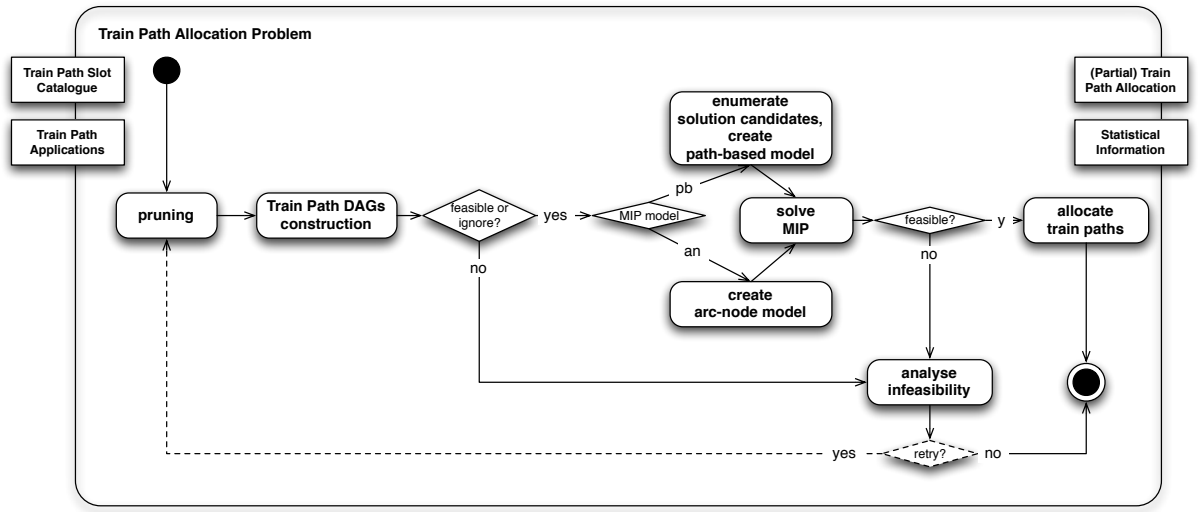


Algorithmic Decision Support for Train Path Allocation

Christian Eichenberger

Diploma Thesis DAS

Institute for Transport Planning and Systems **2016**

Institut für Verkehrsplanung und Transportsysteme
Institute for Transport Planning and Systems

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Contents

List of Figures

List of Tables

Diploma Thesis DAS

Algorithmic Decision Support for Train Path Allocation

Christian Eichenberger
Münstergasse 55
CH-3011 Bern

phone: +41 26 321 13 16

`https://github.com/tuabache/TrainPathAllocation`

2016

Abstract

Traditionally, train paths for freight trains are constructed by an **infrastructure manager** (IM) on a **make-to-order** basis. If train path slots are pre-fabricated in a **train path catalogue**, the available infrastructure can be exploited to higher degree since the train paths can be constructed in a **standardised** and **harmonised** way. This leads to an **assemble-to-order** process: **railway undertakings** (RUs) apply for train paths and infrastructure managers **allocate train paths** to the different possibly conflicting **applications**. The problem of allocating train paths to applications is a mathematical optimisation problem: for instance in a simple linear cost model, the sum of all trip directions and deviations from the requested departure and arrival times should be minimised. In this diploma thesis, a timetable planning workflow is suggested that shows the interface between domain experts and operations research experts, the train path allocation problem of periodical (typically weekly) allocation over a planning horizon is formalised in a mathematical model; a graph representation of the (pruned) solution space and two Mixed-Integer Programming (MIP) formulations are derived; and the algorithms are verified against synthetic and real-world data.

Keywords

Train Path Allocation, Train Path Catalogue, Train Path Application, Make-to-Order, Assemble-to-Order, MIP, Decision Support, Integer Linear Optimisation, Directed Acyclic Graphs, Column Generation, Pruning, Multicommodity Flow

Preferred citation style

Eichenberger, Ch. (2016) Algorithmic Decision Support for Train Path Allocation, **Diploma Thesis DAS, Institute for Transport Planning and Systems**, ETH Zurich, Zurich.

Diplomarbeit DAS

Algorithmic Decision Support for Train Path Allocation

Christian Eichenberger
Münstergasse 55
CH-3011 Bern

Tel: +41 26 321 13 16

`https://github.com/tuabache/TrainPathAllocation`

2016

Zusammenfassung

Herkömmlicherweise werden Trassen für Güterzüge durch eine **Infrastrukturbetreiberin** auf Bestellung hin konstruiert (make-to-order). Indem Trassen **standardisiert** und **harmonisiert vorfabriziert** werden und in einem **Trassenkatalog** angeboten werden, kann die vorhandene Infrastrukturkapazität potentiell besser ausgenützt und ausgewiesen werden: Trassen werden auf Bestellung durch ein **Eisenbahnverkehrsunternehmen** hin nur noch vergeben (alloziert) in einem Assemble-to-Order-Prozess. Die Trassenvergabe wird zu einem mathematischen Optimierungsproblem, in dem z.B. mit einem linearen Kostenmodell die vergebenen Trassen mit ingesamt minimaler Reisezeit und minimaler Abweichung von gewünschter Abfahrts- und Ankunftszeit gewählt werden sollen. In dieser Diplomarbeit wird ein Workflow für die Fahrplanplanung vorgeschlagen mit Schnittstellen zwischen Fachspezialisten der Fahrplanplanung und Operations-Research-Spezialisten; das Trassenvergabeproblem von periodischen (typischerweise wöchentlich) zu vergebenen Trassen wird mathematisch formuliert; eine Graphendarstellung des (beschnittenen) Lösungsraums und zwei ganzzahlige lineare Problemformulierungen werden abgeleitet; und die Algorithmen werden mit synthetischen und reellen Datensätzen validiert.

Schlüsselwörter

Trassenzuweisung, Trassenkatalog, Trassenantrag, MIP, make-to-order, assemble-to-order, ganzzahlige lineare Optimierung, gerichtete azyklische Graphen, Spaltenerzeugung, Pruning, Mehrgüterflussproblem

Bevorzugter Zitierstil

Eichenberger, Ch. (2016) Algorithmic Decision Support for Train Path Allocation, **Diplomarbeit DAS Institut für Verkehrsplanung und Transportsysteme**, ETH Zürich, Zürich.

1 Introduction

1.1 Motivation

Traditionally, train paths for freight trains are constructed upon individual requests on a make-to-order basis: a timetable planner introduces a train path into a timetable at different planning stages (from strategic to operational planning). In passenger transportation, the demand is very sensitive to connections and to regular intervals of services. In train freight transportation, the demand is usually end-to-end, not at a regular interval within a day, and connections to other trains or other means of transportation are typically only required at a non-variable time both at the departure and arrival locations. This situation is summarised by Opitz (2009) as follows:

> Im Vorfeld der Tassenvergabe ist es bereits möglich, Systemtrassen für den [Güterverkehr (im Original abgekürzt: GV)] zu gestalten. Dabei wird zunächst der vertaktete [Personenverkehr (im Original abgekürzt: PV)] für eine Strecke ermittelt und anschliessend die übrige Streckenkapazität mit Systemtrassen ausgefüllt, deren Fahrdynamik durch den typisch verkehrenden [Güterverkehrs]-Zug bestimmt ist. Die Belegung der Systemtrassen durch reelle Trassen erfolgt [in; im Original nach] der Trassenvergabe [...] Dabei muss davon ausgegangen werden, dass bei diesem Verfahren nur der [Personenverkehr (im Original abgekürzt: PV)] vertaktet verkehrt, da bei einer Vertaktung beider keine Bevorzugung des [Personenverkehrs] vor dem [Güterverkehr (im Original abgekürzt: GV)] stattfinden darf [...] Die wesentliche Eigenschaft der Systemtrasse liegt in der Tatsache, dass einzelne Trassen nicht für einen bestimmten Zug reserviert sind [...]

Such placeholders could be pre-fabricated in a systematic way and then allocated to a set of concurrent applications. This problem is called **train path allocation problem**. To the author's knowledge, there is no commercial tool nor published work on the train path allocation problem, and it can be assumed that the problem is tackled in practice by applying a path search heuristic application by application.

