 5, 0.01563; 6, 0.00781; 7, 0.00391; 8, 0.00195; 9, 0.00098; 10, 0.00049; 11, 0.00024; 12, 0.00012; 13, 0.00006; 14, 0.00003; 15, 0.00002; 16, 0.00001

Since the timetables cover some time interval $[t_{min}, t_{max}]$ and we examine delay, we extend the time interval by 10% of its duration, e.g. instead of $[480, 540]$ we examine $[480, 546]$. First we take a look at the number of trains that arrive at their target with a probability of 1.0. For the network N with the timetable T_1 and the previously mentioned extended interval, that number is between 235 and 285 (out of 436) for the different initial distributions and increases for larger p as expected. The number of trains that arrived at their target as a function of the time is depicted in Figure 2, for the simulation results we used the expected arrival time instead of the planned arrival time. The overall difference between the initial distributions is quite small, the curves are shaped very similarly. It is important to note that despite extending the considered time interval not all trains arrive at their target. This is important for our further examinations.

Additionally to the number of trains arriving at their targets until a certain time step, the expected delays at the target are of interest. These are depicted in Figure 3 for the same

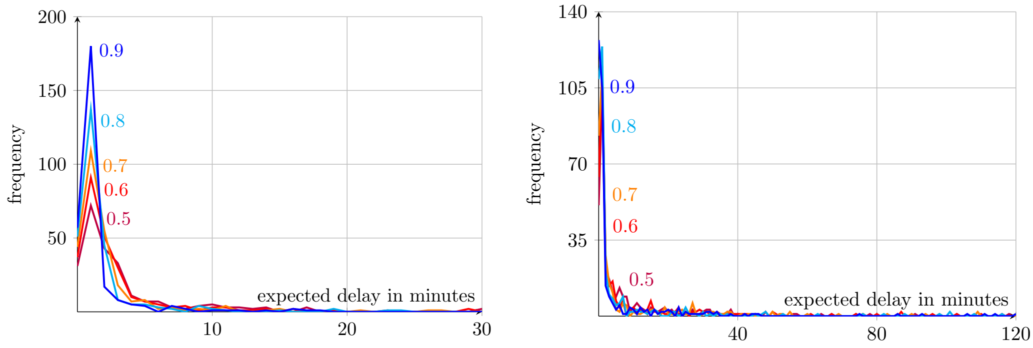

Figure 3 Frequency of expected delays at the targets (in minutes) for infrastructure N, on the left with timetable T_1, on the right with timetable T_3.

input scenarios as used above. The majority of trains is expected to be punctual, with either no expected delay or just one to two minutes. For smaller p expected delays of up to ten, respectively 20 minutes, are still quite common, and a few trains are even expected to be up to 30 minutes delayed in the scenario for the extended interval T_1. This is quite exceptional considering this interval only has a duration of just over an hour. Such exceptionally long expected delays correspond to longer train paths with respect to both duration and number of vertices. These trains tend to be impacted by a larger amount of other trains and are therefore more exposed to secondary delay. For the timetable T_3 the maximal expected delay is even worse with over two hours. In reality the affected trains would be rerouted or cancelled, which our approach does not allow.

So far we mostly evaluated the given timetables, however, as mentioned before, we explicitly chose to model the railway system based on an infrastructure network, as opposed to an activity graph for example, in order to also evaluate the individual infrastructure elements. One possible metric for this is the average change in delays while utilizing a certain infrastructure element. The change in delays for a given infrastructure element is defined as the difference of the trains' delays when departing from and arriving at the element, weighted by the corresponding probabilities. Due to different numbers of trains utilizing the infrastructure elements these values are not comparable, yet, and should be scaled using the total number of trains utilizing the respective element. The fact that not all trains reach their target in the simulated time interval makes it problematic to compute this metric for all infrastructure elements. On some there are still trains with certain probabilities that simply did not depart yet and for which therefore the change of their delay is not known yet. In order to avoid inconsistent results, we only computed this metric for the infrastructure elements that were no longer occupied by any train.

We visualized this metric, for the infrastructure elements for which it could be computed, in Figure 4 for the two input scenarios N_SO and W_M_SW with their respective timetables T_1, since this might be more interesting for larger networks. Since it is possible for trains to reduce delays by driving 5% faster or by reducing their stopping times there are actually some infrastructure elements where trains are expected to reduce their delay on average. For most infrastructure elements the delays are not or just slightly expected to change on average. However, there are also infrastructure elements that cause an expected additional delay of 5 minutes and more on average. Such infrastructure elements could be avoided, at least during their peak times, when scheduling additional trains or be penalized when computing a more robust timetable. It should be noted that it is not always possible

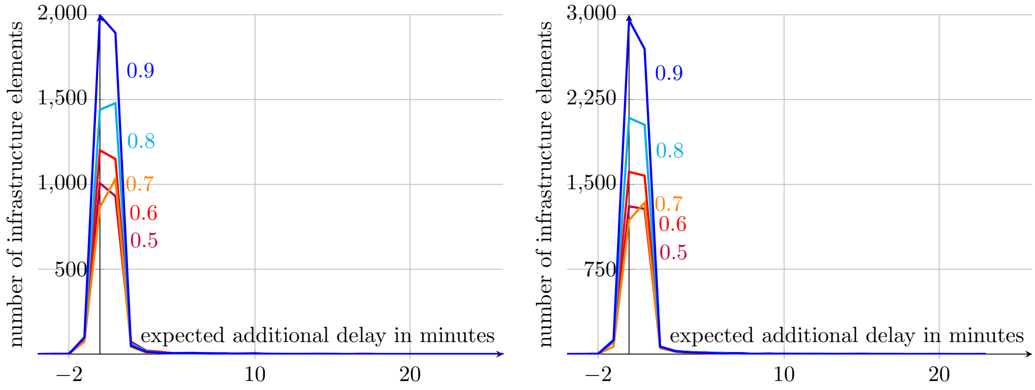

Figure 4 Frequency of delay changes at infrastructure elements (in minutes), on the left for network N_SO, on the right for W_M_SW, in both cases for the respective timetable T_1.

to decrease the utilization of certain infrastructure elements for example for stations with a high demand on passenger transportation.

