Micro–macro transformation of railway networks

Thomas Schlechte\textsuperscript{a}, Ralf Borndörfer\textsuperscript{a}, Berkan Erol\textsuperscript{a}, Thomas Graffagnino\textsuperscript{b}, Elmar Swarat\textsuperscript{a},\textsuperscript{e}

\textsuperscript{a}Zuse Institute Berlin (ZIB), Takustr. 7, 14195 Berlin-Dahlem, Germany
\textsuperscript{b}SBB AG Bern, Mittelstrasse 43, 300 Bern 65, Infrastructure/Timetable and Network Design, Switzerland

\begin{abstract}
This paper presents a bottom-up approach to automatic railway network simplification. Starting from a detailed microscopic level as it is used in railway simulation, the network is transformed by an algorithm to an aggregated level, i.e., to a macroscopic network, that is sufficient for long-term planning and optimization. Running and headway times are rounded to a user defined precision by a special cumulative method. After this “macrotization” trains from a given set of requests are added to the existing timetable by solving an optimal train path allocation problem. The objective of this problem is to maximize a sum of utilities of the allocated trains; the utility can be a constant, some monetary value, etc. The optimized schedule is re-transformed back to the microscopic level in such a way that it can be simulated without any conflicts between the train paths. We apply this algorithm to macrotize a microscopic network model of the highly frequented Simplon corridor in the Alps between Switzerland and Italy. To the best knowledge of the authors and confirmed by several railway practitioners this was the first time that track allocations that have been produced in a fully automatic way on a macroscopic scale fulfill the requirements of the originating microscopic model and withstand an evaluation in the microscopic simulation tool OpenTrack. Our micro–macro transformation method allows for a much faster planning and provides solutions of a quality that are at least comparable to the most sophisticated manual schedules. In this way meaningful scenario analyses can be carried out that pave the way towards a new level of decision support in railway planning.
\end{abstract}

1. Introduction

Timetabling is one of the major planning tasks in railway traffic. It involves two parties. On the one hand the railway operators need to compute a timetable that uses a small number of vehicles and crews and that satisfies passenger demands like short travel and transfer times. On the other hand the infrastructure companies must decide about the allocation of train paths to the train requests of the operators. This is especially challenging when conflicts between different requests occur. In such a situation, in particular, in highly utilized networks, it is critical that infrastructure capacity is not left unused and that good connections are guaranteed at all important points in the network. This is hard to achieve by manual planning. There is therefore a need for methods that allow for the use of optimization algorithms in timetabling to solve models like the periodic event scheduling problem (PESP) (Serafini and Ukovich, 1989) or the train timetabling problem (TTP) (Caprara et al., 2002).

The capacity and the efficiency of railway networks are important research topics in engineering, operations research, and mathematics since several decades. The main challenge is to master the tradeoff between accuracy and complexity in the planning, optimization, and simulation models. Radtke (2008) and Gille et al. (2010) proposed the use of both microscopic and macroscopic models. They applied microscopic models for running time calculations and the accurate simulation of railway operations, and macroscopic models for long term traffic and strategic infrastructure planning. In a similar vein, Schultze et al. (1985) suggested a procedure to insert train paths according to pre-defined priorities in a first step, and to test the reliability of this timetable in a second step by a simulation of stochastic disturbances. An alternative approach is the use of analytical methods, that aim at expressing the railway network efficiency by appropriate statistics, e.g., the occupancy rate. There exist two different research branches. The first is the handicap theory by Potthoff (1980); it is based on queuing models. The second uses probabilistic models to propagate delays; it is mainly based on the work of Schwanhäußer (1974). He also introduced the important concept of section route nodes to analyze the performance of route nodes or stations. Hansen (2010) presents an alternative probabilistic model for a precise estimation of expected buffer and running times. Last not least, there is also a substantial literature on discrete optimization
approaches to timetable optimization. Due to the complexity of railway traffic, most articles consider only coarse macroscopic models with a simplified routing through the railway infrastructure on simple network topologies, such as corridors (e.g., Caprara et al., 2002; Cai and Goh, 1994; Brännlund et al., 1998; Liebchen, 2006; Borndörfer and Schlechte, 2007; Fischer et al., 2008). On the other hand, routing through individual stations has been considered on a much more detailed level (see Zwaneveld et al., 1996; Lusby et al., 2006; Caprara et al., 2007). The interaction of both approaches has only recently been studied. Caimi (2009) uses a top-down approach, while D’Ariano et al. (2008) and Corman et al. (2009) iterate between a routing and a scheduling phase on a network that is aggregated on the level of main signals. The last mentioned authors study the operational variant of the timetabling problem, called rescheduling or dispatching, and use a so-called alternative graph model to reschedule trains through complex stations in order to minimize waiting times. In a similar vein, Corman et al. (2010) proposed a distributed approach, in which independent local routings are adjusted on a superordinate global coordination level.

This paper presents a bottom-up approach of automatic simplification of a complex microscopic railway infrastructure model and applies it in a case study to the Simplon corridor. The term “microscopic” points out that the input data describes the infrastructure on a very detailed level, that makes it possible to simulate the railway traffic with exact track, switch, and platform assignments of the train paths like it would be in the real world. An aggregation technique condenses this microscopic representation to those data that are relevant for planning and optimization purposes. Transforming the data to a less detailed level makes it possible to compute optimal train path allocations and feasible timetables by methods of linear and integer programming. In particular, a timetable is feasible on the microscopic level if the block occupations of the trains do not overlap, and a timetable is feasible on the macroscopic level if the minimum headway times between each two succeeding trains are respected. Of course, the aggregation/de-aggregation has to be done in such a way that (i) enough degrees of freedom remain and (ii) that a feasible train path allocation on the macroscopic level can be transformed back to the microscopic level without creating any conflicts. We describe in this paper a method that does exactly this.

