Methods and algorithms for the development of robust and resilient timetables

Grant Agreement N°: FP7 - SCP0 – GA – 2011 - 265647

Project Acronym: **ON-TIME**

Project Title: **O**ptimal **N**etworks for **T**rain Integration **M**anagement across **E**urope

Funding scheme: Collaborative Project

Project start: 1 November 2011

Project duration: 3 Years

Work package no.: WP03

Deliverable no.: D3.1

Status/date of document: Final, 15/05/2014

Due date of document: 30/04/2014

Actual submission date: [15/05/2014]

Lead contractor for this document: Delft University of Technology
Delft, The Netherlands

Project website: www.ontime-project.eu

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision control / involved partners

Following table gives an overview on elaboration and processed changes of the document:

Revision	Date	Name / Company short name	Changes
1	13/12/2013	TUDelft	Framework
2	15/02/2014	UdB	First draft Ch. 4
3	17/02/2014	TUD	First draft Ch. 5
4	27/02/2014	TUDelft	First draft Ch. 2 & 3
5	24/03/2014	TUDelft	First complete draft
6	23/04/2014	UdB	Update Ch. 4
7	29/04/2014	TUD	Update Ch. 5
8	14/05/2014	TUDelft	Update Ch. 1 & 6, editorial revision

Following project partners have been involved in the elaboration of this document:

Partner No.	Company short name	Involved experts
14	TUD	Anne Binder
15	TUDelft	Rob Goverde, Nikola Besinovic
16	UdB	Roberto Roberti

Executive Summary

This document presents the methods and algorithms for the development of robust and resilient timetables as developed in WP3.

Chapter 1 presents the general ON-TIME WP3 timetabling framework consisting of three integrated modules: a microscopic model, a macroscopic model, and a fine-tuning model, as well as the I/O data transformations from standardized RailML input files. Moreover, the performance measures and objectives are defined that are taken into account in the timetabling approach explicitly: infrastructure occupation, stability, feasibility, robustness, resilience, transport volume, journey time, connectivity and energy consumption.

Chapter 2 presents the ON-TIME WP3 algorithmic framework and defines the three hierarchical levels of network modelling: the microscopic (detection-section level), mesoscopic (block level) and macroscopic (open-track level) networks, which are used by different modules and can be converted into each other enabling a consistent data structure at different levels of detail.

Chapter 3 presents the microscopic model which consists of a suit of microscopic and mesoscopic (block level) algorithms that compute basic timetable elements such as running times, blocking times, and minimum headway times, as well as infrastructure occupation, stability, conflict detection, and bandwidths of train path envelopes. The results of these computations are used in the macroscopic and fine-tuning models. The microscopic model also takes care of the aggregation of processes for the macroscopic data and acts as the kernel for the other two models in an iterative process. The microscopic model feeds aggregated data to the macroscopic model that computes a feasible timetable at macroscopic level which is then transformed back into a microscopic timetable which is checked on conflicts, capacity consumption, and stability. If one of these performance measures is not yet satisfied, new aggregated data is computed and the macroscopic model is called again. Otherwise, energy-efficient speed profiles are computed and bandwidths are determined for optimization by the fine-tuning model.

Chapter 4 considers the macroscopic model which consists of an optimization model and a stochastic delay propagation model with re-optimization for robustness evaluation. The optimization model finds a macroscopic timetable that minimizes a cost function consisting of running, dwell, and transfer times, and costs for cancelling trains, cancelled connections, and periodicity (where relevant). A heuristic algorithm generates multiple timetables that are analysed on robustness using a stochastic model which selects the most robust timetable as final output.

Chapter 5 considers the fine-tuning model which consists of a dynamic programming model that finds the optimal allocation of running time supplements and dwell times within a corridor for local trains taking into account stochastic dwell times at the in-

termediate stops and energy-efficient speed profiles between the stops. The objective is minimizing the expected energy consumption and expected delays at the intermediate and target stations.

Finally, Chapter 6 gives conclusions and classifies the developed timetabling approach at Timetabling Design Level 3 and 4. The approach incorporates stability, feasibility and robustness explicitly in the computation of a high-quality timetable using an integrated approach of deterministic, stochastic, macroscopic and microscopic models. It is based on standardized RailML files, where the RailML Timetable scheme has been extended with microscopic time-distance and energy-efficient speed profile information for punctual running. Timetabling Design Level 4 is obtained regarding the timetable resilience to ad-hoc freight path requests.

Table of contents

Table of figures

1 INTRODUCTION

1.1 The ON-TIME timetabling approach

This document describes an approach to achieve high-quality railway timetables with an integrated set of state-of-the-art timetabling techniques. It is based on the timetabling State-of-the-Art [6] and Functional Design [7] documents. The State-of-the-Art review recommended as an overall functional objective to develop 'a scheduled train-path assignment application, with automatic conflict detection capabilities, that builds on the concept of robust and resilient timetables, has a unified network coverage, is microscopic at selected parts of the control area, is scalable, and pluggable to Traffic Management Systems, with user-friendly interfaces and execution states that correspond to the IM timetabling management milestones.' The Functional Design document [7] defined several performance measures and gave a classification of Timetabling Design Levels based on how well these performance measures are incorporated in the timetabling design process. The Functional Design document also laid down the framework of timetabling modules to achieve the highest possible Timetabling Design Level which is worked out in this document.

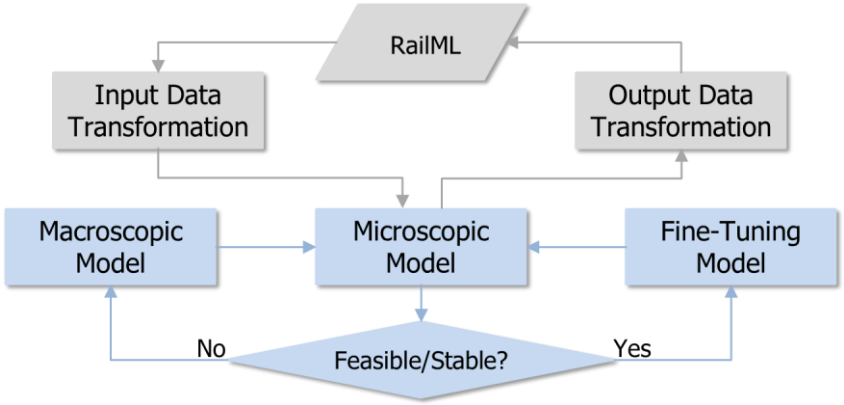

Figure 1. ON-TIME timetabling framework

Figure 1 shows the framework of the developed ON-TIME timetabling approach. The input and output are standardized RailML files. In particular, the output is a standardized RailML Timetable file, which can be used by external models and (simulation) tools such as the ones developed within the ON-TIME project. The RailML input data is transformed into an efficient data format that is used internally by the timetabling modules. The timetabling computation is an iterative process on three levels:

- A microscopic model computes reliable process times at a highly-detailed local level and checks for microscopic feasibility and stability,
- A macroscopic model optimizes a timetable at aggregated network level, and
- A fine-tuning model that computes energy-efficient speed profiles and optimizes the schedules of local trains at corridor level taking into account stochastic dwell times.

The microscopic module is the kernel of the timetabling process and also contains several computation and data conversion tools to provide the macroscopic and fine-tuning modules with the required data.

The approach works roughly as follows. In the first iteration, the microscopic model computes all running and blocking times based on train path requests and infrastructure, interlocking, and rolling stock data. The microscopic network is then aggregated to a macroscopic network which is formulated subsequently as an optimization model and solved in the macroscopic module with respect to global network objectives. At this stage multiple feasible solutions are generated which are analysed on robustness to random delays of all trains using a delay propagation algorithm with re-optimization. The most robust macroscopic timetable is transformed back onto the microscopic network with recomputed process times and checked on conflicts and capacity consumption. If conflicts are detected at the microscopic level or the infrastructure occupation is too high, the microscopic model returns updated aggregated process times and possibly some relaxed constraints (like maximum running times) to the macroscopic model. This iterative process continues until a macroscopically robust timetable has been computed that is also microscopically feasible and stable. At this point, the train schedules on corridors between the main (macroscopic) stations are optimized. First, energy-efficient speed profiles are computed for all trains and the bandwidths for optimizing the schedules of local trains on each corridor are determined. The fine-tuning model then optimizes the arrival and departure times of the local trains at the intermediate stops within the corridor taking into account the stochastic dwell times of these trains and minimizing energy consumption. The final result is a high-quality timetable that is optimized on network and corridor level, and microscopically conflict-free and stable. It is returned as a RailML Timetable file. Chapter 2 provides the full details of the framework.

1.2 Performance measures

The Functional Design document [7] defined several performance measures that were leading in the developed timetabling approach. These performance measures are defined as follows:

- **Infrastructure occupation**: The share of time required to operate trains on a given railway infrastructure according to a given timetable pattern.
- **Timetable feasibility**: The ability of all trains to adhere to their scheduled train paths. A timetable is feasible if (i) the individual processes are realizable within their scheduled process times, and (ii) the scheduled train paths are conflict free, i.e., all trains can proceed undisturbed by other traffic.
- **Timetable stability**: The ability of a timetable to absorb initial and primary delays so that delayed trains return to their scheduled train paths without re-scheduling.
- **Timetable robustness**: The ability of a timetable to withstand design errors, parameter variations, and changing operational conditions.

- **Timetable resilience**: The flexibility of a timetable to prevent or reduce secondary delays using rescheduling (re-timing, re-ordering, re-routing).

The timetabling approach tries to schedule all train path requests with sufficient time allowances for a stable and robust conflict-free timetable and satisfying the UIC infrastructure occupation norms. This might require extending critical running times on corridors with an unacceptable capacity consumption to decrease running time differences. Moreover, the timetable is computed at a precision of 10 s instead of a minute, which avoids capacity waste and unrealizable process times due to rounding to whole minutes.

Next to these indicators the timetable approach considers the objectives that were derived in D1.2 [5] to evaluate the ON-TIME solutions. For timetabling these objectives are defined as follows:

- **Transport volume**: Maximise the delivered passenger kilometres and cargo tonne kilometres over a time window.
- **Journey time**: Minimize the journey times relative to the minimum sectional running time.
- **Connectivity**: Minimise the transfer times between selected services relative to minimum necessary transfer times.
- **Resilience**: Minimise the delay propagation in the system.
- **Energy**: Minimise the sum of energy consumed by trains.
- **Resource usage**: Minimise the track utilisation percentage subject to the minimum traffic demand being met.

In the context of timetabling these objectives were understood as follows. Maximizing transport volume is realized by avoiding cancelled train path requests. This is a trade-off between the number of train paths scheduled, journey time, and infrastructure occupation. The journey times are minimized with respect to the nominal running times consisting of microscopically computed running time and a given minimum running time supplement required for reliable running times. Resilience is subdivided into timetable stability, timetable robustness and timetable resilience as defined above. Track utilization percentage is synonymous to infrastructure occupation.

1.3 Outline

The remainder of the document is outlined as follows. Chapter 2 explains the framework of the timetabling module in detail and also presents the network models at three levels of detail: microscopic, mesoscopic and macroscopic networks, along with the micro-macro network transformation that is essential in the three-level timetabling approach. Chapter 3 then considers the suit of microscopic models and algorithms as well as the conversion algorithms between the microscopic and macroscopic data. The macroscopic model and algorithm is presented in Chapter 4, and the corridor fine-tuning model and algorithm in Chapter 5. Finally, Chapter 6 gives some final conclusions.

2 THE TIMETABLING MODULE

2.1 Algorithmic framework

The structure of the proposed hierarchical framework for timetable design is shown in Figure 2, which indicates the interactions among the different models, their functions as well as the input-output data flow. The framework has a standardized RailML interface that consents to load input data from RailML format. In particular, the RailML data necessary for the initialization of the framework relate to characteristics of the infrastructure (i.e. gradients, speed limits, positions of stations, switches), rolling stock (i.e. mass, composition, braking rate, tractive effort-speed curve), interlocking (i.e. alternative local routes), signalling system (e.g. position and type of signals), and routes/stopping pattern of the train services to be scheduled. These data are transformed by a RailML converter tool into a series of ascii files having a specific format readable by the microscopic model. In this way the microscopic model is initialized and all the other models are consequently built-up.

At this point, the microscopic model computes minimum running times *MinRT* considering timetable design parameters like turn-around times and minimum dwell times at stations. If a macroscopic timetable (*MacroTT*) has not been computed yet, operational running times are calculated by adding the minimum amount of supplement times (e.g. 5% of the running time) to the *MinRT*. Once a *MacroTT* has been produced, the operational running times are computed by respecting the time allowances (supplements and buffer times) and the arrival/departure times given by the timetable. In this case, the function *OperationalRunningTimesComputation* can also calculate the operational running times by identifying train time-distance trajectories that exploit timetable supplement times to minimize the energy consumption.

The operational running times are then set as input to *BlockingTimesComputation* that calculates blocking times for every train. If no *MacroTT* exists yet, blocking times are sent to the function *MinHeadwayComputation* to compute minimum headways. Otherwise blocking times are sent to the *ConflictDetection* module that verifies the feasibility of the timetable. If there are no overlaps of blocking times, the timetable is feasible and it is transferred to the *CapacityEvaluation* function that marks up the timetable as stable if the amount of supplement times respects the thresholds established by the UIC norms [8]. If instead the blocking times of consecutive trains overlap, the timetable is not feasible and new headway times must be computed that allow non-conflicting train services. The function *MinHeadwayComputation* is therefore activated both when no *MacroTT* is available yet, and when the *MacroTT* is infeasible.

The minimum headways are then transferred together with the minimum running times (from the micro model) to the macroscopic model. The latter computes a new *MacroTT* by means of mathematical algorithms that optimize a given objective function corresponding to e.g. maximizing throughput or minimizing waiting times. Once this macroscopic timetable is obtained, it is evaluated on robustness by estimating its ability of absorbing stochastic disturbances to operations.