1.2 Scope and Limitations of this Thesis

This is a first approach to the train path allocation problem; the main focus of this thesis is

- to provide a mathematical formalisation of the train path allocation problem;

- to provide an implementation of a tool, which helps to solve train path allocation problems and to analyse and visualise the reason of infeasibility;
- to verify the algorithm against a real-world data set and to make a general statement on the effectiveness of the algorithm.

It is not the scope of this thesis

- to make a political or commercial statement on the interest of pre-fabricated train path slot catalogues;
- to provide a performance-tuned commercial product;
- to provide a business-ready monetary pricing model;
- to make a detailed mathematical complexity analysis of the solution algorithm;
- to be a user's nor programmer's guide to the implemented tool.

See also Section 6.2 for a more detailed list of further research.

1.3 Declaration of the Data Basis and of the Transparency of the Results

The topic of this thesis has been devised by SMA and Partners Ltd. (2015) and has also been supervised by them. Real-world data was provided by a major European Infrastructure Manager (IM) through SMA and Partners Ltd. (2015) under the following conditions:

- the IM is not named explicitly;
- no results are published that could allow to infer on the data provided;
- the results of this thesis are provided to them and may be used by them.

Therefore, the provided data was only used

- to develop the algorithms;
- to derive synthetic data sets of appropriate size and structure;
- for performance analysis.

The author has not been paid by the data providing unnamed IM nor by SMA and Partners Ltd. (2015) and has not provided any consulting to either of them apart from making this thesis and the source code available. The source code is licensed under the Apache 2.0 License and available under

`https://github.com/tuabache/TrainPathAllocation`.

1.4 Thesis Outline

The thesis has the following structure:
Background: in Section 2, the train path allocation problem is put into the context of the railway planning process and the political decisions which lead to the enforced separation of traditional integrated railway companies into infrastructure managers and railway undertakings; a workflow is suggested that integrates the train path allocation problem into the timetable planning process and existing organisations;
Modelling: in Section 3, the train path allocation problem is formulated as a mathematical problem;
MIP: in Section 4, the train path allocation problem is shown to be a multicommodity flow problem, and two mixed-integer problem (MIP) formulations are derived, which allow the problem to be solved by a commercial MIP solver;
Workflow: in Section 5, the activities involved in solving the train path allocation problem are presented, from the input data to different strategies of dealing with infeasibility;
Conclusion: in the final Section 6, the main results of the thesis and open issues for future research are discussed.

2 Background of the Train Path Allocation Problem

Section Outline

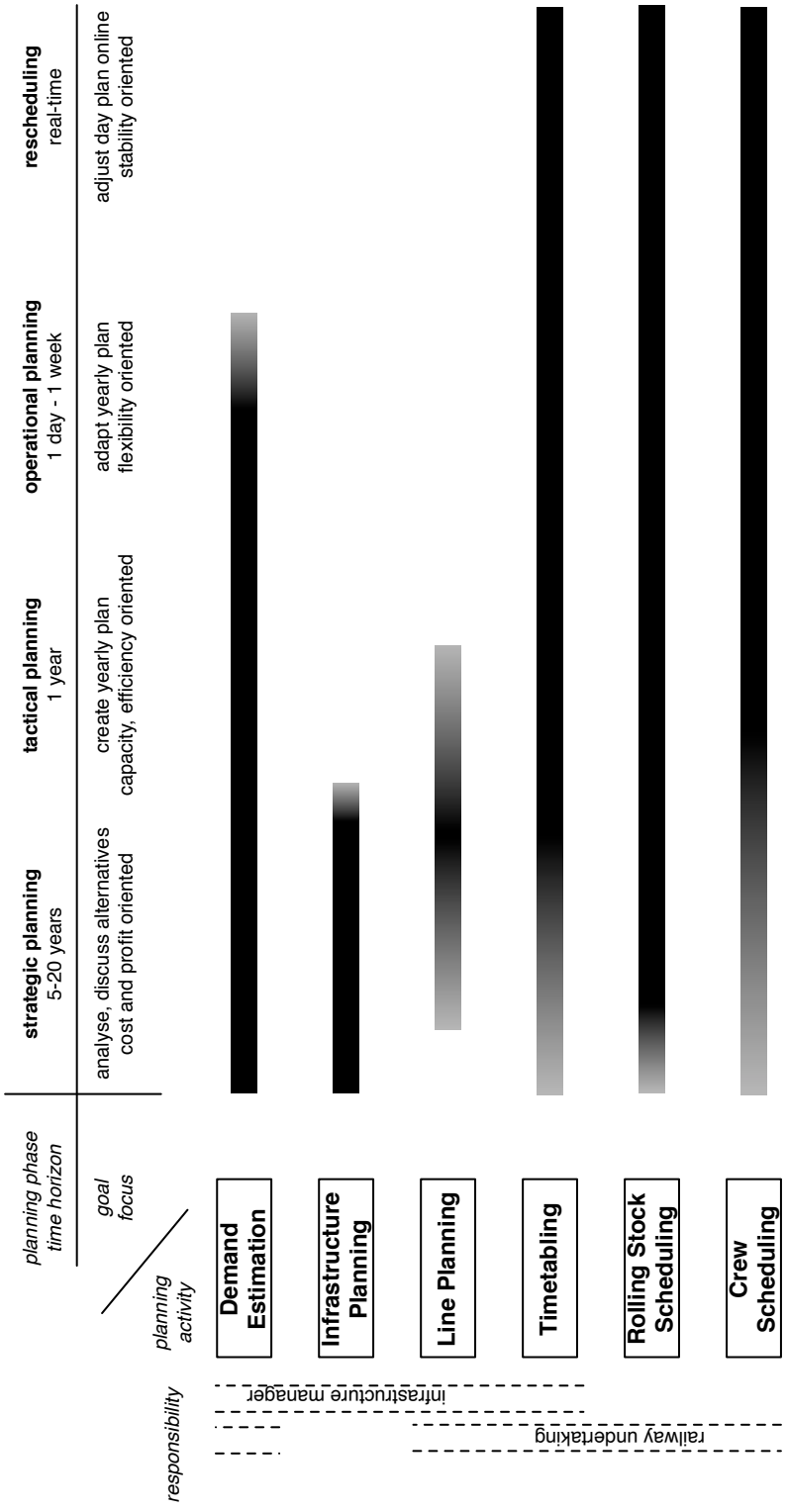
In this section, train path allocation will be motivated in an informal way from its political and organisational context and historical background.

2.1 The Train Path Allocation Problem in the Railway Planning Process

Caimi (2009) (following Bussieck et al. (1997)) identifies six activities in railway planning, as shown in Figure 1:

demand estimation "[Demand estimation is] an estimation of the number of people wishing to travel from an origin to a destination over a certain period of time during the day, or the amount of goods to be transported. This can be conducted with passenger counts, interviews

Figure 1: Activities and their Planning Horizon in the Railway Planning Process: "During strategic planning, all steps are considered and the infrastructure changes are decided. In tactical planning, the line plan is fixed for the yearly basic plan. The timetabling, rolling stock, and crew planning stages are taken into account continuously until the online operation." (Caimi, 2009)



Source: Adapted from (Caimi, 2009)

with current or potential customers, and through sales analysis. The results enable the creation of time-dependent origin-destination matrices [...] or of more aggregated data." (Caimi, 2009)

infrastructure planning "This is a crucial step in the planning procedure because of the long life and the very high investment costs of the railway infrastructure. [...] This [activity] is the responsibility of the infrastructure manager companies, in close collaboration with the public authorities. Ideally, decisions regarding the infrastructure should also be supported by generating a simplified timetable fulfilling the demand, and the infrastructure is then planned for supporting this generated timetable." (Caimi, 2009)

line planning Based on a demand estimation a passenger line plan or freight corridors are defined: "[r]equirements and interests of the different [RUs; in the source: train operating company] are combined together to form a so-called train service intention, which describes the intended train services that the [RUs] plan to offer to their customers. Therefore, the train service intention is a description of the commercial requirements that have to be fulfilled by the train schedule." (Caimi, 2009)

timetabling Based on a formalised or implicit service intention, train runs are scheduled from strategic planning to just-in-time rescheduling: "Once a line plan has been specified and a train service intention is formalised, a timetable has to be constructed which fulfils the requirements defined in the service intention and describes each individual train trip offered to the customers in detail. This is the last step that is published, whereas the next steps are only meant for company-internal planning and are not relevant for the customers." (Caimi, 2009)

rolling stock planning "It consists of defining a set of trips operated in sequence by the same rolling stock unit, usually on a cyclic basis, over a certain period. However, the same unit could then be assigned to a different set of trips in the next period [...]" (Caimi, 2009)

crew scheduling "It usually takes place in two stages: crew scheduling and crew rostering. Crew scheduling is the generation of a set of general work duties (also called shifts) covering all activities in the operations that require a certain type of personnel, such as train drivers or conductors. Crew rostering is then the assignment of personnel to the duties for a certain time horizon (weeks or months), taking into account holidays, crew member characteristics, (such as knowledge of the lines for train drivers) as well as laws and labor agreements. The rostering can be cyclic or non-cyclic." (Caimi, 2009)