We test our method using real world data for the Simplon corridor from Brig (BR) in Switzerland to Domodossola (DO) in Italy provided by SBB Schweizerische Bundesbahnen. The Simplon is one of the major transit corridors in the European railway network. It has a length of 45 km and features 12 stations. The microscopic model for this scenario consists of 1154 nodes and 1831 arcs including 223 signals, which is fairly large, see Fig. 1 for the detailed track layout. Furthermore the routing possibilities at the terminals Brig (on the left hand of Fig. 1) and Domodossola (on the right hand of Fig. 1) and in the intermediate stations Iselle and Varzo, and a rather unusual slalom routing for certain cargo trains through the tunnel lead to complex planning situations. Before describing our micro–macro transformation in detail, we give a short discussion on the pros and cons of microscopic and macroscopic railway modeling, and why they have to be combined in order to arrive at a method that is both accurate and tractable.

Railway infrastructure and train operations are often modeled using simulation programs. In the last 20 years several software programs for simulating train movements were developed (Wendler, 1999; Hürlimann, 2001; Siefer and Radtke, 2005). Almost all railway companies use them to support their operations and planning processes. Simulation systems provide a realistic assessment of different options in infrastructure planning. They allow to study the interactions of large numbers of trains in a network, and, in particular, to evaluate the feasibility of a timetable, i.e., if a timetable works in simulation, it can be trusted to be operable in practice. We used in our work the synchronous simulation system Open-Track that was developed at the ETH Zurich (Hürlimann, 2001), see also Hansen and Pachl (2008) for an overview and a comparison of synchronous and asynchronous simulation systems.

A simultaneous optimization of a large number of train paths at a microscopic granularity is currently out of reach and would also
not be appropriate in many high-level strategic and tactical planning situations. For these purposes, it is better to resort to a macroscopic model of the railway system. Such a macroscopic model contains much less information such that the network size can be reduced significantly. In addition to that, a fixed time discretization can be used in order to make the model amenable to discrete optimization techniques. In Erol et al. (2008) a standardized format for macroscopic railway models was introduced and a number of test instances that model a part of the German long distance network were made freely available. Simplified macroscopic models of the railway infrastructure and estimates of event times, mostly in minutes, have been used with the success in line optimization (Borndörfer et al., 2007) and periodic timetable optimization (Liebchen, 2006).

Our contribution is to present a bottom-up approach to railway network aggregation that starts at the microscopic level, goes to a macroscopic model, and ends again at the microscopic level. We present in Section 2 an algorithmic approach that implements this idea. This approach is tested in Section 3, where we present computational results for different optimization scenarios for the Simplon corridor.

2. Microscopy and macroscopy, or there and back again

Railways are highly complex technical systems, which can be modeled at any level of detail. This modeling effort is no end in itself. Rather, an accurate calculation of running times and precise and unambiguous platform and track allocations are needed to make simulation results match with the real world. The necessary precision can be achieved using microscopic data such as topological gradients, speed-dependent traction forces, speed limitations, and signal positions. However, this type of information is too complex to be handled in an integer programming optimization model. Our aim is therefore to work with a macroscopic model with the property that the results can be interpreted in and re-transformed to the microscopic world and finally operated in reality. The main contribution of this work is to introduce an algorithm that constructs from a microscopic railway model a macroscopic model with the following properties:

- Macroscopic running times can be realized in microscopic simulation.
- Sticking to macroscopic headway-times leads to conflict-free microscopic block occupations.
- Valid macroscopic timetables can be transformed into valid microscopic timetables.

This section defines the microscopic and macroscopic elements of our approach, and it describes a suitable transformation in detail. It is structured as follows. Section 2.1 discusses microscopic railway network models. Section 2.2 motivates our aggregation idea and introduces some details concerning the construction of macroscopic networks. The following Section 2.3 deals with time discretization. Finally, we propose an algorithm that performs the micro–macro transformation in Section 2.4. We remark that although our exposition is based on the simulation tool OpenTrack, the methodology is generic.

2.1. Microscopic railway networks

The main input for the transformation algorithm is a microscopic infrastructure network that is given as a graph \( G = (V,E) \). In our case this data is given in the proprietary format of OpenTrack, the simulation tool used by SBB. OpenTrack uses a special graph data structure in which nodes correspond to so-called double-vertices, that model signals, switches, etc. These consist of a left and a right part, see Fig. 2 for examples and the documentation in Montigel (1992) and Montigel (1994) for a more detailed description. OpenTrack adopts the convention that if a path in \( G \) enters a node at the left end, it has to leave at the right end and vice versa. This assures consistent directions of train routes and that no illegal turn arounds at switches can be done. Every track section between two vertices is modeled as an edge, and every edge has some attributes like maximum speed or length. A double-vertex is introduced at any point where one or more of these attributes change or if there is a switch, a station, or a signal on a track. Fig. 2 shows an example of a double-vertex graph in OpenTrack.