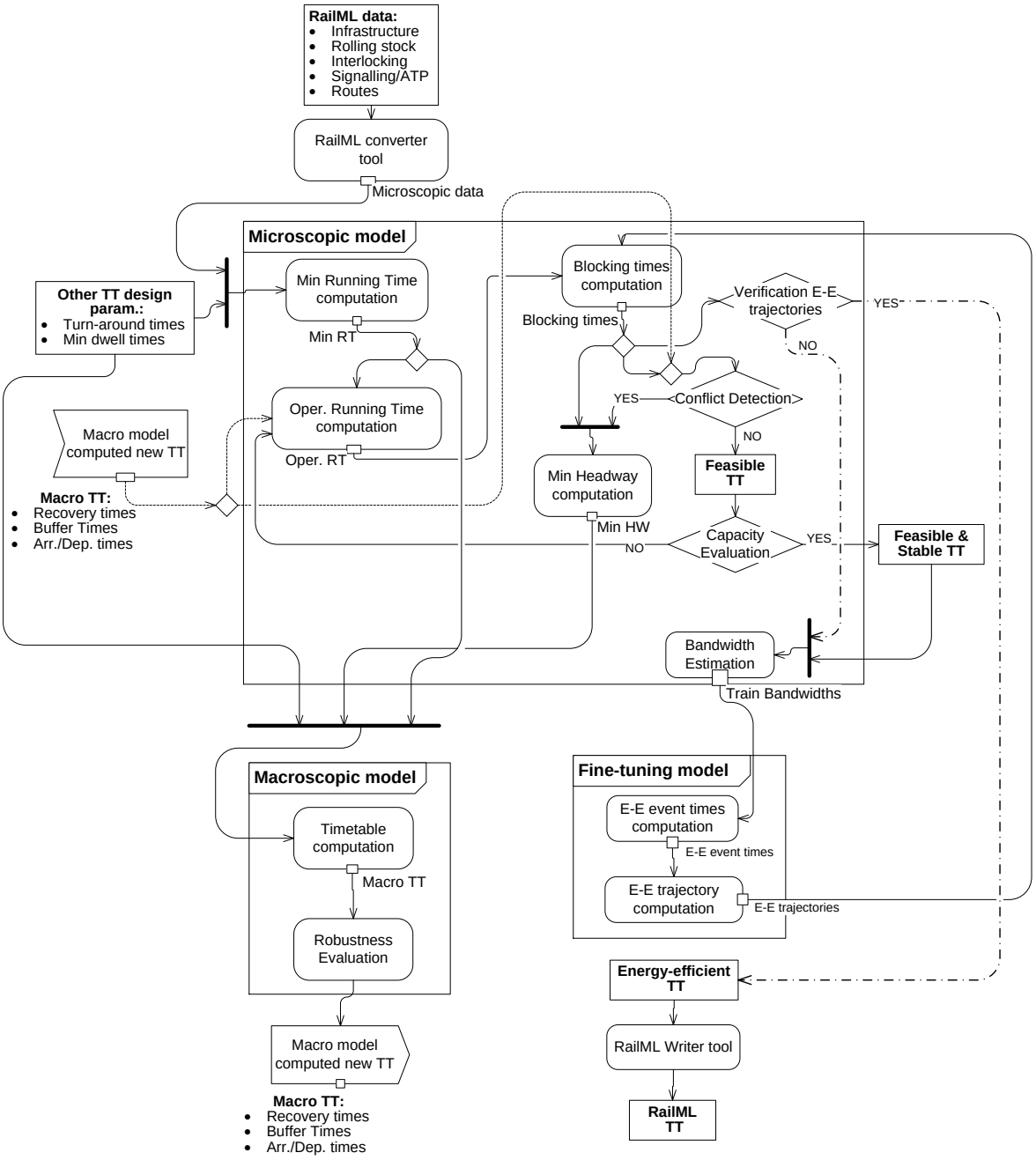

Figure 2. SysML scheme of the hierarchical framework for timetable design

If the timetable has an acceptable level of robustness, an event is triggered that activates the microscopic model by sending the *MacroTT*. At this point, the microscopic model computes again the operational running times and consequently, elaborates blocking times and verifies the feasibility of the timetable. This loop continues until the timetable produced by the macroscopic model is proved to be feasible.

If infrastructure occupation and stability norms are satisfied, we have a feasible and stable timetable, and the model proceeds to the final stage of the fine-tuning model with the computation of the existing bandwidths around train paths. The function *En-*

ergyEfficientEventTimesComputation computes the optimal arrival and departure times for local trains at all intermediate stops on corridors, associated with optimal train trajectories taking into account the stochasticity of dwell times. Successively all optimal time-distance trajectories corresponding to the optimal energy-efficient speed profiles are computed and checked by C*onflictDetection* to assert their feasibility, and the final results are saved in the RailML Timetable format. The final output of the framework is therefore a feasible, stable and robust timetable with improved energy-efficient train trajectories.

2.2 The hierarchical network modelling

The hierarchical framework for timetable design is composed of three models that respectively represent the same network with a microscopic, mesoscopic and a macroscopic level of detail (Figure 3).

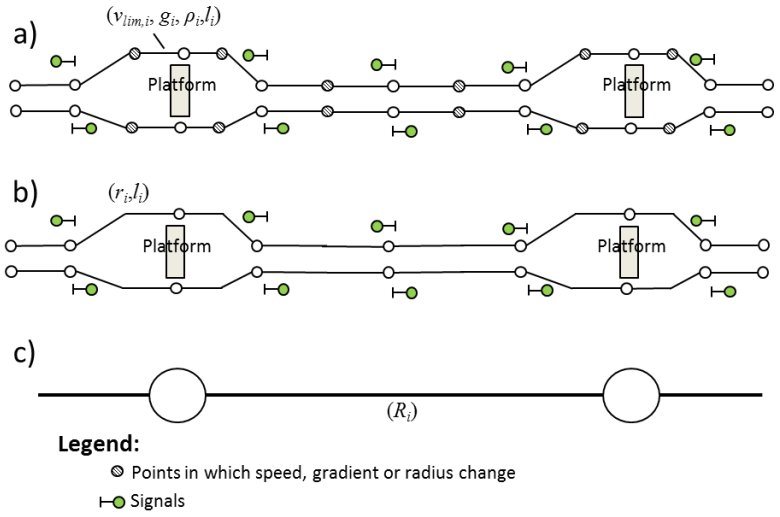

Figure 3. Infrastructure representation of the microscopic model: a) behavioural section, b) track section level and c) macroscopic models

The microscopic model $M = (B, X)$ represents the network at the level of homogeneous behavioural sections (i.e. sections with constant values of speed limit, gradient and curvature radius). This means that this model is a graph having as arcs infrastructure sections, $x_i \in X$, with constant characteristics of speed limit v_{lim}, gradient g_i and radius ρ_i as well as the given length l_i. The set of attributes $(v_{lim,i}, g_i, \rho_i, l_i)$ is associated to arc x_i. Microscopic nodes $b \in B$ represent either points in which these characteristics change, and infrastructure elements like signals, switches, platform tracks, etc.

The mesoscopic model aggregates the homogeneous behavioural sections in block sections for open tracks, and track-free detection sections for interlocking areas. With this kind of aggregation it is possible to consider sectional route release within interlocking areas. The *i*-th arc of such a model has two attributes (r_i, l_i), namely the running time r_i on the arc (as computed by the micro model) and its length l_i. The nodes represent both the joints between consecutive block or track-free detection sections and infrastructure elements such as signals, switches, platform tracks, etc. On top of a

microscopic network, a set of timetable points R, is defined. A timetable point represents an infrastructure point that determines an interaction between a train and passengers (boarding and alighting), or cargo (loading and unloading), or between two trains (converging and diverging tracks). These timetable points are important for planning and operational purposes at stations, junctions, bridges or tunnels.

The macroscopic model aggregates microscopic arcs into corridors between two timetable points resulting in a graph $N = (P, A)$. The creation of a graph N is based on the train line requests. Each macro node $p \in P$ defines a timetable point that allows possible interaction between train lines, i.e., meeting, overtaking, or connection. Each node is determined with its name, type, number of tracks and a set of train lines serving this timetable point. An arc $a \in A$ depicts the corridor between two macroscopic nodes and is attributed with the corresponding number of tracks, the direction of traffic flow (uni-/bidirectional) and the corridor (macroscopic) running time as aggregated from the micro model. Finally, the stops on open track are not considered as elements of P as here is no interaction at macroscopic level.

A train line path request, $l \in L$, consists of a) a list of timetable points, $r^l \subset R$, b) a microscopic route between successive timetable points, c) train characteristics, $t_l \in T$, including rolling stock formation, mass, and train dynamics parameters, d) a service intention presenting the desired periodicity in case of a periodic timetable or a desired departure time in case of a non-periodic timetable, and e) (optionally) minimum dwell time per station.

The procedure for aggregation to a macroscopic network is described in Algorithm 1. First, macroscopic nodes *p* are determined. In a microscopic model M timetable points are all stations, junctions, crossings, and bridges, while macroscopic nodes exclude stations (stops) on an open-track. At this point also a set of train lines L_p using node p is defined. Second, arcs $a \in A$ are derived for each pair of adjacent macro nodes. For each arc a, the following are determined: *a*) the number of tracks by identifying different routes between two nodes using function *DetermineTracks*, and *b*) the flow direction for each of them by function *DetermineDirection*. These two functions are based on detection of routes overlapping, converging and diverging.

Algorithm 1 *MacroNetworkCreation*

Input: set of train lines L, microscopic timetable points R
forall $r \in R$
  if r not 'open track station' //Make aggregated set of macro nodes//
    Create $p = r$;
    $L_p = \{l |\ l \text{ uses } r\}$; //Make set of train lines using the station//
    $P = P \cup \{p\}$;
  end
end
forall $l \in L$
  forall adjacent $p_i, p_{i+1} \in P$
    if $(p_i, p_{i+1}) \notin A$ //if not already in the list *A*//
      Create $a = (p_i, p_{i+1})$;
      DetermineTracks(a);

DetermineDirection(a); //uni- or bidirectional//
$A = A \cup a$;
end
end
Output: $N = (P, A)$

The benefits of our approach based on train line requests for the construction of a macroscopic network are the following:

a) The complexity of the macroscopic network is determined by the complexity of given train lines and their interactions, and not by the complexity of the network topology,
b) All potentially conflicting points are considered,
c) Steady train driving profiles are provided. For example, an intercity train between to stations will have less arcs (in the optimal case only one), while each arc defines a certain time supplements. Less number of different time supplements between two adjacent stopping points is beneficial, as this provides a more equalised allocation of time supplements and less changes in driver behaviour/style afterwards.
d) A considerable degree of freedom is provided for the fine-tuning of timetable times (particularly for stations that are not in *P*), while increasing energy-efficiency of the timetable.

The three networks defined on microscopic, mesoscopic and macroscopic model are used in the various timetabling modules and their interaction as is described in Chapter 3.

3 MICROSCOPIC CALCULATIONS

This chapter details the microscopic timetabling module which consists of a set of microscopic calculation tools. Successively, this section considers the objectives addressed in the microscopic timetabling module, the input data, the output data, the algorithms, and the network transformations to transform microscopic to macroscopic data and conversely.

3.1 Objective

The microscopic calculation tools focus on providing a feasible, stable and robust timetable and giving constraints to the fine-tuning module that provides an energy-efficient timetable. Particularly, we consider the following performance measures defined in D1.2 [5]:

- *Journey time*: Process times are minimised to a certain extent considering that slightly extended process times allow more robust timetable, which is also in the scope of timetable planning [7]. Therefore, the focus is on having reliable running times (and not only minimal) in order to absorb variations in train driving behaviour as well as a solid dwell times that include passenger-related deviations in time needed for alighting and boarding.
- *Stability* (RS1): stability is tackled on the microscopic level by setting time allowances to the minimum process times.
- *Resource Usage*: The infrastructure occupation is determined and minimised iteratively with the macroscopic module until the UIC norms (or any nationally determined norms) are satisfied.
- *Energy*: The microscopic computation model finally determines the bandwidths that are used as constraints in the fine-tuning module to compute train speed profiles enabling energy-efficient driving behaviour.

3.2 Input data

3.2.1 Input data from the architecture database

The input from the basic architecture consists of a set of RailML files (Figure 4) composed of:

a) Microscopic infrastructure data
b) Rolling stock data, including train formations
c) Interlocking, signalling and ATP
d) Available routes
e) Train line requests.

These data are converted to an internal data format of lists in ascii that is used by the microscopic computation models.

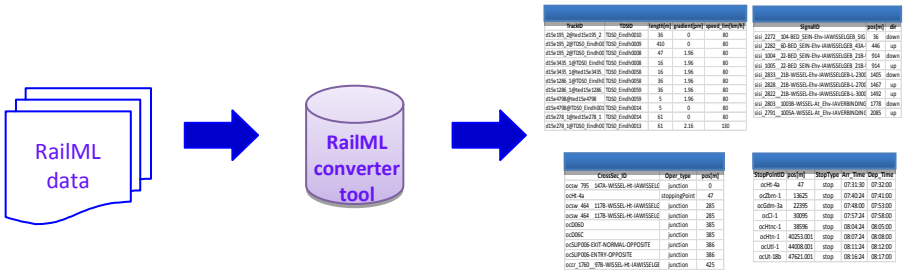

Figure 4. Input data from the architecture

3.2.2 External input

Additional external input given via the user interface is the following.

a) *Dwell times*, either deterministic values or probability distributions

Dwell times can be accurately determined based on realisation data that may be collected by means of e.g. track occupation data, GPS, or tracking sensors on train doors. Based on these recorded data a probability distribution and the corresponding parameters for the *minimum dwell times* can be fitted. Different station types and train types can be considered separately. Nevertheless, if realisation data are not available, also relevant norms or user defined values can be used.

b) Timetable design parameters and norms

Since the aim of timetable planning is to provide an acceptable quality of service, certain *performance parameters and norms* need to be predefined. The set of norms consists of minimum transfer times, turnaround times, minimum running time supplements, and maximum allowed journey times of train lines. The set of *Timetable design parameters and norms* is named *Q*.

3.2.3 Input data from the other timetabling modules

Input from the macroscopic model is a macroscopic timetable that consists of all scheduled event times at all macroscopic timetable points, i.e., arrival and departure times.

Input from the fine-tuning model is a microscopic timetable with all scheduled event times at all microscopic timetable points as well as the train speed profiles corresponding to energy-efficient train trajectories computed by applying an optimal control algorithm.

3.3 Output data

3.3.1 Output data to the macroscopic module

The output data to the macroscopic model are given in an ascii file with an internal data format and consists of:

i) Minimum running times, i.e., the free-flow running times of a train assuming that it runs at the maximum allowed speeds;

ii) Operational running times, i.e., the running time including time allowances that should be scheduled in the timetable;
iii) Minimum headway times between train pairs at macroscopic points.

Recall that minimum dwell times, turnaround times and timetable norms are provided as external input. The microscopic timetable module (re-)computes and checks the performance of the train-dynamics related process times.