Additionally, the difference between given infrastructure elements planned utilization and expected utilization could be computed as a function of time, for example to be used as input for an algorithm computing additional train paths. So this approach offers several possible ways to assess a networks' utilization given a fixed timetable.

5 Conclusion

We presented a symbolic approach to simulate railway timetables probabilistically. Our implementation of this simulation is sufficiently fast to simulate real-world sub-networks of the German railway infrastructure. However, this approach still offers possibilities for improvements and extensions. For example the accuracy of the model could be further refined, e.g. by using train type specific blocking times or more realistic initial delay distributions [20, 21]. The simulation itself could be extended to also handle general primary delays, not only those that occur at the start of each train ride. Concerning the implementation multi-threading could be exploited at least for the capacity computations, we would expect this to decrease the running times especially for larger vertices (with a high capacity) and an increasing number of trains that are occupying them. This is reasonable, because the computation of the capacity probabilities for an infrastructure element with capacity cap and m trains that are occupying it requires $\sum_{i=0}^{\min(cap,m-1)} m \cdot \binom{m}{i}$ multiplications.

Last but not least, we are interested in considering the probabilistic utilization when computing additional train paths, in order to avoid disturbing the existing timetable, also under consideration of uncertainties in the delays of other trains.

References

1 *LUKS*, 2020, (accessed July 1, 2020). URL: https://www.via-con.de/en/development/luks/.
2 *OnTime*, 2020, (accessed July 1, 2020). URL: https://www.trafit.ch/en/ontime.
3 *OpenTrack Railway Technology*, 2020, (accessed July 1, 2020). URL: http://www.opentrack.ch/opentrack/opentrack_e/opentrack_e.html.
4 *RailSys*, 2020, (accessed July 1, 2020). URL: https://www.rmcon-int.de/railsys-en/.

5 Emma Uhrdin Andersson. Assessment of robustness in railway traffic timetables, 2014.

6 Thorsten Büker and Bernhard Seybold. Stochastic modelling of delay propagation in large networks. *Journal of Rail Transport Planning and Management*, 2(1):34–50, 2012. `doi:10.1016/j.jrtpm.2012.10.001`.

7 Valentina Cacchiani and Paolo Toth. Nominal and robust train timetabling problems. *European Journal of Operational Research*, 219(3):727–737, 2012.

8 Andrea D'Ariano. Improving real-time train dispatching: models, algorithms and applications, 2008.

9 Andrea D'Ariano and Marco Pranzo. An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances. *Networks and Spatial Economics*, 9(1):63–84, 2009.

10 Burkhard Franke, Bernhard Seybold, Thorsten Büker, Thomas Graffagnino, and Helga Labermeier. Ontime – network-wide analysis of timetable stability. In *5th International Seminar on Railway Operations Modelling and Analysis*, May 2013.

11 Sabine Radke (Deutsches Institut für Wirtschaftsforschung). Verkehr in Zahlen 2019/2020 (in German), 2019.

12 Rob M. P. Goverde and Ingo A. Hansen. Performance indicators for railway timetables. In *2013 IEEE International Conference on Intelligent Rail Transportation Proceedings*, pages 301–306, 2013.

13 David Janecek and Frédéric Weymann. Luks - Analysis of lines and junctions. In *Proceedings of the 12th World Conference on Transport Research (WCTR 2010), Lisbon, Portugal*, 2010.

14 Christian Liebchen, Michael Schachtebeck, Anita Schöbel, Sebastian Stiller, and André Prigge. Computing delay resistant railway timetables. *Computers & Operations Research*, 37(5):857–868, 2010. Disruption Management. `doi:10.1016/j.cor.2009.03.022`.

15 Andrew Nash and Daniel Huerlimann. Railroad simulation using OpenTrack. *Computers in Railways IX*, pages 45–54, 2004. `doi:10.2495/CR040051`.

16 Alfons Radtke. Infrastructure modelling. *Eurailpress, Hamburg*, 2014.

17 Alfons Radtke and Jan-Philipp Bendfeldt. Handling of railway operation problems with RailSys. In *Proceedings of the 5th World Congress on Rail Research (WCRR 2001), Cologne, Germany*, 2001.

18 Richard L. Sauder and William M. Westerman. Computer aided train dispatching: decision support through optimization. *Interfaces*, 13(6):24–37, 1983.

19 Walter Schneider, Nils Nießen, and Andreas Oetting. MOSES/WiZug: Strategic modelling and simulation tool for rail freight transportation. In *Proceedings of the European Transport Conference, Straßbourg*, 2003.

20 Jianxin Yuan. *Stochastic modelling of train delays and delay propagation in stations*, volume 2006. Eburon Uitgeverij BV, 2006.

21 Jianxin Yuan and Giorgio Medeossi. Statistical analysis of train delays and movements. *Eurailpress, Hamburg*, 2014.