Our transformation approach is based on the consideration of a set of potential routes in \( G = (V,E) \) for trains of standardized train types. A microscopic route is a path through the microscopic infrastructure that is valid for some train type and that starts and ends at a node inside a station or at a node representing a storage siding. Some nodes on the route can be labeled as stops, namely, when the train can potentially stop there, i.e., at nodes representing station platforms, or at stop opportunities on passing tracks. Note that the train routes induce the directions in which the microscopic infrastructure nodes and edges can be used. This will directly influence the definition, e.g., of the headway parameters of the macroscopic model, as we will explain later in Section 2.2. Let \( C \) denote the set of all train types and \( R \) the set of all given routes in \( G = (V,E) \) (note that several routes can belong to one train type).

Train types should be chosen in a clearly discriminable way and conservatively with respect to their “train class” (heavy cargo

---

**Fig. 2.** The topology of a part of a microscopic railway network plotted by the simulation software OpenTrack. Signals can be seen at some nodes, as well as platforms and station labels.
trains, slow interregional or regional passenger trains) to avoid infeasible running times. Detailed simulation data has to be calculated carefully such that precise running times and blocking times in units of \( \delta \) (some discretization step size, e.g., one second) can be computed, see Fig. 3. Running times and blocking times are basic elements of our approach and will be discussed next.

In Pachl (2002) and Brünger and Dahlhaus (2008) the basic laws of dynamics are applied to derive the dynamics of a train movement. These methods have been implemented in state-of-the-art railway simulation software packages, e.g., OpenTrack, in order to come up with plausible values for exact running times (see Nash and Huerlimann, 2004). Different tools differ in their data structures, interfaces, and in some minor interpretations. However, the main concepts of running and blocking times are the same. We remark that our approach cannot only be used in connection with OpenTrack, but that it can be easily adapted to any simulation tool that provides accurate running and blocking times, such as RailSys (Radtke, 2005) or RUT-K (Brünger and Gröger, 2003).

In Europe, blocking times are used to quantify the infrastructure capacity consumption of train movements. The approach is based on the early work of Happel (1950) and Happel (1959) and the intuitive concept to associate the use of physical infrastructure resources over certain time intervals with trains or train movements, see also Pachl (2008) and Klabes (2010) for a comprehensive description of blocking time theory. We will now give a brief discussion of blocking times that contributes to a better understanding of our transformation algorithm.

The origin of the blocking time stairs, shown in Fig. 3, is the well-known train protection system called train separation in a fixed block distance. In this method, the railway network is divided into block sections, which are bordered by main signals. A block section must not be occupied by more than one train at a time. When a signal allows a train to enter a block section, the section is locked for all other trains. In this way, the entire route between the block starting main signal and the overlap after the subsequent main signal is reserved for the entering train.

Fig. 3 shows that the time interval during which a route \( r \) occupies a track segment consists of the relative reservation duration \( \xi_r \) and the relative release duration \( u'_e \) on edge \( e \in E \). The relative reservation duration is the sum of the approach time, the signal watching time, sometimes called reacting time, and the time needed to set up the route. The relative release duration is the sum of the release time, the clearing time, sometimes called switching time, and time needed by the train between the block signal at the beginning of the route and the overlap. The switching time depends significantly on the installed technology (see Klabes, 2010; Schwanhäußer et al., 1992). In order to prevent trains that want to pass a block section from undesirable stops or braking procedures, the block reservation should be finished before the engine driver can see the corresponding distant signal. Then the section stays locked while the train passes the track between the beginning of the visual distance to the caution signal and the main signal and thereafter the block section until it has cleared the overlap after the next main signal. Then the section is released. This regime can be improved in block sections that contain con- or diverging tracks, because in such cases it is often possible to release parts of the section before the train has passed the overlap after the next main signal. We finally remark that blocking times are also used in moving block systems like the future ETCS Level 3 system. Arbitrarily small blocks, i.e., blocks with lengths converging to zero, are considered in simulations of moving block systems in order to emulate the blocking times, see also Emery (2008) and Wendler (2009) for an investigation of the influence of ETCS Level 3 on the headway times. Simulation tools have to respect all these technical details. From an optimization point of view, however, it is sufficient to consider abstract blocking time stairs, regardless of the safety system they stem from or how they were computed.

We summarize the microscopic information that we use in Table 1.

### Table 1

<table>
<thead>
<tr>
<th>Description</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>A microscopic node</td>
<td>( \bar{v} \in V )</td>
</tr>
<tr>
<td>A microscopic edge</td>
<td>( \bar{e} \in E )</td>
</tr>
<tr>
<td>An undirected infrastructure graph</td>
<td>( G = (V,E) )</td>
</tr>
<tr>
<td>A set of directed train routes</td>
<td>( R )</td>
</tr>
<tr>
<td>A time discretization granularity (in seconds)</td>
<td>( \delta \geq 1 )</td>
</tr>
<tr>
<td>Running times for ( \bar{e} \in E ) and ( r \in R ) (measured in ( \delta ))</td>
<td>( d'_e &gt; 0 )</td>
</tr>
<tr>
<td>Release durations for ( \bar{e} \in E ) and ( r \in R ) (measured in ( \delta ))</td>
<td>( u'_e &gt; 0 )</td>
</tr>
<tr>
<td>Reservation durations for ( \bar{e} \in E ) and ( r \in R ) (measured in ( \delta ))</td>
<td>( \xi'_e &gt; 0 )</td>
</tr>
</tbody>
</table>

#### 2.2. Network aggregation

The desired macroscopic network is a directed graph \( N = (S,F) \) for train types \( C \), that is derived from a microscopic network \( G = (V,E) \) and a set of routes \( R \). The construction involves aggregating (inseparable) block sections (paths in the microscopic network \( G \)) to macroscopic tracks \( F \) and station areas (subgraphs of the microscopic network \( G \)) to macroscopic stations \( S \). The aggregation will be done in a way that depends on the given routes \( R \) and on the defined train types \( C \), such that the complexity of the macroscopic network depends only on the complexity of the interactions between the given train routes, and not on the complexity of the network topology, i.e., all interactions between all potential train routes, which is much more. This is a major advantage over other approaches, because the aggregation is detailed where precision is needed and compressed where it is possible.