3.3.2 Output data to the fine-tuning module

The output to the fine-tuning model is given as lists of ascii data according to an internal data format, and consists for each corridor of:

i) The scheduled event times of the macroscopic timetable points;
ii) The available time bandwidth for each train.

3.3.3 Output data to the architecture

The output of the microscopic module is a conflict-free, stable, robust timetable with energy-efficient train speed profiles provided in the RailML timetable format. It consists of:

i) Scheduled event times at all (microscopic) timetable points; and
ii) Microscopic train speed profiles.

Recall that the microscopic module checks the output of the corridor fine-tuning module and then converts all information in a RailML timetable format.

3.4 Algorithms

We separate between functions working on the behavioural section level of the infrastructure network on one hand (the microscopic level), and track-free detection section and block section level on the other (the mesoscopic level). The functions applied on the behavioural section level are the following:

i) *Minimum running time computation,* and
ii) *Operational running time computation.*

The functions applied on the track-free detection and block section level are the following:

i) *Blocking times computation*
ii) *Minimum headway time computation*
iii) *Conflict detection*
iv) *Capacity evaluation*
v) *Bandwidth estimation.*

3.4.1 Minimum running time computation

The minimum running time is the time required for driving a train from one infrastructure point to another assuming conflict-free driving as fast as possible. In this section we will focus on the running time between two stations. Running times are computed

from microscopic train dynamic models that require detailed rolling stock and infrastructure data, including route-specific static speed profiles.

Running times are computed for every line service by means of dynamic Newton's motion equations [4]. The tractive effort is assumed a piecewise function of speed consisting of a linear and one or more hyperbolic parts. The resistance force is modelled based on the Davis resistance equation, a second-order polynomial of speed. The braking rate is defined as a single deceleration rate. A (dynamic) train speed profile and associated running time are modelled as a function of speed depending on distance and a function of time over distance, respectively [1]. The given equations are autonomous first-order ordinary differential equations which are solved by the numerical Dormand-Prince method [3], which is a particular application of the more general Runge-Kutta approach.

The algorithm *MinimumRunningTimesComputation* describes the computation of running time for one section between two consecutive stops, which is applied consecutively if more stops occur in the itinerary. The algorithm is initialised by providing microscopic infrastructure data *M*, train formation *t,* itinerary *r*, initial speed V_{start}, and the end speed V_{end}. Speeds $V_{start}$and V_{end} are equal to zero if the train begins and ends the run from/to standstill. Alternatively, a train can enter the network running at a certain speed $V_{start} > 0$. The computation of the train trajectory (given as a triple [*V,s,t*] of speed, distance and time) is done sequentially for each homogeneous track behavioural section. By doing so, the current speed $V_{current}$ is being tracked.

For the minimum running time computation we differentiate between four driving behaviour phases: accelerating, cruising, decelerating and braking. Decelerating is caused by insufficient train power to maintain a constant speed (e.g., due to a steep slope). Each phase is applied if certain conditions are met:

a) The train accelerates if the speed limit is higher than the current speed
b) The train cruises if the speed limit is equal to the current speed and the engine power is sufficient to overcome the track and train resistances.
c) The train decelerates if the engine does not have sufficient power to preserve the current speed, $f_t < f_r$.
d) The train brakes if the train approaches a given speed limit lower than the current speed $V_{current}$. Then, the braking curve is back-propagated from the point where the lower speed limit should be realized backward until the intersection with the forward trajectory [*V,s,t*] is found.

The output of the algorithm consists of microscopic train trajectories: time-distance and speed-distance diagrams, and minimum running times.

Algorithm 2 *MinimumRunningTimesComputation*

Input: Network *M* = (*B*,*X*), *signals*, *t*, *r*, V_{start}, V_{end}
$V_{current} = V_{start}, [V, s, t] = []$;
forall microscopic points *b, x, signals*
  if $V_{current} < V_{limit}$
    Accelerate();

elseif $V_{current} = V_{limit}$
if $f_t > f_r$(train has enough power)
Cruise();
else
Decelerate();
end
else //brake//
while BrakeCurve ∩ TrajectoryComputed == []
BrakeBackpropagate();
b--;
end
end
end
Output: detailed train trajectory, time-distance and speed-distance diagrams

Algorithm 2 represents the basic running time computation model. This model is applicable for signalling and protections systems like ETCS level 1 and ETCS level 2. However, driver behaviour tends to differ due to the different signalling and ATP systems. For example, the Dutch system needed an additional procedure to be developed. The running time model complying with the Dutch ATP is given in Algorithm 3. It should be noted that function *BrakeATB*(), differs from the braking in Algorithm 2. The train starts braking at the signal where the reduced speed is signalled to a train driver and the target speed has to be reached before the following signal. In a similar manner, the train behaviour determined by different ATP systems can be modelled using function *MinimumRunningTimesComputation* as the core of the algorithm.

Algorithm 3 *MinimumRunningTimesComputation_ATB*

Input: Infrastructure M=(*B*,*X*), *signals,* train formation, route, V_{start},V_{end}
forall *b, x, signals*
if *signal* ~= *'Approach'* //yellowAspect//
MinimumRunningTimesComputation
else //start braking//
while $V_{current} < V_{expected}$
BrakeATB();
end
end
end
Output: detailed train trajectory, time-distance and speed-distance diagrams

The running time for a train line is computed consecutively between each two following stopping points, from standstill to standstill ($V_{start} = V_{end} = 0$). Afterwards, it is been split to running times for every pair of infrastructure points having pass events in nodes. By this disjointed approach it is possible to preserve all characteristics of the train trajectory (speed-distance). Note that if a running time would be computed between each two points this could lead to a discrepancy between V_{start} of the current section and V_{end} of the previous one. In that case at least one of these speed profiles would have to be recomputed, and this would increase total computation time.

3.4.2 Operational running time computation

To initiate function *OperationalRunningTimeComputation*, the macroscopic timetable has to be computed first. MacroTT is defined as the arrival and departure (or passing) times at nodes. As such it represent just the scheduled (macroscopic) running times between two nodes. This is not sufficient for microscopic analyses and therefore the transformation from macro to micro should be undertaken. By doing this, we obtain detailed train trajectories that incorporate the available time supplements. The operational running time represents the recomputed train trajectory (speed-distance, time-distance) that satisfies the scheduled running time between two nodes. Further, the time supplements can be used by: *i*) cruising with a speed lower than the maximum speed, or *ii*) applying optimal control algorithms to determine possible coasting regimes. The latter models provide energy-efficient driving. However, those algorithms are more time consuming. Therefore in the phase where the *macroTT* has not been fully determined yet, it is sufficient to use the former approach. In this way it is possible to keep a satisfactory accuracy with high computation performance.

In order to acquire the operational profile we propose the usage of an *operationalParameter* [%] that represent the ratio between the given static speed limit and an actual (lower) speed that should be used to the consume all time supplement. The *operationalParameter* takes a value between 0 and 100. An *operationalParameter* = 100% represents the minimum running time. This parameter is applied on open-track corridors in order to exploit the running time supplement, while maintaining the maximum speed through stations with restricted speeds (i.e., possible bottlenecks due to the various conflicting routes). The running time with respect to the *operationalParameter* is computed by Minimum*RunningTimeComputation* for an adjusting microscopic infrastructure built in a function *editInfrastructure.*

The function *OperationalRunningTimeComputation* uses an adjusted bisection algorithm to find an *operationalParameter* with a corresponding *OperationalRT* and speed profile as described in detail in Algorithm 4. The function inputs are the scheduled event times from the MacroTT and microscopic minimum running times minRT as well as a tolerated error *errorTolerance* [s] between the operational running time (*operationalRT*) and the scheduled one (*targetRT*). Initially, the operational running time is set equal to the minimum running time and the absolute error absError between the scheduled and this operational running time is just the allocated running time supplement determined simply by subtracting the minimum from the scheduled running time. Also initial values for the upper (*ub*) and lower bounds (*lb*) for the *operationalParameter* are introduced. The algorithm iteratively: 1. computes the running time for the given *operationalParameter*, 2. Adjusts *ub*, *lb* and *operationalParameter* and 3. Computes the absolute error *absError*. Steps 1-3 are repeated until the *absError* is larger than *errorTolerance.*

Algorithm 4 *OperationalRunningTimesComputation*

Input: Network $M=(B,X)$, MacroTT, minimum running times, *errorTolerance,* train lines L
forall pairs (train,arc)

timeSupplement = scheduled running time - minRT
ub = 100
lb = 10
operationalRT = minRT
targetRT = minRT + timeSupplement
absError= |targetRT – operationalRT|
operationalParameter = 100
while *absError > errorTolerance*
M_operational=editInfrastructure(B,X, operationalParameter)
operationalRT = MinimumRunningTimeComputation(*M_operational,signals,t,*
route, V_{start},V_{end})
if *targetRT – operationalRT* > 0
`lb = operation_parameter-(ub-lb)/2`
operationalParameter = lb
else
`ub = operation_parameter-(ub+lb)/2`
operationalParameter = ub
end
absError= |targetRT – operationalRT|
end
end
Output: List of operational running time Z

Figure 5 shows the speed-time trajectories computed by the micro model relative to the minimum running time (part *a*) and the operational running time that uses an energy-efficient coasting phase (part *b*).

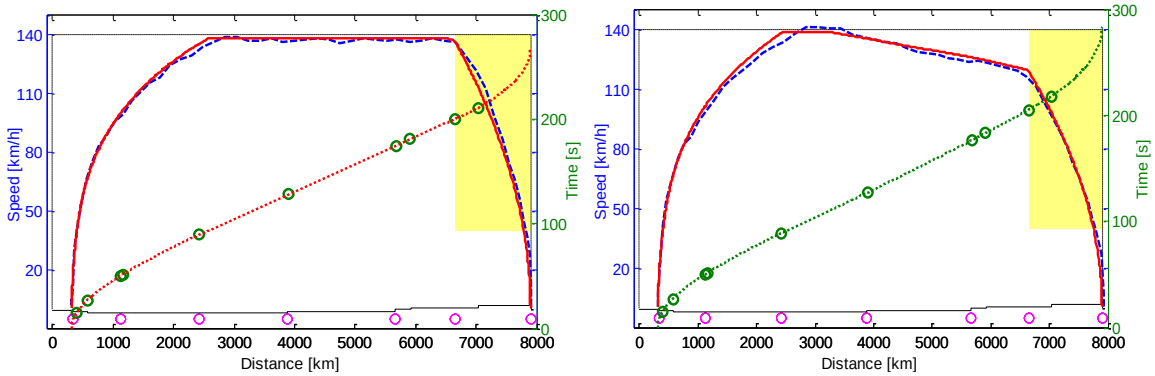

Figure 5. Speed-distance diagrams: a) minimum running time, b) optimal energy-efficient trajectory

3.4.3 Blocking times computation

The blocking time of a section of track (block section or interlocked route) is the time interval that the section is exclusively allocated to a train and therefore blocked for other trains. Blocking times are computed in function *BlockingTimeComputation* by applying the procedure to build up blocking times as described in [4]. If a macroscopic timetable is not available yet, blocking times are computed considering the time-distance trajectories that give the minimum running times. Otherwise blocking times

are calculated based on the operational distance-time trajectories as given by the timetable.

To compute blocking times, the following data is required:

- Topological data including track sections, block sections, overlaps, and clearing points for routes and switches, including lengths or positions
- Running times t_{block} [s] over block sections and track sections in interlocking areas,
- Train lengths l_{train} [m] or clearing times t_{clear} [s] of block sections,
- Setup times t_{setup} [s] for routes and blocks,
- Release times $t_{release}$ [s] for routes and blocks,
- Sight distance l_{sight} [m] or sight time t_{sight} [s] for trackside signals,
- Reaction times $t_{reaction}$ [s] of the driver.

Given the set of blocks *B,* and all relevant time elements the equation for computing the blocking time of the i-th block is as follows:

$$T_i = t_{setup} + t_{sight} + t_{reaction} + t_{apprach} + t_{block} + t_{clear} + t_{release},$$

where $t_{apprach}$ is the running time over a number one or ore preceding blocks depending on the implemented signalling system. For example, for a fixed two-block signalling system $t_{apprach}$ is equal to the running time over the preceding block, i.e., $t_{apprach,i} = t_{block,i-1}$.

Alternatively, if a sight time is not given, then l_{sight} has to be provided and t_{sight} is computed as:

$$t_{sight} = l_{sight}/V,$$

if a train runs with a constant speed, or it is determined from a microscopic train trajectory for a given sight distance l_{sight}.

If a clearing time is unknown then the overlap length $l_{overlap}$ and train length l_{train} must be known to compute:

$$t_{clear} = (l_{overlap} + l_{train})/V.$$

For example, $l_{overlap}$ for the Italian signalling system is defined as the whole following block, i.e., $l_{overlap,i} = l_{block,i+1}$.

The setup time for an interlocking area can be given as a predefined value or calculated based on the route that a train traverses. Then, an equation to do so in the case of successive point control is:

$$t_{setup} = (n_{switch} + 1) \cdot t_{switch} + t_{action},$$

where $t_{switch}[s]$ is a time needed to operate a single switch, n_{switch} is number of switches to be operated along the route, and $t_{action}[s]$ is the time to activate the route order and to control the setup route.

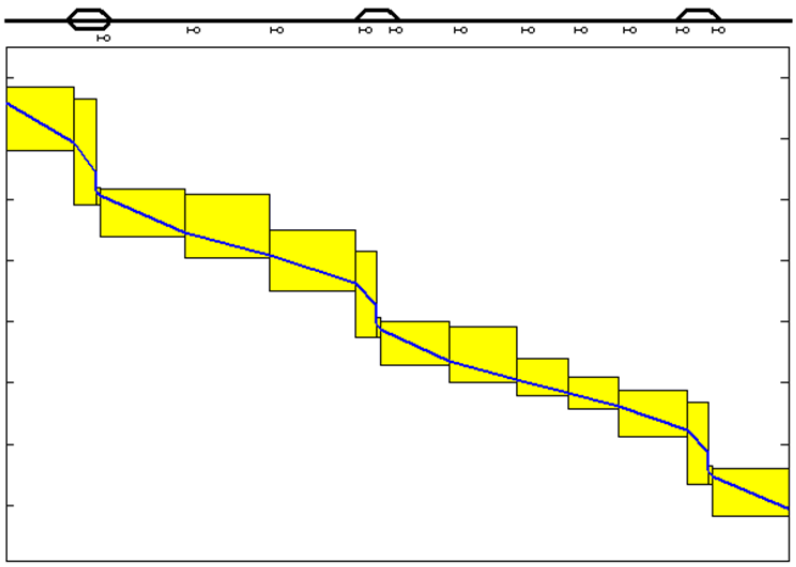

Figure 6. Blocking time stairway

The input for *BlockingTimesComputation* are train lines characteristics *L*, microscopic operational running times *Z*, dwell times *D,* and a set of predefined technical parameters, *tp*, such as clearing time, sighting distance and reaction time. The output is represented with occupation and release times for each train line along the route, $o^l \in O$ and $d^l \in D$, respectively.Formally, the function can be written as $[O, D] = BlockingTimesComputation(L, Z, D, tp)$.