Railway planning is a complex process: although these 6 activities form a hierarchy in the sense that each activity depends on the previous upper one, decisions in subordinate activities may trigger changes in superordinated activities, as discussed by Caimi (2009):

> The [activities] are however not independent of each other and cannot be considered in a purely sequential way. Everything is mutually influenced by everything else:

for instance, a timetable which is more attractive to customers may need additional rolling stock because of an inefficient vehicle turnaround. On the other hand, a slightly worse timetable in terms of customer attractiveness could improve the rolling stock circulation and therefore the cost-effectiveness of the railway offer. Introduction of feedback loops or the integration of several steps are possible ways to overcome these difficulties, but this increases the complexity of the problem that needs to be solved.

The introduction of train path allocation is supposed to introduce a narrow but general interface that may help to divide-and-conquer the complexity.

2.2 Political and Organisational Background of the Train Path Allocation Problem

Traditionally, railway companies were owner of the railway infrastructure and stock and provided train services on their infrastructure. Intending to increase competition and thereby, hopefully, efficiency, the European Union aims at allowing for non-discriminatory access to network infrastructures such as electricity, telecommunications, railway etc. by enforcing independence of infrastructure management and service companies. In the case of railways, the Council of the European Communities (1991, 2001) postulated the separation of

infrastructure manager (IM) "Owner and operator of infrastructure facilities for rail transport (public railway network)" (Federal Office of Transport FOT, 2016)

railway undertaking (RU) "Companies that operate passenger and/or freight transport on their own or third-party infrastructure." (Federal Office of Transport FOT, 2016)

both on the

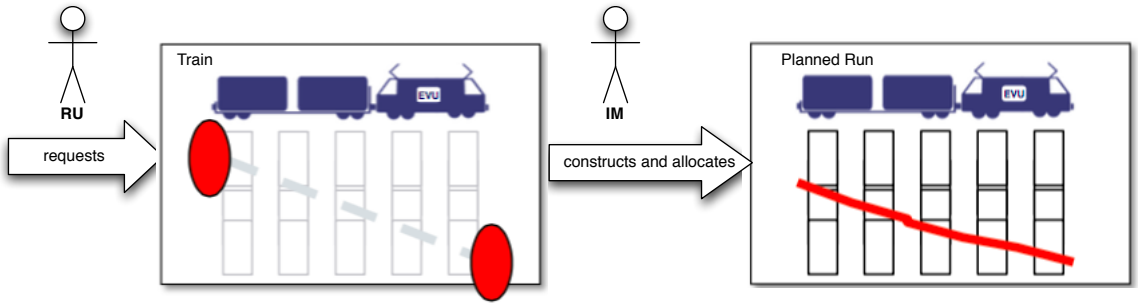
- organisational and
- financial (in particular, proscription of cross-financing)

level (Weidmann, 2011). In order to allow for non-discriminatory access to network infrastructure, key elements are[1]

train path slot "A train path (comparable to a slot for airlines) is the right to run a train defined in terms of its weight, length and loading gauge over a particular section of the rail network

[1] A "train path" may consist of one or more "consecutive segments." Therefore, in this thesis, it is essential to differentiate between a "train path" (reflecting the RU's view) and individual "train path slots" it consists of (reflecting the IM's view).

Figure 2: Make-to-Order Train Path Construction: Construction and Allocation of Train Paths upon Request.



Source: Adapted from (Weiß, 2015)

at a given time." (Trasse Schweiz AG, 2016)

train path slot catalogue "The inventory of pre-constructed paths for freight traffic [...]" (Trasse Schweiz AG, 2016)

train path application " "Train path application" means an application for a train path submitted by a railway undertaking [...]" (Trasse Schweiz AG, 2016)

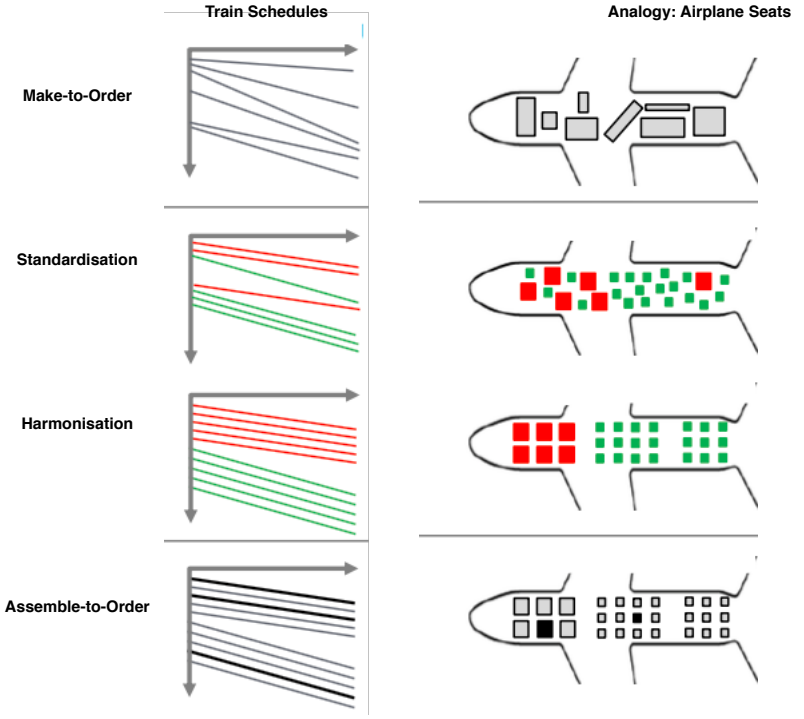
How this separation leads to the train path allocation problem, is shown in Figure 2: a RU traditionally applies for a train run giving

- time and place of departure and arrival,
- operational characteristics such as intermediate stops,
- train characteristics,
- a service level (tolerances),
- days of service;

then, the infrastructure manager constructs, offers, and finally allocates train paths for each individual application. This **make-to-order process** (Weiß, 2015) is characterised by the following features:

- manual construction of train paths for each individual request, expert knowledge is needed to cope with a variety of regulations;
- software only visualises train paths, enforces and tracks the ordering process and stores the data; however, there is no decision support in the sense of automatised plausibility checks or of train path construction;
- lack of transparency in the remaining capacity;
- lack of transparency in the optimality of infrastructure exploitation;

Figure 3: Process Industrialisation: from Make-to-Order to Assemble-to-Order



Source: Adapted from (Weiß, 2015)

- lack of standardisation and harmonisation of train paths, and thereby loss of capacity;
- due to the manual, custom-made construction and order process, orders cannot be made just in time;
- lack of differentiability of the product catalogue

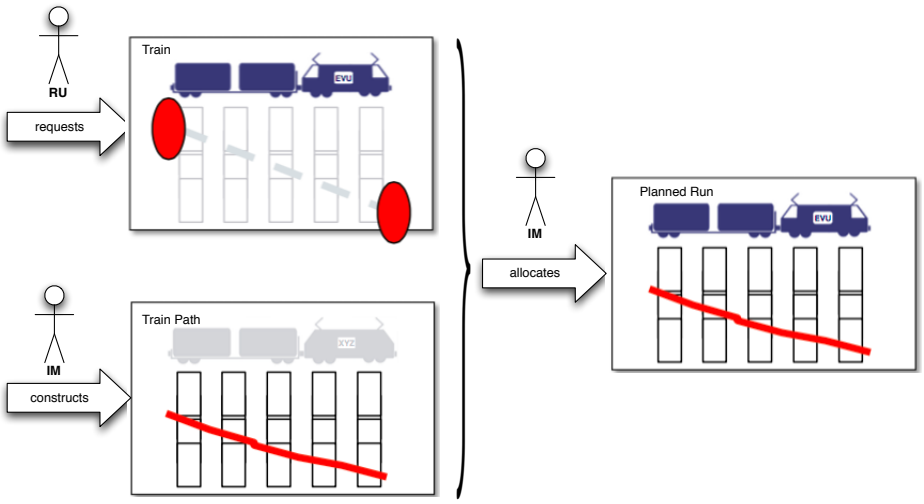
(list compiled from (Beck, 2015) and (Bucher, 2014; SBB, 2016, 2009)). The following two principles can help to achieve a better exploitation of the existing infrastructure (Weiß, 2015):

harmonisation The less driving times vary, the better the infrastructure capacity can be exploited; if the same infrastructure is used for different train types, capacity is best exploited if trains of similar driving times are grouped together.

standardisation In order to simplify harmonisation, train paths with the same characteristics can be constructed reflecting market share segments.