We will now describe the idea of the construction by means of an example. First, all potential departure and arrival nodes at some station that are used by the routes \( R \) are mapped to one macroscopic station node. Additional macroscopic nodes will be introduced in order to model interactions between routes due to shared resources. The potential interactions between train routes in a double-vertex graph are:

- Complete coincidence, i.e., routes have identical microscopic paths.
Convergence, i.e., routes merge at a microscopic node (and traverse it in the same direction).

Divergence, i.e., routes separate at a microscopic node (and traverse until then in the same direction).

Crossing, i.e., routes cross at a microscopic node (and traverse it in the opposite direction).

Let us discuss some of these interactions between train routes at the example of the infrastructure network shown in Fig. 4.

Consider first a single standard train that runs from platform BRRB (we denote any place where stopping is allowed as a platform) to platform IS. Then it is enough to consider just one single track from station BRRB to IS in the macroscopic infrastructure. Note that this macroscopic track could correspond to a long path in the microscopic representation. Consider now additional standard trains from BRRB to IS. Possible interactions and conflicts between these train routes are the self correlation on the directed track from BRRB to IS, as well as the platform capacity for standard trains, which allows, say, exactly one train to wait in BRRB or IS. Another standard track running from BR to IS calls for the definition of a pseudo-station BRTU at the track junction in order to model the train route convergence correctly. Our model distinguishes between regular station nodes, where a train can stop, and pseudo-station nodes, which are not stop opportunities, i.e., in our model trains are not allowed to wait at a pseudo-station or to change their direction there. The pseudo-station BRTU splits the track from BRRB and IS into two tracks: from BRRB to BRTU and from BRTU to IS. The track between BRTU and IS is used both by trains from BRRB to IS and from BR to IS. Hence headway conditions between them must be satisfied in a timetable. But between BR/BRRB and BRTU this holds not, because there each train has its own track. So inserting the pseudo-station helps us to restrict the resource conflict and the headway condition to only the track section that is used by both trains.

If it is possible to run trains on the same microscopic segment in the opposite direction from IS to BRRB, another directed track has to be defined in the macroscopic network. Besides the standard self correlation, the conflict for opposing routes also has to be modeled, see Fig. 3. Diverging or crossing situations between opposing train routes can be handled in an analogous way. Along the lines of these examples, we can exploit aggregation potentials in the infrastructure network by representing several microscopic edges on a route by only one macroscopic track. Of course, macroscopic track attributes can also be compressed. For example, if we assume that the route from BRRB to IS and the route from BR to IS are operated by the same train type, we can use a single value for the running time on the track from BRTU to IS.

After constructing the regular stations, the pseudo-stations, and the tracks between them, the network can be further reduced by a second aggregation step. Again consider the situation in Fig. 4. Suppose platforms BRRB and BR belong to the same station B. If BRTU is a close junction associated with B, then it may be viable to contract nodes BRRB and BR to one major station node B with a directed platform capacity of two as shown in Fig. 5. Of course, by doing so we loose the accuracy of potentially different running times between different platforms of B and the other stations, and we also loose control over the routing through or inside B, which both can produce small infeasibilities on the operational level. However, one can often achieve significant reductions in network sizes in this way, without losing too much accuracy.

2.3. Time discretization

A popular approach to timetabling and train path allocation is based on the use of space–time graphs, i.e., the time is discretized. Similar to the topological aggregation, there is also a tradeoff between model size and accuracy in the temporal dimension. This tradeoff is controlled by the discretization step size. The discretized times in the macroscopic model will be based on microscopic simulation data, which is very precise. In fact, simulation tools provide running and blocking times with an accuracy of seconds (or even smaller). Our aim is to aggregate these values in the macroscopic model. We propose for this purpose a conservative approach, which means that running and arrival times will never be underestimated in the macroscopic model.

2.3.1. Running times

Let \( \Delta \in \mathbb{N} \) be a fixed time discretization, i.e., a unit of time, in which all macroscopic times will be measured; e.g., using a unit of six seconds is denoted as \( \Delta = 6 \). Then a first idea is to simply round up all running times to the next unit of \( \Delta \); let us call this procedure ceiling rounding. Fig. 6 shows the difference between microscopic and ceiling rounded running times for a microscopic running time of \( d_j^r = 74 \) at some track \( j \) in some route \( r \) with respect to different time discretizations \( \Delta \). Fine discretizations like less than 15 s produce small deviations, while larger time discretizations can increase the error significantly. However, in the “lucky” case that \( \Delta \) is a divisor of \( d_j^r \), there is no rounding in the discretization step of Algorithm 1 (first line) such that the error is zero, e.g., for \( \Delta = 37 \) in Fig. 6.

The main problem with ceiling rounding is that the error accumulates along a route as it is shown in Lemma 2.1. In the following, we denote by \( \hat{d}_j^r \) the microscopic running time of route \( r \) on track \( j \), by \( d_j^r \) the discretized running time, and by \( e_j^r \) the cumulative rounding error (in units of the microscopic time unit granularity \( \delta \)). The total rounding error at the end of each route is denoted by \( e^r \) (in units of \( \delta \)).