3.4.4 Minimum headway time computation

For every station and junction of the network (i.e. for every node of the macro model) minimum headway times are computed for each pair of consecutive trains. Specifically, a set of minimum headways is calculated for all the possible cases of interactions between two consecutive trains: e.g. both trains leaving a station, both trains entering a station or one entering and the other leaving. For each one of these cases minimum headways are calculated by applying the UIC compression method where trains are shifted in order that their blocking time stairs are touching but not overlapping. The compression and the computation of the minimum headways are performed by using the method proposed by Van Egmond [9] which is based on max-plus algebra.

The pseudocode of minimum headway computation is given in Algorithm 5 and gives the detailed description of the applied approach. In macro network we recognize three types of timetable points: regular station, end (terminal) station, junction. A junction allows only passing through events, so it is sufficient to compute only one headway for two trains passing the station (*pass-pass*). An end station is a station that are considered as the terminus stations for the defined network, where trains end (or start) their journey. For these stations it is enough to compute only one headway for each pair of sequenced trains, cause all trains naturally stop in (i.e., arrive or depart) and all interaction happens on one side of the station. Finally, a regular station has at least two incoming (and outgoing) directions and hence it defines mode interaction possibilities

for a given two trains. These trains may use dependent incoming and/or outgoing routes. Therefore, each train movement through a station is determined by inbound (*in*) and outbound (out) route, and the interplay of all routes is considered separately. So, four minimum headway pairs for each ordered train pair are distinguished:

1. Inbound route of 1st train – inbound route of 2nd train
2. Inbound route of 1st train – outbound route of 2nd train
3. Outbound route of 1st train – inbound route of 2nd train
4. Outbound route of 1st train – outbound route of 2nd train

So far explained algorithm holds for the most of the variations of train pairs in terms of direction (same of opposite) and the events in considered station (arrival, departure, pass through). However, a special case of two trains using the same platform track has to be treated in a different manner. Let us consider two trains A and B, and both scheduled to stop in a station at the same track platform. If we want to compute the minimum headway (1), mathematically is possible to obtain a certain value, but physically is not possible to let the 2nd train on the platform while the 1st train is still there (this is valid for common platform tracks where the platform is not equipped to accept two trains simultaneously), i.e., not yet departed. In this case, the 2nd train is allowed to enter the platform only after the 1sttrain leaves. Therefore, in this case inbound and outbound routes have to be considered as one train movement and hence, it virtually represents a pass-pass headway.

The output of *MinHeadwayComputation* is a list of headway times computed for each pair of trains and corresponding train movements. One headway, $h \in H$, is determined by a tuple (p, l_1, l_2, m), where *p* is a station, l_1 and l_2, corresponding trains and *m* one of four train interactions as defined above.

Algorithm 5 *MinHeadwayComputation*

Input: macroscopic timetable point *p,* train lines *L*, occupation and release times
forall $[l_1, l_2] \in p$
  if $p == 'junction'$
    compute only pass-pass headway time;
  elseif $p == 'end\_station'$
    compute only in-out OR out-in OR in-in OR out-out headway;
  elseif $p == 'station'$
    if $platfromTrack(l_1) == platfromTrack(l_2)$
      compute only pass-pass headway time;
    else
      compute headways for all in/out combinations;
    end
  end
end
Output: list of minimum headway times *H*

The following example shows how this method works to compute the minimum headway time between two trains A and B passing through the same station (Figure 7-8). These two trains are scheduled to depart at the same time but they have different routes: sections 1, 2, 3, and 5 for train A, and sections 4, 3, 2, and 1 for train B. The minimum headway between these trains is obtained by shifting the departure of one train (in this case train B) in order that their blocking time stairs are just touching. The minimum headway for the station is then identified as the distance between the original departure time of train A (that has not been shifted) and the shifted departure time of train B.

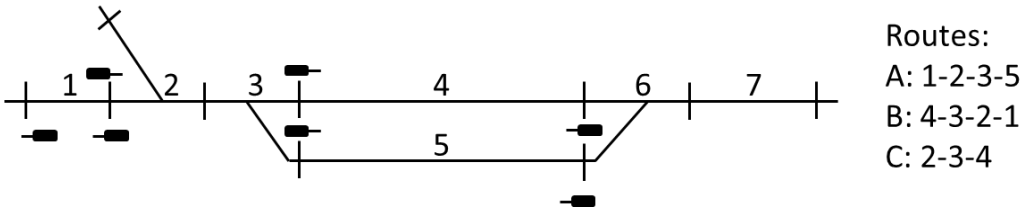

Figure 7. Infrastructure example

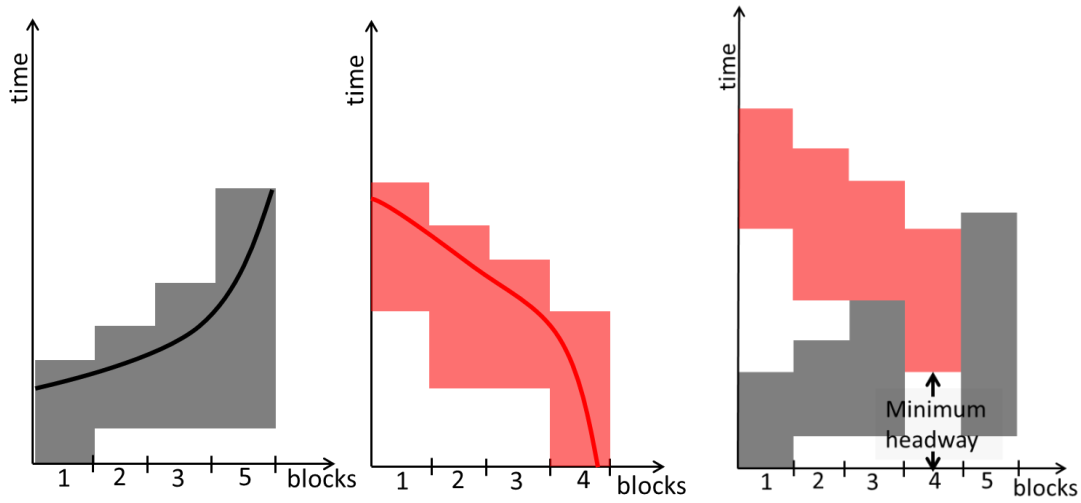

Figure 8. Blocking times of trains A (a) and B (b), and minimum headway (c)

3.4.5 Conflict detection

The function *ConflictDetection* is activated only when a macroscopic timetable has been produced. The aim is to verify the feasibility of the macroscopic timetable by checking: *a*) the absence of track conflicts and *b*) the realizability of scheduled event times (i.e., running times, dwell times, turnaround times). Track conflicts are detected as overlaps of the blocking times provided by the *BlockingTimesComputation* function. Therefore, conflict-freeness is tested comparing each scheduled headway time with the analogous minimum ones, i.e., asserting that allocated headway buffer times are nonnegative. Second, realizability is tested by whether scheduled running and dwell times extend the corresponding minimum values. If the timetable is feasible, it is evaluated in terms of infrastructure occupation and stability. Otherwise, it is necessary

to compute new minimum headways that allow conflict-free train operations. Figure 9 shows the conflicted train paths, while Figure 10 gives the resolution of that conflict.

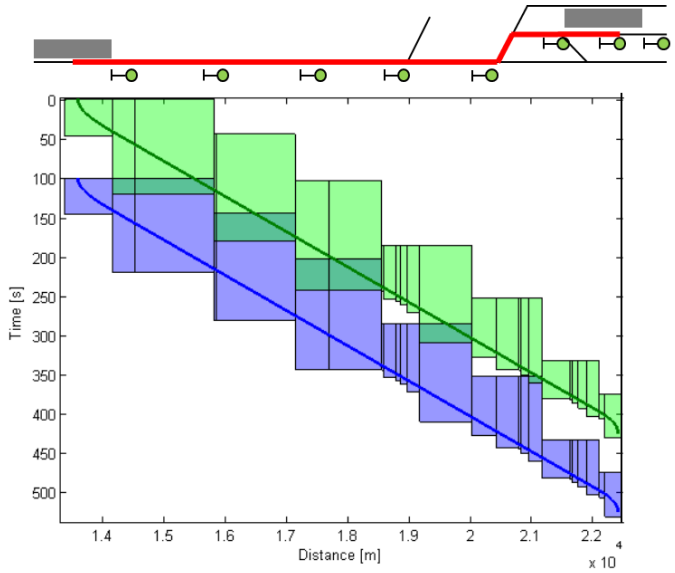

Figure 9. Example of conflicting trains

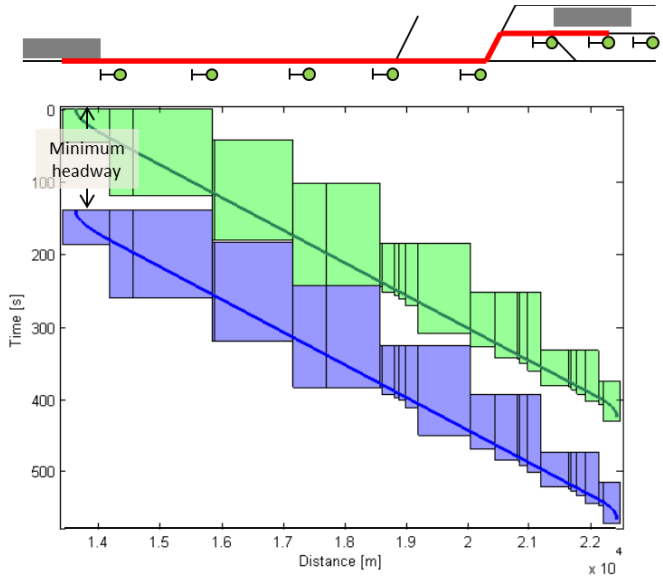

Figure 10. Example of a resolved train conflict

3.4.6 Capacity evaluation

Feasible macroscopic timetables are evaluated by the *CapacityEvaluation* function in terms of capacity as established by the UIC code 406. *CapacityEvaluation* defines three performance parameters:

i) *Infrastructure occupation* is defined as the minimum time needed to operate trains according to a given timetable pattern. To determine this minimum time it is necessary to compress all train paths scheduled in a given time period. This compres-

sion is performed by using again the method proposed by Van Egmond [9]. For example, if in one hour the timetable schedules three train paths A, B and C (Figure 11), the infrastructure occupation is obtained by compressing all trains. Note that train A is added twice for practical reasons, i.e., in order to determine the earliest possible departure of a train from the following period. The yellow line represents the infrastructure occupation for schedule ABC. However, the white space (between black and yellow line) represent unused capacity which might be used to add an extra train.

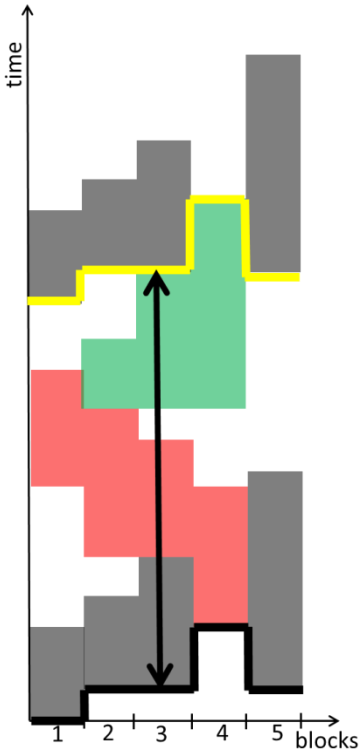

Figure 11. Infrastructure occupation for schedule ABC (d)

ii) *Capacity consumption* is defined as the time share needed to operate trains according to a given timetable pattern taking into account scheduled running and dwell times (instead of the minimum ones) and buffer times. It is computed analogously to the infrastructure occupation.

iii) *Stability.* A timetable is called stable if any train delay can be absorbed by the time allowances in the timetable without active dispatching. Time allowances are defined as the difference between capacity consumption and infrastructure occupation. Therefore, the larger the time supplements and buffer times the better is the ability of the timetable to prevent initial and secondary delays, i.e., the timetable is more stable. If the amount of time allowances is higher than the amount recommended by the UIC code 406, the timetable is considered as stable. Otherwise it is defined unstable and operational running times must be recomputed by e.g. adding additional running times supplement and/or buffer times.

3.4.7 Bandwidth estimation

A train bandwidth defines the flexibility of the fine-tuning optimization process at timetable points. The function *MinHeadwayComputation* is used to estimate them. The minimum headways with a preceding and a following train determine the upper and lower bound for each bandwidth. They are represented by the earliest arrival and the latest departure time at each timetable point.

3.4.8 Aggregate process times

The function *AggregateProcessTimes* is introduced to provide mitigation from microscopic running times (i.e., between each two timetable points) to aggregated process times between two nodes in the macroscopic network that are needed for macroscopic computation. The input for this function is the macroscopic network $N = (P,A)$, a set of microscopic running times Z, and minimum dwell times D. For example, if there exists between nodes p_1 and p_2 two stopping points b_1 and b_2, *then* the total process time for the pair $[p_1, p_2]$ is the sum of all running times between p_1 and p_2 and dwell times in b_1 and b_2. The output are the aggregated process times, i.e., arc weights for macroscopic computation.

3.4.9 Time granularity

The computed process times in the microscopic model are expressed in seconds. However, current macroscopic models do not have the computational capability to accept this level of accuracy, so the time granularity of process times need to be reduced, i.e., discretized. Therefore, one can accept a granularity of 6, 10, 30 or more seconds. In the ON-TIME framework we use a 10 s granularity. It is also important to apply this process in the latest phase of computations, i.e., after computing all process times. The rounding of all times is done in the function *TimeGranularity*, for the given time discretisation factor θ. Always rounding to the upper value leads to allocation of unnecessary reserve times and hence overestimated capacity consumption, while rounding down gives unrealizable process times. Therefore, we use an innovative smart rounding method, that has the objective to control the rounding error by combining rounding up and rounding down.