The application of these two principles is depicted in Figure 3: the demand is anticipated and the offer is constructed by standardising and harmonising the structure of the offer, reducing the original problem of free train path construction within the remaining capacity to an allocation problem from a well-defined train path catalogue.

Figure 4: Separation of Train Path Construction and Allocation in the Assemble-to-Order Process



Source: Adapted from (Weiß, 2015)

The pre-fabrication of train paths according to these principles leads leads from a make-to-order to an **assemble-to-order process** (Weiß, 2015):

make-to-order manual construction (and allocation) upon request;

assemble-to-order manual or automatised pre-fabrication of standardised and harmonised train path slots (collected in a train path slot catalogue) and allocation in batches or upon request.

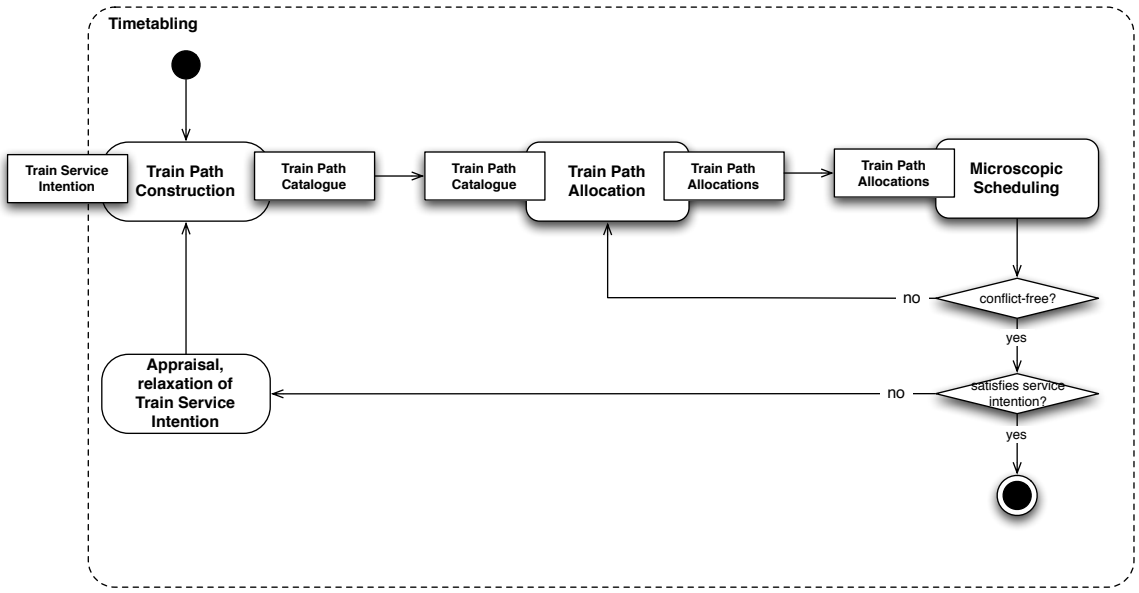
If train paths are pre-fabricated, they can be be allocated upon request from railway undertakings; this assemble-to-order Process is depicted in Figure 4

Therefore, at the political level, the European Union decided to institute such train path catalogues for freight trains on certain Corridors (European Parliament and Council of the European Communities, 2010) with a one-stop shop controlling non-discriminatory access to the corridor and coordinating capacity along the corridor.

2.3 The Train Path Allocation Problem in Timetabling

Since railway planning is a complex process as seen in Section 2.1, it is essential to define stable, but generic interfaces between different activities. One such interface in timetabling could be the construction of a train path catalogue, separating construction and allocation of train paths,

Figure 5: Timetabling Activities with Separation of Train Path Construction and Train Path Allocation (UML 2.0 Activity Diagram). Refer to Figure 1 for the context of timetabling within railway planning.



Source: Own representation, based on ideas from (Caimi, 2009)

as shown in Figure 5:

train path construction based on a service intention, a train path catalogue is constructed respecting

- safety: headway and non-collision constraints, i.e., train path slots are mutually non-conflicting by construction,
- connections and time dependencies (for passenger trains);

train path allocation from a train path catalogue, train path slots are allocated to train path applications;

microscopic scheduling if it is possible to find a feasible allocation at the macroscopic level, the solution has to be augmented by the microscopic details in the search of a conflict-free scheduling on this level, in particular in system nodes;

appraisal, relaxation of train service intention if the service intention cannot be satisfied, the service intention has to be re-defined: "Here, commercial considerations dominate technical ones. The importance of each service has to be analysed, e.g., from its economical point of view, its impact on other services, or its fairness towards all train operating companies. Based on this, a less restrictive [service intention] has to be generated and the train scheduling procedure started over again." (Caimi, 2009)

For freight trains, goods are transported from source to destination, and there are typically no connection and time dependencies; hence, in this case, the separation of construction and allocation of train paths seems promising.

For passenger trains, connections and time dependencies (and changes to them) are sensitive issues for a commercially attractive offer and hence play a major role in the construction of train paths; these additional constraints apply to ensure an integrated operating concept of a coordinated regular timetable (or clock-face timetable), where there is a common axis (or axes) of symmetry for all (or at least among the major) lines in the network and where transfers are scheduled: "Connections between trains also belong to a commercial description, as they permit the passenger to efficiently change trains to continue their trip. The minimum connection time depends on the infrastructure of the railway station, in particular on the distances passengers have to walk. Upper bounds are defined as the acceptable waiting times for the passengers. Time dependencies between train events are also an important commercial requirement, as they separate two different train runs in time that are covering the same demand, at least partially."(Caimi, 2009) Referring to Figure 5, the train path catalogue passed as input to train path allocation could only contain slots and applications for freight trains, whereas regular passenger services are directly allocated in the train path construction activity.

Timetabling typically works on a prototypic slice which is then repeated throughout the planning horizon, possibly with intermediate aggreations and modifications: "Timetable planners can work out problems on an elementary time slice, the basic framework, which equals the period. The basic framework is then repeated throughout the daily operational range (from, let us say, 6 a.m. to 10 p.m.)." (Tzieropoulos and Emery, 2009) For passenger trains, the time frame is typically one hour (or a whole day) and can be modelled by time dependencies between the different runs and modelled as a Periodic Event Scheduling Problem (PESP), for which various algorithms for macroscopic timetabling have been developed (Caimi, 2009; Herrigel-Wiedersheim, 2015). For freight trains, the time frame is typically one week, or, in a hierarchical approach, a prototype day for repeated services. These diffences are summarised in Table 1.

The focus of this thesis is on freight trains. Therefore, time dependencies and connection dependencies among different services will not be considered. Here, the following cases are distinguished:

non-periodical a train path applications asks for a single run, for instance from s to t departing at 14:00 on April 1st and arriving at 21:00 on the same day;

periodical simple a train path application asks for a periodical run, for instance from s to t on Mondays, departing at 14:00 and arriving at 21:00;

periodical with periodicity a train path application asks for a periodical run, repeated on

Table 1: Differences in Timetabling for Passenger and Freight Services. Different goals lead to different timetabling constraints and planning processes.

	passenger service	freight service
periodical	mostly	mostly, but demand for short-term
location of coordin.	at most intermediate stations	end-to-end
time coordination	minutes	hours
regular-interval	yes, differentiated e.g., peak/off-peak/evening	no (at most once)
days of service (periodicity)	same for groups of services e.g., working day timetable	indvidual
memorisability	important	unimportant
time frame	(hour $\rightsquigarrow$) day$\rightsquigarrow$ week $\rightsquigarrow$ planning horizon	(day$\rightsquigarrow$) week $\rightsquigarrow$ planning horizon

Source: Own representation

several operation days (the periodicity), for instance from s to t on Monday and Friday, departing at 14:00 and arriving at 21:00.