Lemma 2.1. Let \( r \in R \) be a train route in the macroscopic network \( N = (S,J) \) with length \( n_r \) – which is the number of macroscopic tracks of route \( r \) – and running times \( d_j^r \) measured in units of \( \delta \) for each track \( j \in r \). Let further \( \Delta \gg \delta \) be a fixed time discretization. Simply rounding up the running times \( \hat{d}_j^r \) for each track to a multiple of \( \Delta \) produces a worst-case rounding error of \( (\Delta – 1)n_r \).

Proof. For each track we have a maximum possible rounding error of \( \Delta – 1 \). In the worst-case this could occur on all \( n_r \) tracks of \( r \).

Such big rounding errors lead to undesirable extensions of travel times and an inefficient use of the infrastructure capacity. We therefore propose an alternative approach in terms of a more sophisticated cumulative rounding technique. This procedure aims to control the rounding error by only tolerating small deviations between rounded and microscopic running times. The idea is simple: considering running times for each route on each track
with respect to the cumulative rounding error, it is sometimes allowed to round down, because enough implicit buffer time was collected on the way. We must, however, make sure that running times are never rounded to zero, because in our model zero running times are not counted as infrastructure usage, and this can lead to infeasible timetables. A formal description of the procedure applied to a single track of the macroscopic infrastructure network is given in Algorithm 1. In case of the first track of a route \( r \) we set the artificial single track of the macroscopic infrastructure network is given in Lemma 2.2. There are two cases:

1. Let \( \tilde{d}_j^n = (k-1)\Delta \leq \epsilon_{j_{n}}^r \). Then we round down and set \( \epsilon_{j_{n}}^r = \epsilon_{j_{n-1}}^r - (\tilde{d}_j^n - (k-1)\Delta) \).

Because of \( \Delta \leq \tilde{d}_j^n \), a rounding down to zero could not appear. By definition of \( k \) it clearly follows that \( \epsilon_{j_{n}}^r < \epsilon_{j_{n-1}}^r \).

And due to the “If” condition in the algorithm it is obvious that \( \epsilon_{j_{n}}^r = \epsilon_{j_{n-1}}^r - (\tilde{d}_j^n - (k-1)\Delta) \geq 0 \).

2. Consider the other case, that is \( \epsilon_{j_{n}}^r < \tilde{d}_j^n - (k-1)\Delta \). Then \( \epsilon_{j_{n}}^r \) is set to \( \epsilon_{j_{n}}^r + (k\Delta - \tilde{d}_j^n) \). By \( \tilde{d}_j^n \leq k\Delta \) it is evident that \( 0 \leq \epsilon_{j_{n}}^r \).

At last we have to consider the upper bound. It follows that \( \epsilon_{j_{n}}^r = \epsilon_{j_{n-1}}^r + (k\Delta - \tilde{d}_j^n) < \tilde{d}_j^n - (k-1)\Delta + k\Delta - \tilde{d}_j^n = \Delta \).

With the above described rounding technique there is still one problem left. Lemma 2.2 does not apply for the case when there exists a track \( j \) where \( \tilde{d}_j^n < \Delta \). Then it is not allowed to round down. This could imply a worse upper bound for our rounding procedure as shown in Lemma 2.3.

**Lemma 2.3.** We consider the same rounding procedure and the same assumptions as in Lemma 2.2 except for the case that there is a set \( B \subseteq \{1, \ldots, n_t\} \) where \( \tilde{d}_j^n \leq \Delta \) holds for each \( b \in B \). Then the upper bound for the cumulative rounding error \( \epsilon_{j_{n}}^r \) is equal to \((|B| + 1)\Delta\).

**Proof.** We again use an induction technique. At the beginning we look at the first track, where \( \tilde{d}_j^n \leq \Delta \). In this case we have \( (k-1)\Delta = 0 \) and therefore \( k = 1 \). Due to the prohibition that a macroscopic running time equals zero, we set \( \epsilon_{j_{r}}^r = \epsilon_{j_{r-1}}^r + (k\Delta - \tilde{d}_j^n) \). It follows that \( \epsilon_{i_{j}}^r = \epsilon_{i_{j-1}}^r + (k\Delta - \tilde{d}_j^n) < \Delta \).

Note that, as shown in Lemma 2.2, the rounding error does not grow, if the running time on the current track is greater than \( \Delta \).

Next we consider the case, that we have yet a number of \( i \) tracks with a running time less than \( \Delta \) and the \( i + 1 \) track is occurred. To simplify the notation, the precedent track is denoted by \( i \). Then it follows that \( \epsilon_{i_{j}}^r = \epsilon_{i_{j-1}}^r + (k\Delta - \tilde{d}_j^n) < i\Delta + \Delta - \tilde{d}_j^n = (i+1)\Delta \).

Fig. 7 compares the two rounding methods by illustrating the minimum, average, and maximum rounding errors of the macroscopic running times at the end of example routes for all considered train types through the Simplon corridor with respect to time discretizations varying from 0 to 60 s. The routes have a length of at most ten macroscopic tracks. It is apparent that cumulative rounding dampens the propagation of discretization errors substantially already for short routes. Let us assume a discretization of \( \Delta = 60 \). In that case the maximum rounding error can be reduced from more than 400 s to approximately 100 s by our rounding approach.
### Algorithm 2

**Algorithm 2:** Rounding method for computing discretised minimum headway times.

**Data:** Task $j = (s_1, s_2) = (s_1, \ldots, s_m) \in J$ with $s_1, s_2 \in S$, release duration $d_{rj}^s$, and relative reservation duration $d_{rj}^e$ with $r_j, r_j \in R$, $c_j, c_j \in C$, $c_j \in C$, $i = 1, \ldots, m$, time discretisation $\Delta > 0$.