The function takes as input the following data: the macroscopic network N, a set of aggregated process times Z_{agg}, headway times H, dwell times D, timetable design parameters Q, and granularity factor θ. The output includes (rounded) macroscopic process times Z_{macro}, dwell times D_{macro} and headways H_{macro}, which are necessary for macroscopic computations.

3.5 Network transformations

3.5.1 Microscopic to macroscopic conversion

We developed a conversion algorithm for transforming from the microscopic to the macroscopic model. The procedure is described in Algorithm 8. In the first phase, the initialisation of the network has been done. After that, the algorithm performs the

computation of minimum running times on the microscopic model (homogenous behavioural sections), which is followed by the computation of blocking times and minimum headways. Headways are determined for all possible interactions between each two trains. The last two are executed on the block section level of infrastructure network.

Once all process times are computed on the microscopic model, we carry out the aggregation of process times based on the given macro network *P* and the time discretisation by θ. By doing so, we obtain macroscopic process times that are necessary for macroscopic computation.

Algorithm 8 Micro-to-MacroConversion

Input: Micro network *M*=(*W,B*), dwell times *D,* timetable design parameters *Q,* granularity factor θ

$N = MacroNetworkCreation()$
$Z = OperationalRunningTimesComputation(l, r, b, errorTolerance, Z_{macro})$
$[O, D] = BlockingTimesComputation(L, Z, D, tp)$
forall $p \in P$
  forall m //all interactions of train lines in a macro node//
    $h(p, l_1, l_2, m) = MinHeadwayComputation\ [p, o^{l_1}, d^{l_1}, o^{l_2}, d^{l_2}]$
  end
end
$Z_{agg} = AggregateProcessTimes(N, Z, D, Q)$
$(Z_{macro}, D_{macro}, H_{macro}) = TimeGranularity(Z_{agg}, D, H, \theta)$
Output: $N = (P, A), Z_{macro}, D_{macro}, H_{macro}$

3.5.2 Macroscopic to microscopic conversion

After obtaining a macroscopic timetable, a posteriori analyses has to be undertaken. Namely, the timetable properties as feasibility, stability and robustness have to be satisfied. For that purpose we defined a macroscopic to microscopic conversion algorithm (Algorithm 9).

First, from the event times in *macroTT* the algorithm determines: *a*) macroscopic process times, and *b*) the distributed time allowances. These values are used to compute detailed operational running times and blocking times for each line. Sequentially, *ConflictDetection* and *CapacityEvaluation* are performed, and if one of those is not satisfied, the new process times (headways and running times) are computed and sent again to the optimisation model (*runMacroModel*) to recompute *macroTT*. After a new *macroTT* is acquired, the Algorithm 9 starts from the beginning. Once *macroTT* satisfies an acceptable quality of service, *BandwidthEstimation* is applied to all train lines and send to the *fine-tuning model*, which computes microscopic energy-efficient train trajectories using an optimal control algorithm. The final output of the model is a feasible, stable and robust timetable with energy-efficient train trajectories.

Algorithm 9 Macro-to-MicroConversion

Input: micro network *M*, macroscopic event times *E*, lines *L*, macroscopic process times Z_{macro}, D_{macro}, headway times H_{macro}, timetable design parameters *Q*, discretization factor θ

forall Lines $l \in L$
    $OperationalRunningTimesComputation(l, r, b, errorTolerance, Z_{macro})$
end
$ConflictDetection()$
if timetable is not conflict-free
  Recompute new macroscopic process times
  runMacroscopicModel
else
  $CapacityEvaluation()$
  if capacityNorms are not satisfied
    Recompute new macroscopic process times
    runMacroscopicModel
  else
    BandwidthEstimation
    while not verified E-E train trajectories OR not existing E-E train trajectories
      BandwidthEstimation
      Send list of relevant trains to fine-tuning model
      Verification of energy-efficient train trajectories
    end
  end
end
Output: RailML timetable

4 MACROSCOPIC TIMETABLE COMPUTATION

This chapter details the macroscopic timetabling module which consists of an optimization model. Successively, this section considers the objectives addressed in the macroscopic timetabling module, the input data, the output data, and the optimization model and solution algorithm.

4.1 Objective

The objective of the macroscopic timetable computation is to provide a timetable that is feasible, from a macroscopic point of view, and achieves the following goals indicated in Deliverable 1.2:

- *Transport Volume*: within the time horizon considered, the passenger/cargo-tonne delivers is maximized;
- *Journey Time*: the journey times of the trains are minimized;
- *Connectivity*: the connection times between different pairs of trains are minimized;
- *Resilience*: the delay propagation within the system is minimized by providing both stable and robust timetables;
- *Resource Usage*: the track occupation is minimized.

The only objective, among those defined in Deliverable 1.2, that is not considered by the macroscopic timetable computation is *Energy Consumption*, which is considered at a later stage by the Corridor Fine-Tuning module.

4.2 Input data

The macroscopic timetable computation module receives as input the following data (all provided by the microscopic calculation module):

- Macroscopic railway network
- Trains, paths and routes
- Headway times
- Running and dwell times
- Connections

4.2.1 Macroscopic railway network

The railway network is represented by a multi-graph $N = (S, A \cup E)$, where the vertices represent a set S of stations (corresponding to commercial and non-commercial stops), and the sets of arcs A and edges E represent mono-directional and bidirectional tracks between pairs of stops, respectively. For each station $i \in S$, the capacity indicated with cap_i is known. For each pair of stations $i, j \in S$, the number of arcs (mono-directional tracks), α_{ij}, between station i and station j, the number of arcs (mono-directional tracks), α_{ji}, between station j and station i, and the number of edges (bidirectional tracks), β_{ij}, between the two stations i and j are known.

4.2.2 Trains, paths and routes

The set of trains is indicated by $T = T_P \cup T_F$, where T_P represents the set of passenger trains and T_F the set of freight trains. For each train $t \in T$, let cat_t be its category (e.g. "passenger – regional", "passenger – intercity", "freight", and so on), $f_t \in S$ the first station, $l_t \in S$ the last station, and $S_t \subseteq S$ the set of stations to stop at. Notice that for each freight train $t \in T_F$, $S_t = \{f_t, l_t\}$.

Let L be the set of all families/lines of trains bonded by a periodicity constraint, i.e. $L = \{L_1, L_2, \dots, L_{|T|}\}$, where $L_j \subseteq T$, $j = 1, \dots, |L|$, and $L_j \cap L_k = \emptyset$, $1 \le j, k \le |L|$, $j \neq k$. For each family L_j, $j = 1, \dots, |L_j|$, an ideal period time per_j, a minimum period time $\underline{per_j}$, and a maximum period time $\overline{per}_j$ are given, representing the ideal, minimum, and maximum time between the stops at each station of two consecutive trains in the same family. We assume that the trains of a given family $L_j = \{t_1, t_2, \dots, t_{|T_j|}\}$, $j = 1, \dots, |L|$, are ordered in increasing order of departure times. Moreover, we assume that, for each train, either the path (i.e., the sequence of traversed tracks with the corresponding travel times) or the route (i.e., the sequence of traversed tracks without the corresponding travel times) is provided.

4.2.3 Headway times

For each pair of trains t_1, $t_2 \in T$ and each station $s \in S$, the nominal headway time $h^{dd}_{t_1 t_2 s}$ ($h^{da}_{t_1 t_2 s}$) between the departure of train t_1 from station s and the departure (arrival) of train t_2 from (at) station s, and the nominal headway time $h^{ad}_{t_1 t_2 s}$ ($h^{aa}_{t_1 t_2 s}$) between the arrival of train t_1 at station s and the departure (arrival) of train t_2 from (at) station s are given. Any headway time is equal to 0 whenever the two trains do not meet at a station.

4.2.4 Running and dwell times

For each train $t \in T$ and each track either a mono-directional one $a = (s_1, s_2)$ or a bidirectional one $e = \{s_1, s_2\}$, the nominal running time $\underline{r}_{ta}$ and $\underline{r}_{te}$, respectively, and the maximum running time $\overline{r}_{ta}$ and $\overline{r}_{te}$, respectively, are given.

For each train $t \in T$ and each station $s \in S_t$, it is given the nominal dwell time $\underline{w}_{ts}$ and the maximum dwell time $\overline{w}_{ts}$.

For each train $t \in T$, it is also given the maximum running time $\overline{R}_t$ and the maximum dwell time $\overline{W}_t$.

4.2.5 Connections

Let Q be the set of all connections between pairs of trains at given stations. Each connection $q = (t_1, t_2, s)$, where t_1, $t_2 \in T$, $s \in S$, is characterized by a nominal and a maximum time, $\underline{u}_q$ and $\overline{u}_q$, respectively, to fulfil the connection constraint.

4.3 Output data

The macroscopic timetable computation provides the other modules with a macroscopic timetable, which is feasible from a macroscopic point of view, consisting, for each train, of a path or the indication that the train has been cancelled. A path of a train is an ordered sequence of tracks and provides, for each of the traversed tracks, the times where the trains enters and leaves the track.

4.4 Algorithm

The algorithm developed for the macroscopic timetable computation is based on the following integer linear programming model having six different types of variables.

Let P_t be set of all feasible paths for train $t \in T$, where a path is feasible if it visits all stations S_t and does not exceed the maximum running and dwell times, $\overline{R}_t$ and $\overline{W}_t$, for train t. For a given path $p \in P_t$ and a given station $s \in S_t$, let τ_{sp}^D and τ_{sp}^A be the departure and arrival time of train $t \in T$ at station $s \in S_t$ in the path $p \in P_t$.

Moreover, let us define the following penalties:

- π_t^{canc}: penalty paid for cancelling train $t \in T$;
- π_t^{runn}: penalty for each time unit of running time exceeding the nominal one for train $t \in T$;
- π_t^{dwell}: penalty for each time unit of dwell time exceeding the nominal one for train $t \in T$;
- π_q^{conn}: penalty for the connection time exceeding $\underline{u}_q$ for connection $q \in Q$;
- $\overline{\pi}_q^{conn}$: penalty for missing connection $q \in Q$;
- π_j^{per}: penalty for violating the ideal period time between two train of family $L_j \in L$;
- $\overline{\pi}_j^{per}$: penalty for violating the periodicity constraint of a train of family $L_j \in L$.

The cost c_p of path $p \in P_t$ is given by the running and dwell time exceeding the nominal ones penalized according to penalties $\pi_t^{runn}$and π_t^{dwell}, respectively.

Let K be the set of all path cliques, where each path clique is a set of paths in conflict (i.e., at most one of them can be part of a feasible timetable) because of violated headway times or violated station capacity.

Given the following sets of variables:

- Binary variable x_p equal to 1 if path $p \in P_t$ of train $t \in T$ is selected (0 otherwise),
- Binary variable ξ_t equal to 1 if train $t \in T$ is cancelled (0 otherwise),
- Integer variable y_q representing the connection time exceeding $\underline{u}_q$ for connection $q \in Q$ if connection $q \in Q$ is not missed,
- Binary variable χ_q equal to 1 if connection $q \in Q$ is missed (0 otherwise),
- Integer variable $z_{t_1 t_2 s}^+$ representing the difference (if positive) between the departure times of trains $t_1, t_2 \in L_j$ at station s and per_j,

- Integer variable $z^{-}_{t_1 t_2 s}$ representing the difference (if positive) between per_j and the departure times of trains $t_1, t_2 \in L_j$ at station s,
- Binary variable $\zeta_{t_1 t_2}$ equal to 1 if at least one of the two trains $t_1, t_2 \in L_j$, is cancelled and, therefore, the periodic service for family L_j is missing,

The algorithm developed in the macroscopic timetable computation is based on the following formulation of the problem:

$$\begin{aligned}
min \quad & \sum_{t\in T}\sum_{p\in P_t} c_p x_p + \sum_{t\in T} \pi_t^{canc}\xi_t + \sum_{q\in Q}\pi_q^{conn} y_q + \sum_{q\in Q}\overline{\pi}_q^{conn}\chi_q \\
& + \sum_{L_j\in L}\sum_{t_1\in L_j}\sum_{t_2\in L_j}\sum_{s\in S_{t_1}} \pi_j^{per}\left(z^{+}_{t_1 t_2 s} + z^{-}_{t_1 t_2 s}\right) && (1)\\
& + \sum_{L_j\in L}\sum_{t_1\in L_j}\sum_{t_2\in L_j} \overline{\pi}_j^{per}\zeta_{t_1 t_2}
\end{aligned}$$

$$s.t. \quad \xi_t + \sum_{p\in P_t} x_p = 1 \qquad t\in T \qquad (2)$$

$$\xi_{t_1} + \xi_{t_2} \le 2\chi_q \qquad q=(t_1,t_2,s)\in Q \qquad (3)$$

$$\sum_{p\in P_{t_2}} \tau_{sp}^{D} x_p - \sum_{p\in P_{t_1}} \tau_{sp}^{A} x_p \ge \underline{u}_q - M\chi_q \qquad q=(t_1,t_2,s)\in Q \qquad (4)$$

$$\sum_{p\in P_{t_2}} \tau_{sp}^{D} x_p - \sum_{p\in P_{t_1}} \tau_{sp}^{A} x_p \le \underline{u}_q + y_q - M\chi_q \qquad q=(t_1,t_2,s)\in Q \qquad (5)$$

$$\sum_{p\in K_k} x_p \le 1 \qquad k=1,\dots,|K| \qquad (6)$$

$$\xi_{t_1} + \xi_{t_2} \le 2\zeta_{t_1 t_2} \qquad L_j\in L, t_1,t_2\in L_j : t_1<t_2 \qquad (7)$$

$$\sum_{p\in P_{t_2}} \tau_{sp}^{D} x_p - \sum_{p\in P_{t_1}} \tau_{sp}^{D} x_p \le (t_2-t_1)per_j + z^{+}_{t_1 t_2 s} + M\zeta_{t_1 t_2} \qquad L_j\in L, t_1,t_2\in L_j : t_1<t_2, s\in S_{t_1} \qquad (8)$$

$$\sum_{p\in P_{t_2}} \tau_{sp}^{D} x_p - \sum_{p\in P_{t_1}} \tau_{sp}^{D} x_p \le (t_2-t_1)per_j - z^{-}_{t_1 t_2 s} - M\zeta_{t_1 t_2} \qquad L_j\in L, t_1,t_2\in L_j : t_1<t_2, s\in S_{t_1} \qquad (9)$$

$$z^{+}_{t_1 t_2 s} \le \overline{per}_j - per_j \qquad L_j\in L, t_1,t_2\in L_j : t_1<t_2, s\in S_{t_1} \qquad (10)$$

$$z^{-}_{t_1 t_2 s} \le per_j - \underline{per_j} \qquad L_j\in L, t_1,t_2\in L_j : t_1<t_2, s\in S_{t_1} \qquad (11)$$

$$x_p \in \{0,1\} \qquad p\in P_t, t\in T \qquad (12)$$

$$\xi_t \in \{0,1\} \qquad t\in T \qquad (13)$$

$$y_q \in \{0, \overline{u}_q - \underline{u}_q\} \qquad q\in Q \qquad (14)$$

$$\chi_q \in \{0,1\} \qquad q\in Q \qquad (15)$$

$$z^{+}_{t_1 t_2 s}, z^{-}_{t_1 t_2 s} \in \mathbb{Z}_+ \qquad L_j\in L, t_1,t_2\in L_j : t_1<t_2, s\in S_{t_1} \qquad (16)$$

$$\zeta_{t_1 t_2} \in \{0,1\} \qquad L_j\in L, t_1,t_2\in L_j : t_1<t_2 \qquad (17)$$

Where M is a large enough number.