Notice that the non-periodical case can be reduced to the simple periodical case by restricting the planning horizon to one period. Furthermore, the simple periodical case is a special case of the repeated periodical case. In this thesis, the theory for the simple periodical case will be developed first, and then later a sketch will follow how it can be generalised to the repeated periodical case (see Appendix D).

Figure 6 shows a workflow for train path allocation with the following three activities:

subproblem choice choose a subset of applications to be allocated; this activity can be either a planner's activity in iterative timetabling, or, in an iterative solution process for the train path allocation problem, an activity performed by an operations research expert or by software (e.g., if a hierarchical decomposition is applied, where first all-week applications are allocated and one-day applications only later on);

train path allocation problem try to solve the train path allocation subproblem; this activity is mainly automatic and is the main topic of this thesis;

appraisal, relaxation remove or add or modify train path applications; again commercial considerations dominate technical ones; this activity typically involves human negotiations.

The suggested workflow has different mutually non-exclusive interpretations:

- *organisational*: solving the train path allocation problem is a black box to domain-experts, who specify a subproblem they wish to be decision-supported by operations research experts, and who get feedback on infeasibility;
- *decompositional*: iteratively choosing new subproblems helps to allocate train paths, e.g., all-week applications are dealt with first, then simple applications and finally non-periodical applications in a narrow planning horizon;
- *commercial*: unallocated train paths reflect the infrastructure managers remaining capacity and could be allocated to a train path application from a RU in a just-in-time manner.

The suggested workflow for train path allocation is an iterative, interactive and open (or online (Wikipedia, 2015)) process:

iterative the train path allocation problem may be infeasible or practically intractable, and the applications may have to be reconsidered or the train path catalogue may have to be modified;

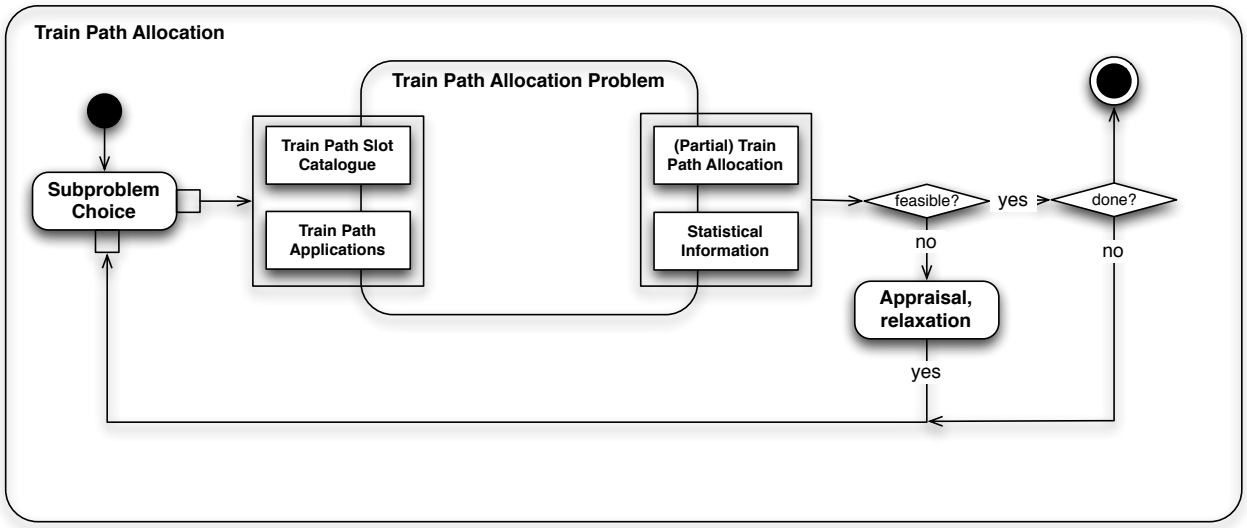
interactive human interpretation and action is required if the train path allocation problem cannot be solved or can be applied for the appraisal of the results;

open/online from one iteration to the next, train path slots and requests can be added or removed.

Section Synopsis and Discussion

In this section, the train path allocation was put in the context of the railway planning process in general and of timetabling in particular, based on concepts from (Caimi, 2009). It has become clear that many activities (such as the catalogue construction or microscopic scheduling) are beyond the scope of this thesis, which presupposes a train path catalogue and a set of train path applications as input data. The focus of the thesis is on freight trains without consideration of time nor connection dependencies between train path applications. It remains a conjecture whether the construction of a systematic train path catalogue with standardised and harmonised train paths helps to increase the degree of utilisation of the existing railway infrastructure.

Figure 6: Train Path Allocation Activity Diagram (UML 2.0 Activity Diagram). Refer to Figure 5 for the context of train path allocation within timetabling. Solving the train path allocation problem can be seen as a black-box activity in the view of domain experts and white-box to research specialists, who develop and run algorithms and interpret the results.



Source: Own representation

3 Modelling the Train Path Allocation Problem

In this section, the train path allocation problem will be defined in a formal way in terms of graph theory. The formulation will reflect the assumptions made in the previous Section 2.3: in particular, it presupposes a train path catalogue within a given macroscopic topology (infrastructure and corridor) and produces a feasible allocation for a set of train path allocations. The model reflects the commercially relevant aspects of train path application: the published offer, demand and delivery.

Section Outline

The underlying macroscopic railway topology will be defined as an undirected graph of system nodes and train path sections. An infrastructure manager's commercial offer will be represented by a train path slot catalogue with dwell times representing the technical time required at system nodes. Railway Undertakings may then apply for train paths between two system nodes at given start and end times and maximum earlier departure and maximum later arrival durations.

These notions define the train path allocation problem, which aims at satisfying train path applications by finding an allocation of train paths from the train path slot catalogue. If the train path allocation problem has more than one solution satisfying all train path applications, an optimisation criterion will allow to decide which solution to choose among all feasible train path allocations, for instance by minimising the sum of all train path durations plus their earliness and lateness with respect to the application's requested start and end time.

3.1 Macroscopic Topology and Routes

System nodes are those points where a train path of the commercial offer can start or end, e.g. a major railway station, a freight terminal or a point for ancillary services. It may be important to construct a train path catalogue to make best use of the available infrastructure along highly used corridors of the railway infrastructure (**condensation areas** (Caimi et al., 2011)) but not in the rest of the infrastructure (**compensation areas** (Caimi et al., 2011)). In this view, system nodes are the points interfacing between condensation corridors and compensation areas. An example of this interfacing role of system nodes is shown in Figure 7: here, the system nodes s and t serve as interfaces between the compensation areas (dashed lines and dots) and the macroscopic topology of train path sections (non-dashed black straight lines) and system nodes (filled black dots) covered by the train path catalogue.

System nodes abstract from the microscopic topology as shown in Figure 8: for instance, a train run from s_2 to s_3 may start at two different main signals within system node s_2 at the microscopic level; at the macroscopic level, it is not defined which one will be taken and feasibility has to be verified at the microscopic level. Also notice that, although a train run from s_4 to s_3 via s_2 requires extra time for the change of direction at s_2, this is not reflected in the macroscopic topology; these restrictions will be encoded in the train path catalogue and dwell time restrictions, as will be seen below.

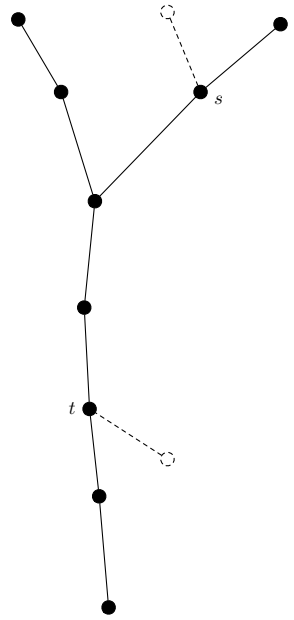
Definition 3.1 (Macroscopic Topology). *A **macroscopic topology** is a graph $\mathcal{T} = (S, E)$, consisting of*

- *a finite set $S \neq \emptyset$ of **system node**s (also called of nodes (or vertices) of the graph),*
- *$E \subseteq \{\{s,t\} : s \neq t, s,t \in S\}$ of **train path section**s (also called edges of the graph).*

The set of all train path sections involving a system node s is denoted $\delta(s) = \{e \in E : s \in e\}$. ♣

Notice that it is not formally required that the macroscopic topology be connected, although this will be the case in real-world scenarios.