**Result:** Minimum headway time $h(j, j, c_1, c_2)$ for train type sequence $c_1, c_2$ on track $j$

**begin**

$h = 0$

for $x = (s_1, s_1) \in r_1 \cap r_2$ do

$h = \max(d_{rj}^s + d_{rj}^e, h)$ ; // update timing separation

**return** $\lfloor \frac{h}{\Delta} \rfloor$.

### 2.3.2. Headway times

Based on the occupation and release times in Fig. 3, it is possible to define a minimum time difference after which a train can succeed another train on the same track or after which a train can pass another train from the opposite direction. We restrict ourselves w.l.o.g. to the consideration of minimum headway times for the combination of departure events, i.e., headway restrictions for arrivals can be transformed in a straight-forward manner to departure ones. **Algorithm 2** describes the calculation of the *minimum headway time* for the case of two routes $r_1$ and $r_2$ that traverse a track in the same direction. (We assume that both the trains have the same departure time at $s_1$ when calculating the blocking times.) Here, we denote the corresponding train types by $c_1, c_2 \in C$.

In case of crossing or opposite routes $r_1$ and $r_2$ on a bi-directional track $j = (s_1, s_2)$, which we call single-way track, the headway time is calculated differently. By definition each single-way track $j$ has exactly one counterpart $j = (s_2, s_1) \in J$, which is directed in the opposite direction, and block feasibility with respect to this opposite direction must be ensured by means of a second headway matrix. The entries of this matrix are calculated as follows. Let $j = (e_1, \ldots, e_m)$ be traversed by the directed route $r_j$. Then the minimum headway time for a departure of a train of type $c_2$ on an opposite route $r_2$ at station $s_2$ after a departure of a train of type $c_1$ on route $r_1$ from station $s_1$ is:

$$h(j, j, c_1, c_2) = \sum_{i=1}^{m-1} d_{rj}^s + d_{rj}^e + d_{rj}^e = d_{rj}^e.$$  

This time can be discretized by rounding. In practice additional standard buffer times are added to all headway times in order to increase the robustness of the timetable.

### 2.4. An Algorithm for micro–macro transformation

**Algorithm 3** puts the pieces together in order to transform a railway infrastructure network from a microscopic level to a macroscopic level. The method has been implemented in a software tool NETCAST (Erol, 2009). The procedure consists of three main steps, namely, **macroscopic network detection (ND), aggregation (AG),** and **time discretization (TD),** which will be discussed in this subsection.

**Macroscopic network detection (ND)** means to construct the directed graph $N = (S, J)$ from the microscopic network $G = (V, E)$ and a set of train routes $R$. Denote by $B(r)$ the set of stations visited by route $r \in R$, i.e., the set of microscopic nodes where the train could stop and/or is allowed to wait. All visited stations become macroscopic station nodes. If an interaction, i.e., a convergence, divergence, or crossing, between two routes is detected, one or two pseudo stations are created, respectively. This detection is done by a simple pairwise comparison of train routes. An important aspect of network detection is that the mapping from a microscopic node to its macroscopic representative is unique, i.e., a microscopic node belongs to at most one junction or station in the microscopic model and hence to at most one (pseudo) station in the macroscopic model.

The resulting set of stations $S_{\text{tmp}}$ is further compressed in the aggregation (AG) step by the routine $\text{aggregateStations}()$, that enforces the imaginable aggregations as informally described in Section 2.2.2. At this point, the macroscopic network detection is finished with respect to the set of stations. It remains to divide the routes $R$ into tracks with respect to the macroscopic stations $S$. Here, $\text{nextStation}(r, s)$ denotes the subsequent station of station $s$ on train route $r$. It is important to note that there can be more than one track between two stations, especially after aggregation steps have been carried out. A typical example is two tracks between the same aggregated macroscopic stations, that correspond to physically different microscopic track sections. Fig. 2 illustrates this situation: if each of the stations $S_4$ and $S_5$ is modeled by exactly one node in the macroscopic model, the two microscopic tracks between them correspond to different microscopic track sections.

**Algorithm 3:** An algorithm for micro–macro transformation of railway networks.

**Data:** microscopic infrastructure graph $G = (V, E)$, set of routes $R$, stations $B(r)$, train types $c(r) \in C$, $r \in R$, time discretisation $\Delta > 0$.