The objective function (1) guarantees that the timetable achieved minimizes the total cost, given by the sum of: (a) the cost of the paths selected; (b) the cost for cancelling trains; (c) the cost for exceeding the nominal connection time; (d) the cost for missing connections; (e) the cost for not respecting the ideal period time; and (f) the cost for missing a periodic constraint. Constraints (2) impose on the model that each train is either cancelled or scheduled. Constraints (3) ensure that if both trains corre-

sponding to a connection are cancelled, then a penalty for missing the connection is paid. Constraints (4) state that if both trains of a connection are scheduled, then the difference between the corresponding departure and arrival times at the station where the connection takes place must be not less than $\underline{u}_q$. Constraints (5) trigger the penalty for exceeding the nominal connection time for each connection having both train scheduled. Constraints (6) are clique constraints that impose on the provided timetable to be conflict-free (i.e., no headway and capacity constraints are violated); notice that constraints (6) can also be used to model simultaneous arrival or departures of pairs of trains at given stations, if these have to be considered as hard constraints. Constraints (7) trigger penalties for missing periodic constraints. Constraints (8) and (9) ensure that penalties are paid if the ideal period time is not satisfied. Constraints (10) and (11) impose upper bounds on variables z^+ and z^-. Constraints $(12)-(17)$ set the domains of the variables of the model.

The macroscopic timetable computation uses an algorithm that computes a heuristic solution of model $(1)-(17)$. Such a solution is achieved by running a randomized multi-start greedy heuristic that performs a given number of macro-iterations, each one providing a conflict-free macroscopic timetable. At each macro-iteration, all lines/trains are processed, one at a time, and are inserted into the incumbent macroscopic timetable. The insertion of a train/line into the incumbent timetable corresponds to fixing a variable into the model $(1)-(17)$ and is performed by running an exact dynamic programming recursion that identifies a feasible min-cost path that is compatible with the paths assigned to the previously processed trains/lines. Such a dynamic programming recursion computes functions $f(t,d,e)$ with three state-variables, where t is the instant of time at which the event e (be it an arrival, a departure, or a pass at a given station along the route of the train) takes places, and d represents the total stretch of the path (computed as the exceeding running and dwell times over the nominal ones).

Among all the generated timetables, the macroscopic timetable computation chooses the most robust one. The selection is performed by generating a certain number (e.g., a few hundreds) of different instances, each one derived by simulating a random delay for each of the trains, and then re-optimizing all such instances. The re-optimization phase is aimed at assessing the "capacity" of the timetable of absorbing possible delays and includes only retiming of the trains (i.e., increasing or decreasing the scheduled dwell and running times without affecting the macroscopic feasibility) but with no rerouting nor reordering. The re-optimization computes a cost for each of the instances which is proportional to the time for absorbing the delays, and then computes a *robust* cost of the original timetable obtained as a convex combination of the costs of the considered instances. The timetable with the lowest robust cost is provided as the result of the macroscopic timetable computation.

The macroscopic timetable computation uses an algorithm that computes a heuristic solution of model $(1)-(17)$. Such a solution is achieved by running a three-phase algorithm: in the first phase, a lower bound based on model $(1)-(17)$ is computed through column generation; in the second phase, a *randomized multi-start greedy*

heuristic generates a set of feasible timetables by taking advantage of the dual information achieved in the first phase; in the third phase, the set of feasible timetables generated at Phase 2 are assessed in terms of stability and robustness, and the best one is provided as output. The output of the algorithm is a conflict-free macroscopic timetable that is both stable and robust.

The macroscopic timetable computation algorithm may be run several times in a loop that exchanges information with the microscopic module in order to guarantee that the final macroscopic timetable is also feasible from a microscopic point of view.

5 CORRIDOR FINE-TUNING

This chapter details the corridor fine-tuning module which consists of computing energy-efficient train speed profiles and adjusted arrival and departure times on corridors between two main stations. This section considers successively the objectives addressed, the input data, the output data, and the algorithm.

5.1 Objective

The objectives of the corridor fine-tuning are connected to the objectives defined in Deliverable 2.1. As the travel time, connectivity, capacity usage, stability and robustness objectives are already optimized within the iterative microscopic-macroscopic timetable computation, this module focuses on the minimization of energy consumption.

Besides the optimization of the planned driving trajectory according to energy efficient driving strategies, the corridor fine-tuning focuses on the optimization of the planned running times between important networks points. Consequently, the journey times and the capacity usage parameters are not influenced. However, due to the fact that especially time optimal driving consumes high energy (Figure 12), planned running times which minimize the probability of time optimal driving can improve railway operations.

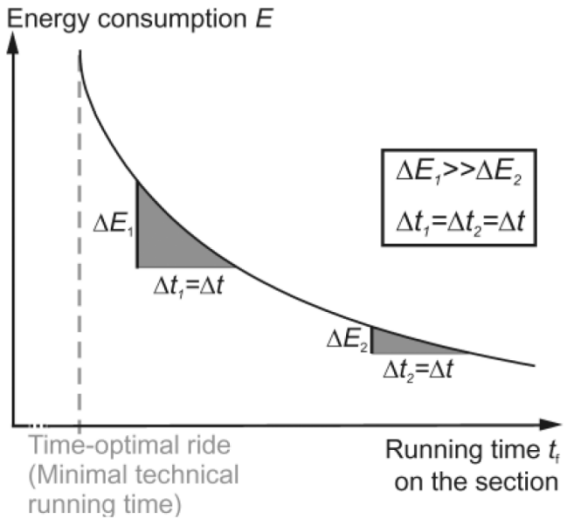

Figure 12. Characteristic dependency of energy consumption on running time

By using this described approach of corridor fine-tuning especially schedules for regional trains can be found, which optimize the running times between intermediate station in order to minimize the consumed traction energy while respect the stopping and passing point at important network points given by the macroscopic timetabling process.

5.2 Necessary definitions within the fine tuning module

The corridor fine-tuning module requires the definition of restrictions that influence the optimization. However, these criteria might be constraints given by the operator or infrastructure manager in order to avoid penalties or customer dissatisfaction.

5.2.1 Hard criteria

- ϑ - maximum delay at target station

  The probability of delays at the measurement points or important network points can never be reduced to 0. There is always the probability that large dwell times at consecutive stops might occur which exceed the planned dwell times and leads to delays within the network. Consequently, a maximal delay at the target station must be defined which should not be exceeded with a certain probability ψ. This hard criterion should be determined very sensitively, as it influence the limits of the optimization and the achieved results.

- ψ - tolerated quantile at target station

  Additional to ϑ, the maximal quantile of punctual trains at the target station ψ has to be defined. It symbolizes the amount of trains, which must arrive within the maximum tolerated delay. Similar to the definition of ϑ, it cannot be achieved that 100% of the trains arrive punctual. This hard criterion should be determined very sensitively, as it influence the limits of the optimization and the achieved results.

- δ - maximum delay at intermediate station

  The maximum delay at intermediate stations is the tolerated timetable deviation at minor important stops. The amount of tolerated delay limits the flexibility of the optimization. It should represent the tolerance of the passenger towards timetable deviations.

- No acceptance of early departure

  As the public timetable is the relevant information given to the passenger, no early departures should be allowed in order to avoid passenger dissatisfaction.

5.2.2 Weighting factors

The weighting factors balance between the considered criteria. Thereby the criteria

- Total expected energy consumption
- Expected target station delay
- Total expected intermediate delay

are respected. As the different criteria have different importance for the train operators, a weighting has to be considered. However, a solution method should find a compromise between the different objectives and should be stable towards the defined weighting factors.

The presented approach requires the definition if the weighting factors within different levels of the optimization. These levels are:

- within the running time optimisation and
- within the timetable optimisation.

The approach of running time optimisation is similar to real-time running time optimisation, such as done e.g. in WP 4. It represents the optimisation and running time modifications of the subsection running times in case of small timetable deviations. The algorithm of running time optimization which is used in this approach is influenced by the given timetable. Consequently different timetables enable different flexibilities towards the real-time operational control. Within the presented approach, the possibility of running time control within a given timetable is evaluated. In WP3 the focus is on this timetable evaluation. Hence, the optimal timetable can be found if the expected values of the different criteria for different timetables are measured.

The optimal timetable has to be found by the use of multi-objective optimization methods which have to be applied on the expected values for the different timetables. Therefore the definition of further weighting factors is necessary.

5.3 Input data

The fine-tuning module needs the timetable for the trains to be optimized computed at the previous optimization levels, dwell time distributions at all intermediate stations, and energy consumptions for different running times on each subsection between two consecutive stations. The latter can also be calculated within the fine-tuning module using algorithms of WP6 based on the RailML data. In detail, the input data consists of

1. A list of the train lines to be optimized. This list is given in an internal format with for each train line: The train line ID (first column), the successive stops/stations (second column), event type (third), scheduled arrival and departure times (4th and 5th column), the bandwidth given by the earliest arrival and the latest departure (6th and 7th), and an indicator for a macroscopic timetable event (8th column) indicating if the event times in the station/stop are fixed and therefore not subject to fine-tuning.
2. Dwell time distributions.
3. Energy consumption for different running times between each two consecutive stations or the (Infrastructure and Rolling Stock) RailML data to compute it.

The optimization will take place along corridors, i.e. between each two important stations (major hubs, terminal stations, stations with important connections or other operational constraints).

The input train line data represents the preliminary timetable for the train operations. It is the output of the preceding timetabling process and provided by the microscopic module. It contains the arrival and departure points at important network points for the train lines which should be optimised and also specifies the preliminary feasible ar-

rival and departure times at the intermediate stops that were computed by the microscopic module.

The bandwidths of the train path schedules limit the flexibility of the optimization process at intermediate network points. Due to headway conflicts with preceding or following trains, restrictions in the flexibility at intermediate stops can occur. At specific points in the network, especially intermediate stations, the earliest possible and latest allowed arrival and departure times have to be defined and transferred to the fine-tuning algorithm. These data are also calculated by the microscopic module.

The dwell time distributions are necessary input data to the stochastic dynamic timetable optimisation tool. As the uncertainty and variations included in the dwell times influence the running time optimization, the data should be available as accurate as possible and a distinction between the different intermediate stations should be made. These data could be obtained from historical measurement data, e.g. train door opening and closure times determined in the train or track occupation data filtered for undisturbed operations. Long dwell times due to red signal aspects or waiting for the published departure times have to be filtered from the dwell time data. TU Delft has developed algorithms based on track occupation data from which realized dwell time distributions or their fits can be obtained for Dutch corridors.

Figure 13 and 14 show examples of dwell time distributions of different stations within a German regional train network.

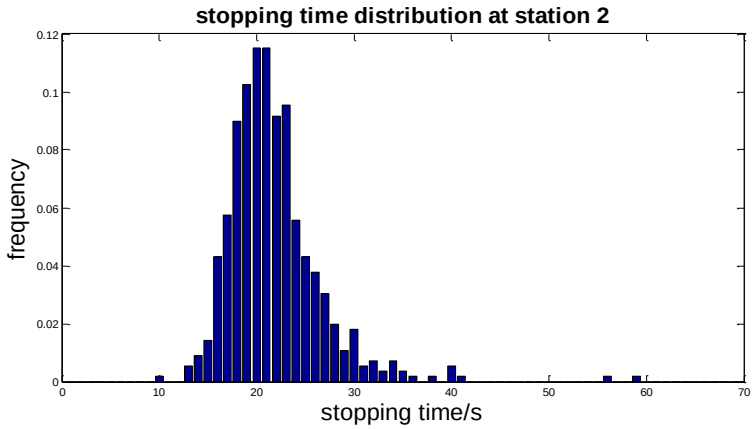

Figure 13. Exemplary dwell time distribution at intermediate station

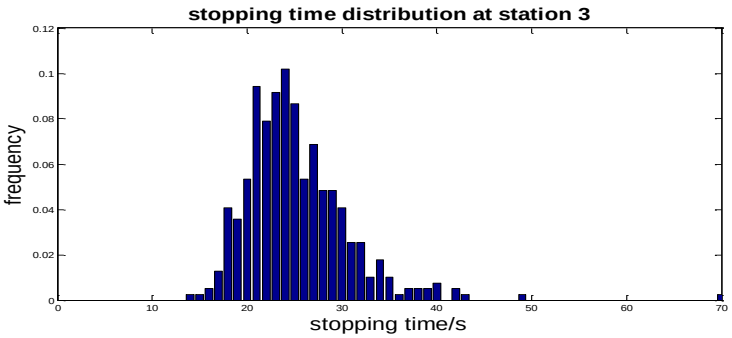

Figure 14. Exemplary dwell time distribution at intermediate station

Energy consumption for different running times between each two consecutive stations can be calculated from the following data:

a) Infrastructure data: The Infrastructure data must contain information about the track design including positions and IDs of stations, conflict points but also static speed information and gradients. A unique identification of each track element, its position, link to other track elements and characteristics must be available. The level of detail of the infrastructure data depends also on the approach of the energy consumption calculation.

b) Rolling Stock data and energy-consumption data: The level of detail of the infrastructure data and rolling stock data depends on the approach of the energy data input. It can be calculated within the fine tuning process using similar approaches for energy calculation as in WP 4/WP6. In this case the complete rolling stock data have to be provided. These data are used to determine the energy consumption for each possible running time on a section under the assumption that energy-efficient driving is applied. This assumption can be made within automatic train control systems and in case driver advisory systems for running time control are applied.

WP3 uses the same algorithms for energy consumption calculation as WP4/WP6 in order to ensure consistency within the project. Hence, energy data for each subsection are computed by offline calculations using the algorithm from WP4/WP6 which is implemented in Java and requires the RailML files as input.