Figure 7: System Nodes as Interfaces between Condensation Corridors and Compensation Areas. Traffic from a point (dashed dot) not covered by the train path catalogue must go through a system node (black dot).



Source: Own representation

Example 3.2 (Macroscopic Topology Models). Figure 13 shows three example macroscopic topologies, which are used throughout this thesis for illustration purposes and verification of the theory. Size and structure of Topology III are similar to the topology of the real-world data set.♣

Routes capture movements in a macroscopic railway topology over multiple edges.

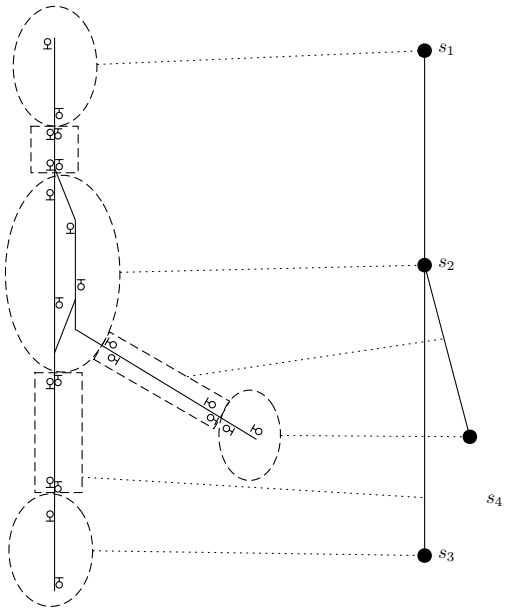
Definition 3.3 (Routes). *Given a macroscopic railway topology $\mathcal{T} = (S, E)$, a directed graph $\mathcal{D}_\mathcal{T} = (S, A)$ can be created by directing each edge $\{s,t\} \in E$, i.e., $A = \bigcup_{\{s,t\}\in E}\{(s,t),(t,s)\}$. A path $r = (s_1, \ldots, s_n) \in S^n$ in the directed graph $\mathcal{D}_\mathcal{T} = (S, A)$ is called a* ***route*** *from system node s_1 to s_n, i.e.,*

- $(s_i, s_{i+1}) \in A$, *for* $i = 1, \ldots, n-1$, *and*
- $s_i \neq s_j$, *(for all* $i \neq j$, $i, j \in \{1, \ldots, n\}$*).* ♣

3.2 Time Frame

Train paths are allocated with respect to a **planning horizon**, during which the same pattern is repeated n times. Typically the repeated **period** τ is a week and the planning horizon is the

Figure 8: Macroscopic Topology as Imprecise Aggregated View of Microscopic Elements: On the left-hand side, some elements of the microscopic level such as main signal positions and switches are depicted (but no release/clearing points after signals and switches of the interlocking system), and their corresponding aggregated elements of a macroscopic topology are depicted on the right-hand side. Aggregations are depicted by dashed boxes for train path sections and ovals for system nodes. The separation suggested here is at the block's home signal (coming out of the system node onto the train path section) and exit signal (going into the system node), which are not at exactly the same position in both directions and the aggregation is somewhat imprecise on the geographical level; however, different driving times in the train path section and through system nodes can be reflected in the train path slot catalogue.

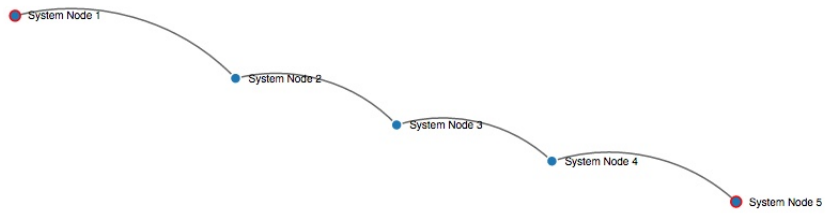


Source: Own representation

timetable period, which is typically one year. This model can of course be applied iteratively to different planning horizons one after another as long as the train path slots are available during the whole planning horizon and the application refers to the whole planning period. This is depicted in Figure 10(a): the planning horizon consists of n of periods of length τ (typically a week). Train services may also supposed to run repetitively at the same times daily or on a specific subset of operation days of the week, as shown in Figure 10(b): the same train run is scheduled on a subset $\Pi \subseteq \{1, \ldots, N\}$ of subperiods of length T, $\tau = NT$ over the whole planning period $n\tau$ (typically T is a day and $N = 7$, i.e., τ is a week). In this section, the case of train runs which are supposed to run repetitively at the same times daily or on a specific subset of operation days of the week, is not considered here. The following approach may be applied to the repetitive case: "flatten" repetitions into several individual train path slots and train path applications as shown in Figure 10(c); obviously it is not ensured here that the scheduling is

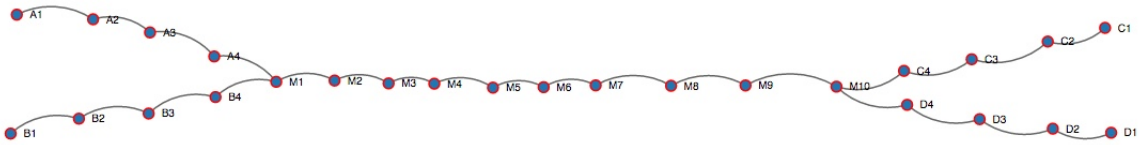
Figure 9: Synthetic Examples of Macroscopic Topologies

(a) Sample Topology I: line-shaped, small: 5 system nodes, 4 train path sections



(b) Sample Topology II: Bottleneck with claws, small: 6 system nodes, 5 train path sections

(c) Sample Topology III: Bottleneck with claws, large: 26 system nodes, 25 train path sections

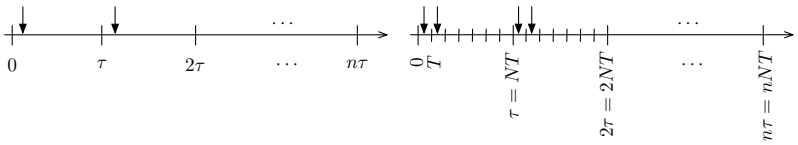


Source: Own representation

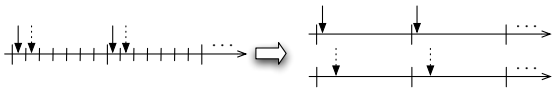
Figure 10: Periodical Train Path Allocation: with and without Periodicity

(a) Simple Periodical Train Path Allocation (without Periodicity)

(b) Periodical Train Path Allocation with Periodicity



(c) Flat Train Path Allocation: train path slots and applications are split into independent simple ones, periodicity is ignored.



Source: Own representation

the same on all requested days of service Π; in other words: the conversion to the simple case happens at the loss of the information on repetitions.

In the periodical setting, it is not possible to state that t_2 is after t_1, but only how much time it takes from t_1 to t_2: there is no notion of (symmetric) distance *between* two moments of time, but only of (directed) distance *from* one moment *to* another (which in general is different from the distance from the latter to the former moment, as exemplified in Figure 11).