**Result:** macroscopic network $N = (S, J)$, with stations $S$ and tracks $J$

**begin**

$N_{\text{tmp}} := \emptyset$; foreach $r \in R$ do

create $s$ ; // create standard station

$N_{\text{tmp}} = N_{\text{tmp}} \cup \{s\}$

foreach $(r_1, r_2) \in (R \times R)$ do

while divergence or convergence between $r_1$ and $r_2$ is found do

create $p$ ; // create pseudo station

$N_{\text{tmp}} = N_{\text{tmp}} \cup \{p\}$

while crossing between $r_1$ and $r_2$ is found do

create $q$ ; // create pseudo stations

$N_{\text{tmp}} = N_{\text{tmp}} \cup \{q\}$

$S := \text{aggregatesStations}(N_{\text{tmp}})$;

$J := \{(s_1, s_2) \in S \times S \mid r \in R \text{ with } s_2 = \text{nextStation}(r, s_1)\};$

**TD**

foreach $j \in J$ do

foreach $r \in R$ do

$c_{rj}^t := \text{calculateRunningTime}(j, r, c, \Delta)$;

foreach $(r_1, r_2) \in (R \times R)$ do

if $j$ is single-way then

$\text{calculateHeadway}(j, j, r_1, r_2, \Delta)$;

if $j$ is single-way then

$\text{calculateHeadway}(j, j, c(r_1), c(r_2))$;

return $N = (S, J)$;

The *time discretization* (TD), the calculation of the rounded running and headway times, is the last step of the algorithm. We denote the running time of train type $c$ on track $j$ on route $r$ by $d_{rj}^{t,c}$, the headway time for the self correlation case, i.e., when a train on route $r_2$ follows a train with route $r_1$, by $h(j, j, r_1, r_2)$, and the headway time for the single-way case by $h(j, j, r_1, r_2)$. The running times are calculated by the cumulative rounding procedure $\text{calculateRunningTime}()$ according to **Algorithm 1**. The function $\text{calculateHeadway}()$ provides the headway times according to **Algorithm 2** and formula (1). The running times for each route, and the headway times for each pair of routes are calculated (and conservatively) aggregated according to the assignment of routes to
train types \( c \in C \). If there are several routes for the same train type, the maximum running and/or headway time is taken. In case of the Simplon corridor these differences are rather small for each single macroscopic track, i.e., below 6 s for our largest macroscopic model. We remark that we have omitted a discussion of so-called running modes of trains (stopping in or passing through a station) in this exposition, but running and headway times with respect to running modes are implemented in the micro–macro transformation tool \textsc{netcast}.

Fig. 8 shows a macroscopic network model for the Simplon corridor that has been generated using Algorithm 3. We summarize the resulting macroscopic data in Table 2. The micro–macro transformation produces a bijective mapping of sub-paths of routes to tracks and a mapping of microscopic nodes to stations that is used to re-transform paths of the macroscopic model back to the microscopic one.

In addition, this automatic transformation provides by construction a re-transformation of macroscopic train paths to microscopic ones. Note that only in the case of aggregation of stations in step (AG), we can choose from different microscopic routes to re-translate macroscopic train paths.

3. Case study Simplon

We tested our micro–macro transformation approach on real world data for the Simplon corridor as already mentioned in the introduction. The first step was to choose six standard train types, namely, two types of passenger trains, regional (R) and intercity trains (EC), one motor-rail type \textit{GV Auto}, and three types of freight trains, viz., standard freight trains \textit{GV MTO}, container trains \textit{GV SIM}, and “rolling highway” trains \textit{GV RoLa}. Trains of the latter two types must not use tracks on one side of the Iselle–Preglia tunnel because of their width. This necessitates a so-called slalom route when such trains depart from Brig. The passenger train paths were given as fixed, i.e., our case study dealt with the saturation of the corridor by freight trains subject to a given passenger timetable. For these six aggregated train types we considered up to 28 different routes through the microscopic network \( G = (V, E) \). These differ in their stopping patterns and in their routing through the important station Varzo, where over-width trains can pass each other.

In addition to the 12 existing stations, some pseudo-nodes were defined in order to model all train interactions correctly. Detecting convergences, divergences, and crossings as described in Section 2 produces a network \( N = (S, J) \) with 55 station nodes and 87 tracks. In order to evaluate the accuracy of our approach and the aggregation potential we produced several networks. Conducting some

Fig. 7. Comparing errors from ceiling rounding (left) and cumulative rounding (right) for different time discretizations varying between 1 and 60 s.

Fig. 8. Constructed aggregated macroscopic network by \textsc{netcast}.

| Table 2 |

<table>
<thead>
<tr>
<th>Description</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>A set of train types</td>
<td>( C )</td>
</tr>
<tr>
<td>A directed network</td>
<td>( N = (S, J) )</td>
</tr>
<tr>
<td>A set of (abstract) stations</td>
<td>( S )</td>
</tr>
<tr>
<td>A set of (abstract) tracks</td>
<td>( J )</td>
</tr>
<tr>
<td>Running times on track ( j \in J ) for ( c \in C ) (measured in ( \Delta ))</td>
<td>( d^c_j )</td>
</tr>
<tr>
<td>Headway times on track ( j \in J ) for ( c_1, c_2 \in C ) (measured in ( \Delta ))</td>
<td>( h(j, j, c_1, c_2) )</td>
</tr>
<tr>
<td>(opposite direction) Headway times on single-way tracks ( j, j' ) for ( c_1, c_2 \in C ) (measured in ( \Delta ))</td>
<td>( h(j, j', c_1, c_2) )</td>
</tr>
</tbody>
</table>
further aggregations, especially in station areas, we constructed a network `simplon_big` with 18 stations and 40 tracks. A second network `simplon_small` with 12 stations and 28 tracks was built with an even coarser station model.