5.4 Output data

The output of the fine-tuning module is the modified timetable for the local trains including the modified arrival and departure times at intermediate stations. This data is added to the RailML Timetable format by the microscopic module. Moreover, the RailML Timetable format is extended at this point to include information on the scheduled optimal speed profile. This is implemented as switching points for the energy-efficient driving regimes: FullPower, MaxSpeed, Coast, Brake, and Stop, with the scheduled time and speed at the switching points. For this the optimal energy-efficient speed profile for the local trains is recomputed based on the RailML data and the Timetable RailML updated accordingly.

```
<ns4:switchingPointsForRunningSection>
  <switchingPoint trackId="d15e30_Ut_Gdm22" positionInMOnTrack="6912.0" regime="FullPower" tractiveEffort="127500.0" brakingEffort="0.0"
    speedInKmPH="0.0" time="22:28:00" />
  <switchingPoint trackId="d15e30_Ut_Gdm22" positionInMOnTrack="6952.0" regime="FullPower" tractiveEffort="127500.0" brakingEffort="0.0"
    speedInKmPH="30.0" time="22:28:09" />
  <switchingPoint trackId="d15e2029_40" positionInMOnTrack="1973.0" regime="MaxSpeed" tractiveEffort="4847.0" brakingEffort="0.0"
    speedInKmPH="80.0" time="22:28:33" />
  <switchingPoint trackId="d15e2029_36" positionInMOnTrack="0.0" regime="MaxSpeed" tractiveEffort="6937.0" brakingEffort="0.0"
    speedInKmPH="80.0" time="22:34:13" />
  <switchingPoint trackId="d15e3007_1" positionInMOnTrack="212.0" regime="Coast" tractiveEffort="0.0" brakingEffort="0.0" speedInKmPH="80.0"
    time="22:34:22" />
  <switchingPoint trackId="d15e3007_1" positionInMOnTrack="386.0" regime="Brake" tractiveEffort="0.0" brakingEffort="-268620.0"
    speedInKmPH="79.0" time="22:34:30" />
  <switchingPoint trackId="d15e3007_1" positionInMOnTrack="496.0" regime="Brake" tractiveEffort="0.0" brakingEffort="-265216.0"
    speedInKmPH="30.0" time="22:34:38" />
  <switchingPoint trackId="d15e3007_1" positionInMOnTrack="515.0" regime="Stop" tractiveEffort="0.0" brakingEffort="0.0" speedInKmPH="0.0"
    time="22:34:42" />
</ns4:switchingPointsForRunningSection>
<ns4:switchingPointsForRunningSection>
  <switchingPoint trackId="d15e3007_1" positionInMOnTrack="515.0" regime="FullPower" tractiveEffort="127500.0" brakingEffort="0.0"
    speedInKmPH="0.0" time="22:35:00" />
  <switchingPoint trackId="d15e3007_1" positionInMOnTrack="555.0" regime="FullPower" tractiveEffort="127500.0" brakingEffort="0.0"
    speedInKmPH="30.0" time="22:35:09" />
  <switchingPoint trackId="d15e3007_3" positionInMOnTrack="342.0" regime="MaxSpeed" tractiveEffort="4847.0" brakingEffort="0.0"
    speedInKmPH="80.0" time="22:35:33" />
  <switchingPoint trackId="d15e3007_3" positionInMOnTrack="1327.0" regime="Coast" tractiveEffort="0.0" brakingEffort="0.0" speedInKmPH="80.0"
```

Figure 15. Timetable RailML extended with scheduled speed profile specification

The scheduled speed profile can be provided to the drivers so that is clear what is expected from the driver behaviour with respect to the running time supplements. This can be realized by giving e.g. the entire speed profile to the next station or just the switching time and target speed to the next driving regime. Note that one of the causes of running time variability between drivers is the different behaviour with respect to dealing with the running time supplement. By providing the scheduled energy-efficient driving regimes to the drivers they can adhere better to the schedule while using the least energy consumption. Of course, in case of delays the scheduled energy-efficient speed profiles are replaced by time-optimal running to recover from the delay as fast as possible. This solution can be used by trains that do not have a Driver Advisory System installed with dynamically computed speed advice, such as developed in WP6.

5.5 Algorithm

5.5.1 Definitions

The search space represents the volume of all potential solutions of the running time optimization problem. The definitions are illustrated in Figure 16. Here, a railway corridor with a total amount of I stations -each identified by i- will be considered. Station $i=I$ is defined as the target station and the section between station i and $i+1$ is indexed with j. The total number of sections between station 1 and I is given by $J=I-1$. A timetable consisting of planned arrival and departure times is evaluated and optimized.

The minimal and maximal running times on the sections and the minimal and maximal dwell times at each intermediate station are given. The minimal times are defined by technical limitations, the maximal times depend on definitions given by the operator, e.g. minimum allowed speed on the sections and maximum dwell time at the intermediate stations in undisturbed operation.

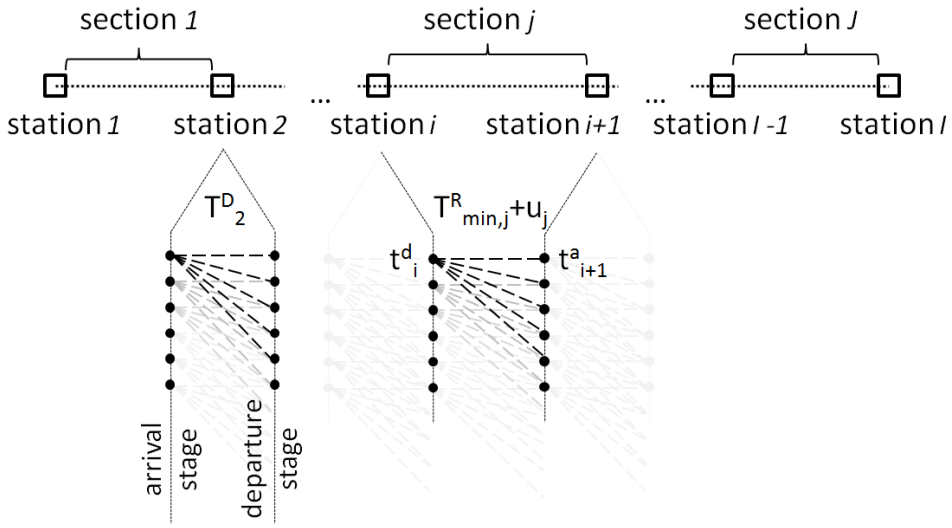

Figure 16. Definition of the corridor system model

The possible arrival and departure times at the intermediate stations have to be found within the optimization process. The total amount of allowance time which should be allocated among the section is defined by the timetable of the macroscopic timetable module.

Stage transition

In contrast to the network macroscopic module in which the dwell times are assumed to be constant, the stochasticity of the dwelling process is considered here. Each intermediate station stop is modelled with an arrival and departure stage.

The transition of the departure from a station to the following arrival at the next station is assumed to be a deterministic process. This assumption can be ensured by the use of driver advisory systems or in automatic train operation. Consequently, the arrival time depends on the minimal running time and the used allowance on this section. Because no departure before the planned departure time shall be allowed, the departure time does not only depend on the arrival time and dwell time, but also on the (to be optimized) scheduled departure time.

In the system model each dwelling process is regarded as independent event in undisturbed operations below the limit of capacity. However, the total planned allowance time might not be sufficient to cover all randomly occurring long dwell times along a corridor. There is still the probability that the trains will arrive with a delay at the target station. Hence, a maximum tolerated threshold at the target station ϑ and the minimum percentage of trains ψ which shall arrive with a maximum delay of ϑ have to be defined and limit the search space.

Limitation of the search space

The upper bounds for arrival and departure times at the intermediate stations are limited by the probability that the train will arrive at the target station within the tolerat-

ed threshold. If the probability is below ψ this arrival or departure time will not satisfy the constraints given by the operator and is therefore not a possible solution. Consequently, each tolerated arrival or departure time must fulfil the equation.

5.5.2 Quality criteria

For quantifying the quality of a timetable, three criteria $Q_q, q \in [1,2,3]$, are chosen which are considered important from the operators or passengers point of view. As these criteria are influenced by the stochasticity of the dwell times and the non-linear constraints of the given timetable, they depend on the distribution function and cannot be determined nominally, but their expected values at each arrival and departure stage have to be calculated with a recursive approach. Thereby the constraint that early departures are not permitted has to be ensured which leads to waiting within the station and consequently to the same criteria as the departure at the planned departure time.

The three quality criteria that are considered are the following:

- **Expected energy consumption**: The first quality criterion Q_1 is the amount of consumed energy depending on rolling stock and infrastructure characteristics of the line, but also on the available time allowance on each section which is assumed to be used for energy-efficient driving. The energy consumption for a given time allowance can be calculated for each section using algorithms described in the literature. The expected energy consumption for each arrival and departure stage can be determined, indicated as $Q_{1,i}^{Arr}(t)$ and $Q_{1,i}^{Dep}(t)$ for each station i.
- **Expected target station delay**: As described, delays at the target station cannot be avoided. However, an expected delay at the target station, which could be important from the operator's point of view, can be calculated for each departure and arrival time at all intermediate stations. These variables are indicated as $Q_{2,i}^{Arr}(t)$ and $Q_{2,i}^{Dep}(t)$ for each station i.
- **Expected delay at intermediate stations**: Although the intermediate stations are regarded as minor objective concerning the operational stability, occurring delays are relevant from the passenger point of view. Therefore the total amount of expected intermediate delay is the third optimization criteria, indicated as $Q_{3,i}^{Arr}(t)$ and $Q_{3,i}^{Dep}(t)$ for each station i.

The values of the quality criteria have to be calculated recursively backwards at each station *i* using the following equations [2]. In these equations, t_i^{Arr} and t_i^{Dep} denote the actual arrival and departure time at station I, $t_{TT,i}^{Arr}$ and $t_{TT,i}^{Dep}$ the timetable arrival and departure time at station *i*, $T_{min,i}^{D}$ and $T_{max,i}^{D}$ the minimum and maximum dwell time at station *i*, $T_{min,j}^{R}$ the minimum running time at section j, and u_j the allowance time on section j.

$$Q_{q,i}^{Dep}\left(t_i^{Dep}\right) = Q_{q,i}^{Dep}\left(t_{TT,i}^{Dep}\right), \qquad t_i^{Dep} < t_{TT,i}^{Dep},\ q = \{1,2,3\},$$

$$Q_{1,i}^{Dep}\left(t_i^{Dep}\right) = E_j\left(u_j\right) + Q_{1,i+1}^{Dep}\left(t_{i+1}^{Arr} = t_i^{Dep} + T_{min,j}^{R} + u_j\right), \quad t_i^{Dep} \geq t_{TT,i}^{Dep},$$

$$Q_{2,i}^{Dep}\left(t_i^{Dep}\right) = Q_{2,i+1}^{Dep}\left(t_{i+1}^{Arr} = t_i^{Dep} + T_{min,j}^{R} + u_j\right), \qquad t_i^{Dep} \geq t_{TT,i}^{Dep},$$

$$Q_{3,i}^{Dep}\left(t_i^{Dep}\right) = Q_{3,i+1}^{Dep}\left(t_{i+1}^{Arr} = t_i^{Dep} + T_{min,j}^{R} + u_j\right) \qquad t_i^{Dep} \geq t_{TT,i}^{Dep},$$

$$Q_{1,i}^{Arr}(t) = \int_{T_{min,i}^{D}}^{T_{maz,i}^{D}} f_i(\tau)\, Q_{1,i}^{dep}(t_i^{Arr} + \tau)d\tau,$$

$$Q_{2,i}^{Arr}(t) = \int_{T_{min,i}^{D}}^{T_{maz,i}^{D}} f_i(\tau)\, Q_{2,i}^{Dep}(t_i^{Arr} + \tau)d\tau$$

$$Q_{3,i}^{Arr}(t) = \max\left(0, t_i^{Arr} - t_{TT,i}^{Arr}\right) + \int_{T_{min,i}^{D}}^{T_{maz,i}^{D}} f_i(\tau)\, Q_{3,i}^{Dep}(t_i^{Arr} + \tau)d\tau,$$

using the terminal values at the target station:

$$Q_{1,I}^{Arr}(t_I^{Arr}) := 0,\ Q_{2,I}^{Arr}(t_I^{Arr}) := \max(0, t_I^{Arr} - t_{TT,I}^{Arr}),\ Q_{3,I}^{Arr}(t_I^{Arr}) := 0.$$

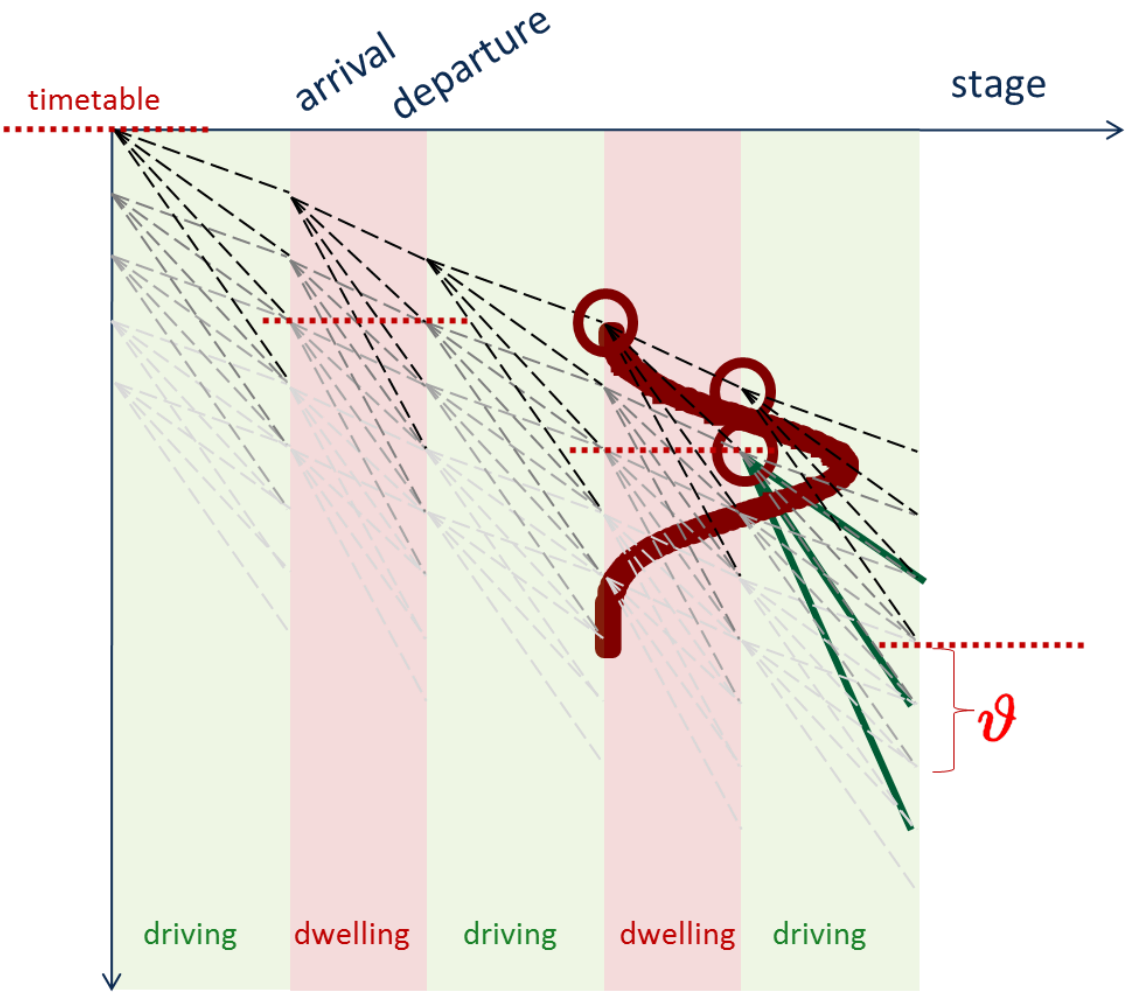

Figure 17. Illustration of the dynamic programming problem

5.5.3 Multi-criteria running time allowance allocation

All quality criteria depend on the allocation of the allowance time along the train run. Therefore they are modelled as Markovian, i.e., the value of a criterion at a certain state does not depend on the previous states, but on the present state and the sequence of decisions that follow. Because all quality criteria are Markovian, the multi-stage optimization problem can be solved by dynamic programming, see Figure 17. Thereby for each departure time a decision has to be made about the amount of allowance time to spend until the arrival stage at the next station. The decision is made based on the expected values of the criteria within a multi-criteria approach.