Definition 3.4 (Time Frame). *Let $\tau \in \mathbb{R}$ be a positive real number, called* ***period****. Then, the half-closed interval $[0, \tau[$ is called* ***time frame****. In the time frame, time intervals may "wrap around τ"; formally, for $t_1, t_2 \in [0, \tau[$,*

$$[t_1, t_2]_\tau = \left\{ t : 0 \leq t < \tau, \left\{ \begin{array}{ll} t_1 \leq t \leq t_2 & \text{if } t_1 \leq t_2, \\ t \leq t_2 \text{ or } t \geq t_1 & \text{else} \end{array} \right\} \right\}. \tag{1}$$

Further, for two times $t_1, t_2 \in [0, \tau[$,

$$t_2 \ominus_\tau t_1 \quad = \quad (t_2 - t_1) \mod \tau \tag{2}$$

represents the distance from t_1 to t_2 or the duration of the interval $[t_1, t_2]$, and

$$t_1 \oplus_\tau t_2 \quad = \quad (t_1 + t_2) \mod \tau \tag{3}$$

represents the time t_2 after t_1. Finally,

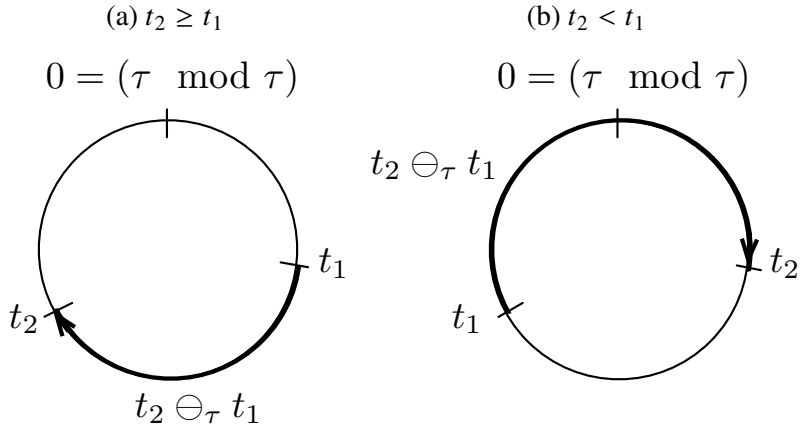
$$\Delta^+_\tau(t, t_1, t_2) \quad = \quad \left\{ \begin{array}{ll} 0 & \text{if } t \in [t_1, t_2]_\tau \\ t_2 \ominus_\tau t & \text{else,} \end{array} \right. \tag{4}$$

$$\Delta^-_\tau(t, t_1, t_2) \quad = \quad \left\{ \begin{array}{ll} 0 & \text{if } t \in [t_1, t_2]_\tau \\ t \ominus_\tau t_1 & \text{else,} \end{array} \right. \tag{5}$$

represent ***lateness****, i.e., the distance from the interval's upper bound t_2 to t, and* ***earliness****, i.e., the distance from t to the lower bound t_1, respectively.* ♣

Notice that the times in an interval $[t_1, t_2]_\tau$ are completely ordered.

Figure 11: Time Frame for Periodical Train Path Allocation: distance $t_2 \ominus_\tau t_1$, where time runs clock-wise modulo τ.



Source: Own representation

3.3 Train Path Slots and Train Paths

A train path slot links a train path section with direction and time. Notice that start and end times of a train path slot do not represent the time of mutually exclusive blocking of a resource at the microscopic level; blocking stairways work at the microscopic level and would typically be used in the construction of train paths slots or in order to check the microscopic feasibility and robustness of the overall allocation.

Definition 3.5 (Train Path Slot). *A **train path slot** in a macroscopic railway topology $\mathcal{T} = (S, E)$, $\mathcal{D}_\mathcal{T} = (S, A)$, with respect to a time frame $[0, \tau[$, is a quadruple $v = (s, t, \delta, \alpha)$ where*

- $(s, t) \in A$,
- *departure time $\delta \in [0, \tau[$ and arrival time $\alpha \in [0, \tau[$.*

For $v = (s, t, \delta, \alpha)$, the short-hand notations s_v, t_v, δ_v [2] and α_v will be used to denote v's source and target node and requested departure and arrival times, respectively. ♣

In order for this definition to be consistent with reality, slot durations must be strictly smaller than τ, i.e., the difference $\alpha \ominus_\tau \delta$ represents the physical slot duration of a slot v.

A train path catalogue represents the commercial offer of an Infrastructure Manager; dwell times allow for differentiation among incoming train path slot and next system node; in this

[2] Caveat: $\delta(s_v)$ denotes the set of neighbours of the system node s_v in the macroscopic topology, whereas δ_v denotes the scheduled departure time of the train path slot v.

way, turnaround times can be modelled. However, differentiation in driving behaviour (different standardised speed levels) cannot be modelled.

Definition 3.6 (Train Path Slot Catalogue). *A **train path slot catalogue** in a macroscopic railway topology $\mathcal{T} = (S, E)$ and with respect to a time frame τ is a pair (V, β), consisting of*

- *a finite set V of train path slots with respect to τ*
- *minimum dwell times $\beta_{v,s}$ for $v \in V$ and $s \in \delta(t_v)$, $s_v \neq s$,*

constituting the partial mapping $\beta : (v, s) \mapsto \beta_{v,s}$. ♣

Train path slots are the building blocks of train paths along a route.

Definition 3.7 (Train Path). *Given a train path catalogue V in a macroscopic railway topology $G = (V, E)$, a **train path** is a finite sequence $p = (v_1, v_2, \ldots, v_n)$ of n different train path slots $v_i = (s_i, t_i, \delta_i, \alpha_i)$, $i \in \{1, \ldots, n\}$, such that*

$$\left(\sum_{i=1}^{n} \underbrace{(\alpha_i \ominus_\tau \delta_i)}_{\text{slot time}} + \sum_{i=2}^{n} \underbrace{(\delta_i \ominus_\tau \alpha_{i-1})}_{\text{eff. dwell time}} \right) = (\alpha_n \ominus_\tau \delta_1), \tag{6}$$

(in words: the sum of slot lengths and (effective) dwell times equals the duration from the departure of the first slot to the arrival of the last slot modulo τ.) ♣

Equation (6) ensures that train paths are shorter than the period τ. Other formulations without this limitation are possible (Serafini and Ukovic, 1989).

On the consumer side, railway undertakings deposit requests to travel from one system node to another at some departure and arrival times.

Definition 3.8 (Train Path Application). *A **train path application** in a macroscopic railway topology $\mathcal{T} = (S, E)$ with respect to a time frame τ is a sextuple $k = (s_k, t_k, \delta_k, \alpha_k, \delta_k^-, \alpha_k^+)$ where*

- *departure time $\delta_k \in [0, \tau[$ and arrival time $\alpha_k \in [0, \tau[$.*
- *maximum earlier departure $\delta_k^- \in [0, \tau[$ and maximum later arrival $\alpha_k^+ \in [0, \tau[$*

such that $\delta_k^- + (\alpha_k \ominus_\tau \delta_k) + \alpha_k^+ < \tau$. ♣

The interpretation is the following: the application k asks for a train path from s_k to t_k with requested departure time δ_k and arrival time α_k; however, the applicant may not have knowledge

Table 2: Pro Memoria: Notation Train Path Allocation Problem

$s, t \in S$	system nodes
$[0, \tau[$	time frame
$u, v, w \in V$	train path slots
$\beta_{v,s}$	dwell time after v going to s
$p = (v_1, \ldots, v_n)$	train path
$k \in K$	train path application
s_k, s_u	source system node of a request/train path slot
t_k, t_u	terminal system node of a request/train path slot
δ_k/δ_u	requested/scheduled departure time
α_k/α_u	requested/scheduled arrival time
δ_k^-,α_k^+	maximum earlier departure / later arrival
$\check{\delta}_k = \delta_k \ominus_\tau \delta_k^-, \hat{\alpha}_k = \alpha_k \oplus_\tau \alpha_k^+$	earliest departure / latest arrival

Source: Own representation

about the available train path catalogue nor on the other applications; therefore, the application has to allow for some margin, i.e., the application allows for a train path within $[\delta_k \ominus_\tau \delta_k^-, \alpha_k \oplus \alpha_k^+]_\tau$. An application that allows for a larger margin takes into account that it may get a costlier and less attractive train path; for instance, the total cost c_p^k of a train path $p = (v_1, \ldots, v_n)$ may be defined as the sum of its **earliness** with respect to the requested departure time δ_k, its duration, and its **lateness** with respect to the requested arrival time α_k, i.e.,

$$c_p^k = \underbrace{\Delta_\tau^-(\delta_{v_1}, \delta_k, \alpha_k)}_{earliness} + \underbrace{(\alpha_{v_n} \ominus_\tau \delta_{v_1})}_{duration} + \underbrace{\Delta_\tau^+(\alpha_{v_n}, \delta_k, \alpha_k)}_{lateness}. \tag{7}$$

Of course, the applicant desires a train path of minimal cost; other definitions of c_p^k are possible, for instance lateness could be penalised exponentially. Hence, monetary discounts could be granted based either on the application bounds or the effective cost c_p^k of the allocated train path. Application bounds could also be bundled into service levels with corresponding monetary discounts. It is beyond the scope of this thesis to work out such a monetary pricing scheme. As discussed below in Section 4, a large class of cost functions c_p^k are possible (where c_p^k is short-hand notation for the cost function $c : (p, k) \mapsto c_p^k$).