The macroscopic model was verified using a dense manual reference timetable created by the authors. This timetable runs 14 passenger and 21 freight trains in the time window from 8 to 12 am through the Simplon. We abused our train path optimization

Fig. 9. OpenTrack traffic diagram of an optimized timetable, re-transformed to the (microscopic) simulation level using our micro–macro transformation. All trains entering and leaving the corridor are shown from left (Brig) to right (Domodossola) during 10:00 (top) and 12:00 (bottom). The dotted lines represent macroscopic train movements; they are linear. The “real” (simulated) timetable is plotted using solid lines; here, acceleration and braking phases are clearly perceivable.
module TS-opt (Borndörfer et al., 2009) to reproduce this timetable in our macroscopic model by requesting exactly these 35 trains. And indeed, if a fine discretization of, e.g., \( \Delta = 6 \text{ s} \), is used, it is possible to reproduce the timetable accurately. Fig. 9 compares the reproduced macroscopic timetable and its re-transformed microscopic counterpart as simulated using OpenTrack.

With an accurate macroscopic model, we set out for optimization runs. The goal was to saturate the residual capacity of the corridor (remember the passenger trains are given as fixed) by scheduling additional freight trains (GV MTO, GV SIM, GV RoLa). That is a challenging task because of the large deviations for the running times of passenger and freight trains, i.e., an intercity train needs approximately 30 min and a GV RoLa 50 min, respectively.

To this purpose, we defined some artificial demand by creating two sets of train requests covering a 24 h time horizon. Both of these sets feature a lot of competing train paths. The first set, request1, contains 390 train requests including 63 fixed passenger trains; this set contains a manually constructed test timetable. The second set, request2, contains 255 train requests (including the passenger train requests); in this set, the freight train requests are uniformly distributed and mixed over the time horizon without any advance knowledge of the corridor.

The linear objective function was a dominant profit value for each train request, i.e., also for the fixed ones, minus a small penalty for deviations from optimal arrival and departure times. In case of the fixed passenger trains there was no flexibility with respect to the routing. Arrival and departure times were fixed to the minute, i.e., for a time discretization of \( \Delta = 6 \text{ s} \) there are 10 possible arrival and departure times.

The aim was to produce various dense timetables to evaluate the feasibility of the results of our transformation approach. We remark that our study ignores certain capacity restrictions in the station areas at Brig and Domodossola, i.e., the incoming train flows that can be handled there. All computations were done on machines with a 3 GHz Intel Quad Core Processor and 8 GB RAM on Suse-Linux 11.2. CPLEX 12.1, was used as a LP and MIP solver (IBM ILOG CPLEX, 2009). TS-opt was able to solve all 6 instances shown in Table 3 to proven optimality. In addition, we retransformed the macroscopic timetables to microscopic ones that were simulated in OpenTrack without any block occupation conflicts. This demonstrates that the micro–macro transformation works for real-world instances and that an accurate and conservative aggregation can maintain timetable feasibility.

Table 3 shows the influence of different time discretizations on solution time and solution quality for request sets request1 and request2 using the simpson_small network. As expected, a coarser time discretization reduces solution times, but decreases solution quality (in terms of numbers of trains). In case of request1 the difference is 30 trains. Almost the same can be observed for request2 where we could produce an optimal solution with 143 trains for the discretization \( d_f = 30 \text{ s} \) and 176 trains for \( d_f = 6 \text{ s} \). The main results of this Simplon case study, i.e., all details and a comparison of the resulting timetables, can be found in Borndörfer et al. (2010).

It was, however, a surprise for us that the effect is already so large in this range. This hints at a potential for more fine-grained and more “local” time discretization methods. In addition, the results document that the network aggregation is by no means trivial and that the choice of the discretization is crucial. The results also demonstrate the potential of aggregation methods, i.e., the presented micro–macro transformation, to allow for exact optimization approaches to railway track allocation.

## 4. Conclusion

In this paper we proposed an algorithmic bottom-up approach to transform a microscopic railway network to an aggregated macroscopic network model and back. The transformation is done in such a way that the macroscopic model contains all the information that is necessary in order to compute a conflict-free track allocation. Our micro–macro transformation algorithm constructs the macroscopic network structure by analyzing interactions between standard train routes. In this way, the algorithm can ignore or compress parts of the network that are not used, and still account for all route conflicts by constructing suitable pseudo stations. Time is discretized by a cumulative rounding procedure that minimizes the differences between aggregated and real running times. We tested our approach at the example of the challenging Simplon railway corridor. Our micro–macro transformation approach produced macroscopic models of the Simplon corridor that were small enough to allow for a simultaneous optimization of more than 300 train paths. In this way, it was possible for the first time to compute an efficient and operable (i.e., operable in our simulation setting) 24 h timetable for the Simplon corridor by an optimization algorithm. Further potential applications at a comparable level of complexity includes capacity analyses for large nodes (cf. the current “Stuttgart 21” discussion about the construction of a new railway station in the city of Stuttgart) or network links (such as the planned high-speed link between the cities of Erfurt and Nuremberg), or the optimization of mine railways, which typically include some single-track segments in tunnels that become bottlenecks in times of high demand). Another important issue is the use of our method in larger networks than a corridor like the Simplon. We feel confident that our method is also applicable to more complex settings, but so far no exact microscopic infrastructure data for other networks is available to us.

### Acknowledgments

We thank Martin Bailer for the contributions to time discretization and cumulative rounding, and Daniel Hürlimann for his excellent support for the simulation tool OpenTrack. Finally, we thank Martin Grötschel, Gottfried Illgmann, and Klemens Polatschek for their important support in organizing and realizing this project, and in particular the Simplon case-study. Furthermore, we want to thank three anonymous referees for improving this paper by their valuable comments. This work was funded by the Federal Ministry of Economics and Technology (BMWi), project Trassenbörse, Grant 19M7015B.
References