Here, the weighted metrics method is used to determine a substitute objective criterion, as this method is supposed to be less sensitive to the choice of weighting factors compared to other methods of multi-criteria decision making.

5.5.4 Timetable evaluation

The output of the allowance allocation process are the expected values of the relevant criteria for the remaining train run until the target station for each possible departure and arrival time at each station. Hence, the values obtained for the given departure time at the first station give an indication about the timetable itself, because the criteria are significantly influenced the timetable.

In order to find the optimal timetable, multiple timetables φ_k can be analysed. In a first approach, complete enumeration is used to find all potential combinations of arrival and departure times with respect to the minimal running and dwell times and the constraint. After the stochastic running time allocation process the optimization criteria can be used within a further multi-criteria optimization approach to determine the best timetable. The applied weighted metrics method requires the definition of weighting factors for each timetable evaluation criterion:

$$\boldsymbol{\varphi}^* := \min_{\boldsymbol{\varphi}_k} \sqrt{\sum_q v_q (Q_{q,1}^{Dep}(\varphi_k) - \min_{\varphi_k} Q_{q,1}^{Dep}(\varphi_k))}.$$

6 CONCLUSIONS

This document gave a description of an approach to compute high-quality timetables, with the following highlights:

- Microscopic calculation of running and blocking times taking into account all running route details at section level (gradients, speed restrictions, signalling),
- Microscopic conflict detection guaranteeing a conflict-free timetable
- Timetable precision of 10 s, minimizing capacity waste and unrealizable running times
- Incorporation of (UIC) infrastructure occupation and stability norms
- Macroscopic network optimization with respect to travel times, transfer times, cancelled train path requests and associated cancelled connections
- Macroscopic robustness analysis using stochastic simulation to obtain the most robust network timetable
- Macroscopic stochastic optimization of timetables for local trains on corridors between the main stations, taking into account stochastic dwell times at short stops on the corridor
- Energy-efficient speed profiles computed and incorporated for all trains
- Input from standardized RailML files (Infrastructure, Rolling Stock, Interlocking, Timetable)
- Output provided in standardized RailML Timetable file with scheduled train paths at (track-free detection) section level, extended with energy-efficient speed profile information.

The Functional Design document [7] defined four timetabling design process levels with increasing performance with respect to dealing with delays and disturbances. The ON-TIME timetabling design approach corresponds to the highest two levels. The successive levels are as follows:

Level 0 (Low quality). No conflict detection nor stability analysis are incorporated in the design process leading to poor timetables unless traffic is very low,

Level 1 (Stable). Stability analysis is an integrated part in the timetabling process so that the time allowances are proven to have good delay absorption behaviour for minor perturbations,

Level 2 (Feasible). Both conflict detection and stability analysis are incorporated in the design process resulting in proven conflict-free and stable timetables so that trains will run undisturbed in normal traffic

Level 3 (Robust). Robustness analysis is incorporated in the design process on top of conflict detection and stability analysis, resulting in proved robust timetables that can cope with normal statistical variations in operations

Level 4 (Resilient). Rescheduling measures are taken into account in the timetabling design process resulting in flexible timetables that enable efficient rescheduling in a late (ad-hoc) stage of the timetabling process or in real-time traffic management to deal with larger perturbations.

Note that the timetabling design levels are cumulative. For example, top level 4 is only achieved if the IM supports level 3 working with methods to prove stability, feasibility, and robustness and on top of this anticipates on ad-hoc train path requests or traffic management measures in the timetabling process.

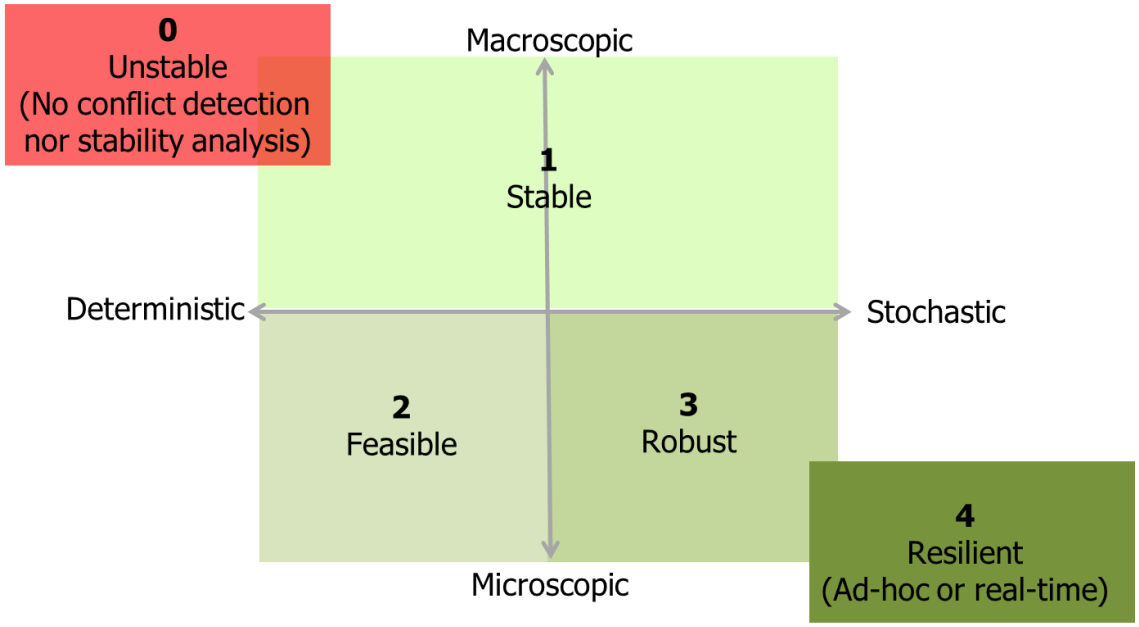

Figure 18. Timetable design levels and their requirements on tool functionalities

The subsequent timetabling design levels require timetabling models with increasing functionalities. Figure 18 projects the design levels on two axes for macroscopic versus microscopic models and deterministic versus stochastic models. To prove stability macroscopic algorithms can be used, while proving feasibility (conflict-free train paths) requires microscopic models based on e.g. blocking times. To prove robustness stochastic models are required additionally. Resilience requires knowledge of the traffic control measures, or in the ad-hoc timetabling phase sufficient spare capacity to insert additional train paths in an existing timetable.

The developed ON-TIME timetabling approach provides a timetabling design Level 3. Feasibility and stability is guaranteed by microscopic blocking time models, which feed a macroscopic model that optimizes the timetable at network level taking also into account robustness via a stochastic delay propagation and re-optimization model. Furthermore, energy-efficiency and robustness at a corridor level is taking into account using another stochastic model. This represents a sustainability dimension on top of the performance with respect to delays and disruptions.

Resilience timetabling is taken into account only explicitly with respect to scheduling freight paths. The timetabling algorithms allow inserting additional (freight) train paths whilst sufficient residual capacity has been reserved to guarantee that a stable conflict-free timetable can be found. In this case the passenger timetable might be adjusted a bit depending on the freight path requests and the maximum speed which are

detailed in a multi-speed freight catalogue. The performance of this will be tested in a case-study and reported in deliverable D3.2.

Resilience with respect to real-time traffic management has not been explicitly tested, but the stability and robustness design properties of the timetabling approach will have a positive effect on traffic controllability. This kind of resilience requires predictable traffic control decisions according to strict rules or methods that can then be incorporated in the timetabling design process. This requires a high level of automation in real-time traffic management which is currently not yet practiced. This might explain that at the moment resilience timetabling is not applied by any railway. The developed real-time traffic management algorithms in WP4 could however be used for this purpose as well, but this is a new line of research.

7 GLOSSARY

7.1 Variables of the macroscopic model

S	Set of commercial and non-commercial stops
cap_i	Capacity of stop $i \in S$
α_{ij}	Number of monodirectional tracks between stop i and stop j
β_{ij}	Number of bidirectional tracks between stop i and stop j
A	Set of monodirectional tracks
E	Set of bidirectional tracks
T	Set of trains
T_P	Set of passenger trains
T_F	Set of freight trains
cat_t	Category of train $t \in T$
f_t	First station of train $t \in T$
l_t	Last station of train $t \in T$
S_t	Set of station train $t \in T$ must stop at
L	Set of lines
per_j	Ideal period time for line L_j
$\underline{per_j}$	Minimum period time for line L_j
$\overline{per}_j$	Maximum period time for line L_j
$h^{dd}_{t_1t_2s}$	Nominal headway time between the departure of train t_1 from station s and the departure of train t_2 from station s
$h^{da}_{t_1t_2s}$	Nominal headway time between the departure of train t_1 from station s and the arrival of train t_2 at station s
$h^{ad}_{t_1t_2s}$	Nominal headway time between the arrival of train t_1 at station s and the departure of train t_2 from station s
$h^{aa}_{t_1t_2s}$	Nominal headway time between the arrival of train t_1 at station s and the arrival of train t_2 at station s
$\underline{r}_{ta}$	Nominal running time of train $t \in T$ along arc $a \in A$
$\underline{r}_{te}$	Nominal running time of train $t \in T$ along edge $e \in E$
$\overline{r}_{ta}$	Maximum running time of train $t \in T$ along arc $a \in A$
$\overline{r}_{te}$	Maximum running time of train $t \in T$ along edge e$\in E$

$\underline{w_{ts}}$	Nominal dwell time of train $t \in T$ at station $s \in S_t$
$\overline{w_{ts}}$	Maximum dwell time of train $t \in T$ at station $s \in S_t$
$\overline{R_t}$	Maximum running time of train $t \in T$
$\overline{W_t}$	Maximum dwell time of train $t \in T$
Q	Set of connections
$\underline{u_q}$	Nominal connection time for connection $q \in Q$
$\overline{u_q}$	Maximum connection time for connection $q \in Q$

7.2 Variables of the fine-tuning model

v_q	Weighting factor of criteria *q* within the running time optimization level
w_q	Weighting factors of criteria *q* within the arrival and departure time optimization level
φ_k	Combination of possible arrival and departure times
$Q_{q,i}^{Dep}$	Quality criteria *q* at the departure stage of station *i*
$Q_{q,i}^{Arr}$	Quality criteria *q* at the arrival stage of station *i*
t_i^{Arr}	Possible arrival time at station *i*
$t_{TT,i}^{Arr}$	Timetable arrival time at station *i*
t_i^{Dep}	Possible departure time at station *i*
$t_{TT,i}^{Dep}$	Timetable departure time at station *i*
$E_j(u)$	Energy consumption in subsection *j* in case the amount of *u* allowance time is used in this subsection
I	Target station
J	Last section just before target station
f_i	Dwell time distribution at station *i*
$T_{max,i}^{D}$	Maximum dwell time at station *i*
$T_{min,i}^{D}$	Minimum dwell time at station *i*
$T_{min,j}^{R}$	Minimum running time at section *j*
ϑ	Maximum tolerated threshold at target station
φ_k	Timetable variant *k*

REFERENCES

[1] Besinovic N., Quaglietta E., Goverde R.M.P. (2013). A simulation-based optimization approach for the calibration of dynamic train speed profiles. *Journal of Rail Transport Planning and Management*, in press, 2013.

[2] Binder A., Albrecht T. (2013). Timetable evaluation and optimization under consideration of the stochastical influence of the dwell times. In: Albrecht T., Jaekel B. (Eds), *Proceedings of the 3rd International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS 2013)*, Dresden, December 2-4, 2013.

[3] Butcher J.C. (2003). Numerical Methods for Ordinary Differential Equations. Wiley, London.

[4] Hansen I.A., Pachl J. (2008). *Railway Timetable and Traffic*. Eurailpress, Hamburg.

[5] ON-TIME (2012). *Deliverable D1.2: A framework for developing an objective function for evaluating work package solutions (Cost Function)*. Report ONT-WP01-D-UOB-025-01.

[6] ON-TIME (2013a). *T3.1: Assessment of State-of-Art of Train Timetabling*. Report ONT-WP03-I-EPF-008-03.

[7] ON-TIME (2013b). *T3.4: Functional design of Robust and Resilient Timetable models*. Report ONT-WP03-I-UDB-009-03.

[8] *UIC Code 406: Capacity*. International Union of Railways, Paris, 2nd Edition, June 2012.

[9] Van Egmond R.J. (2000). An algebraic approach for scheduling train movements. *The 8th International Conference on Computer-Aided Scheduling of Public Transport (CASPT 2000)*, Berlin, 21-23 June, 2000.