3.4 Train Path Allocation Problem

A set of train path applications and a train path catalogue lead to the problem of allocating train path slots to these applications.

Definition 3.9 (Train Path Allocation Problem). *A structure $\pi = (S, E, V, \beta, K, c)$ where*

(P1) $\mathcal{T} = (S, E)$ *is a macroscopic railway topology,*
(P2) (V, β) *is a train path catalogue,*
(P3) K *is a finite set of train path applications and*
(P4) $c : (p, k) \mapsto c_p^k \in \mathbb{R}^+$ *attributes a non-negative cost to train path p for request k*

is called a ***train path allocation problem****. A* ***feasible train path*** *or* ***solution candidate*** *for request k is a train path $p = (v_1, \ldots, v_n)$ such that*

(U1) ***requested departure and arrival locations****:*

$$s_{v_1} = s_k, \qquad t_{v_n} = t_k \tag{8}$$

(U2) ***requested departure and arrival times****:*

$$\delta_{v_1}, \alpha_{v_n} \in [\check{\delta}_k, \hat{\alpha}_k]_\tau, \tag{9}$$

(U3) ***dwell times****:*

$$\delta_{v_{i+1}} \notin [\check{\delta}_k, \alpha_{v_i} \oplus_\tau \beta_{v_i,s}]_\tau, \qquad s = t_{v_{i+1}}, \text{ for all } i \in \{1, \ldots, n-1\} \tag{10}$$

(in words: the next departure $\delta_{v_{i+1}}$ at $t_{v_i} = s_{v_{i+1}}$ to $t_{v_{i+1}}$ must be at least $\beta_{v_i,s}$ after the arrival at t_{v_i}.)

A ***train path allocation*** *is a mapping $\mathcal{M} : V \to K \cup \{\diamond\}$ such that the set of train path slots*

$$\mathcal{M}^{-1}(k) = \{v \in V : \mathcal{M}(v) = k\} \tag{11}$$

constitutes a unique feasible train path

$$p_{\mathcal{M}}(k) = (v_1, \ldots, v_{|\mathcal{M}^{-1}(k)|}) \tag{12}$$

for all $k \in K$. The train path slots $\mathcal{M}^{-1}(\diamond)$ are unallocated. For a train path allocation problem π, the set of train path allocations is denoted $Sol(\pi)$. If, for a train path allocation problem π, there is no train path allocation f (i.e., if $Sol(\pi) = \emptyset$), then π is called ***infeasible****. An*

allocation $\mathcal{M} \in Sol(\pi)$ is ***optimal*** *if it achieves the minimal total cost*

$$\min_{\mathcal{M} \in Sol(\pi)} \sum_{k \in K} c^k_{p_{\mathcal{M}}(k)}). \tag{13}$$

A request $k \in K$ is called (in)feasible if the train path allocation problem $\pi_k = (S, E, V, \beta, \{k\})$ restricted to k is (in)feasible, respectively. ♣

Note that no time for dispatching at the start nor the end node is included in the model; however, such times can easily be modelled by modifying the applications requested departure and arrival times. Refer to Table 2 for a list of the notation used.

Example 3.10 (Sample Scenario I (without Earliness) in Sample Topology II). Consider Topology II of Example 3.2 with

- 4 train path slots per hour for every train path section (starting at :00, :15, :30, :45) of duration 20 minutes,
- global minimum dwell time of 5 minutes for every slot and all next system nodes,
- 1 train path application between the pairs $\{A_1, B_1\} \times \{C_1, D_1\} = \{(A_1, C_1), (A_1, D_1), (B_1, C_1), (B_1, D_1)\}$ per hour, all 4 hourly applications requesting to start at :00, to have duration 90 minutes, maximum later arrival of 60 minutes, but no earlier departure.

Consider now the bottleneck train path sections between system nodes $M_1, \ldots, M_{10}$. Trains can enter the bottleneck at M_1 coming from A_1 or B_1 at :00, :15, :30, and :45. Notice that all applications are to depart at :00 with no earlier departure permitted. Hence, the feasible train paths are

- start :00, enter M_1 at :30; duration and total cost 80 minutes;
- start :15, enter M_1 at :45; duration 80 minutes, lateness 5 minutes, and total cost 85 minutes;
- start :30, enter M_1 at :00; duration 80 minutes, lateness 20 minutes, and total cost 100 minutes;
- start :45, enter M_1 at :15; duration 80 minutes, lateness 35 minutes, and total cost 115 minutes.

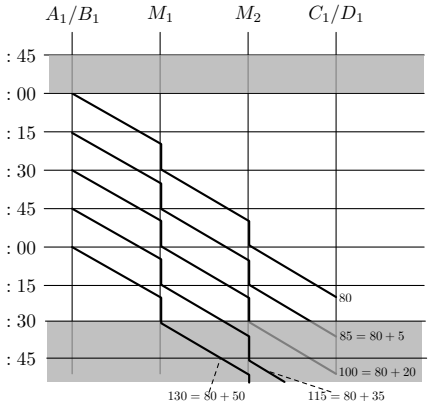
Idealised space-time trajectories of this situation are depicted in Figure 12(a). ♣

Example 3.11 (Sample Scenario II (with Earliness) in Sample Topology II). Consider the same train path allocation problem with he following modification:

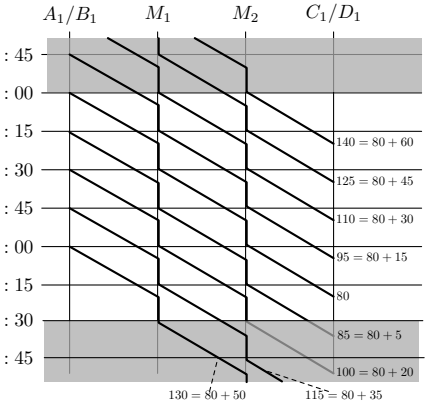
- allow for 60 minutes earlier departure and arrival.

Figure 12: Idealised Space-Time Diagrams for the Examples of Sample Topology II. The shaded areas represent times of earliness and lateness. The trajectories are labelled with their total cost.

(a) Space-time Diagram for Train Path Allocation Problems in Topology II, Scenario I (without Earliness)



(b) Space-time Diagram for Train Path Allocation Problems in Topology II, Scenario III (with Earliness)



Source: Own representation

Then, more train paths are feasible; the best four have a lower average cost:

- start :00, enter M_1 at :30; duration and total cost 80 minutes;
- start :15, enter M_1 at :45; duration 80 minutes, lateness 5 minutes, and total cost 85 minutes;
- start :45, enter M_1 at :15; duration 80 minutes, earliness 15 minutes, and total cost 95 minutes;
- start :30, enter M_1 at :00; duration 80 minutes, lateness 20 minutes, and total cost 100 minutes.

Idealised space-time trajectories of this situation are depicted in Figure 12(b). ♣

Example 3.12 (Sample Scenario III (Conflict in Sample Topology II)). Consider the same train path allocation problem with the following modification

- only 3 train path slots per hour for every train path section (starting at :00, :20, :40) of duration 20 minutes;

This train path allocation problem is infeasible since there are four train path applications per hour for entering the bottleneck at M_1, but only 3 slots entering the bottleneck at M_1 per hour.♣

Example 3.13 (Sample Scenario IV (Large) in Sample Topology III). The following scenario will be used to test the implementation in a large-scale scenario. Consider the sample topology of Figure 9(c) with

- 4 train path slots per hour for every train path section (starting at :00, :15, :30, :45) of duration 20 minutes;
- 1 train path application per hour between leaves of the topology, i.e., between the pairs $\{A_1, B_1\} \times \{C_1, D_1\} = \{(A_1, C_1), (A_1, D_1), (B_1, C_1), (B_1, D_1)\}$, all 4 hourly applications requesting to start at :00, to arrive $17 \cdot 30 = 8$ hours and 30 minutes later, maximum later arrival of $\alpha^+ = 60$ minutes, but no earlier departure ($\delta^- = 0$), minimum dwell time globally $\beta = 5$. ♣

Section Synopsis and Discussion

The notation used in this section follows that of (Bertsimas and Weismantel, 2005; Bertsimas and Tsitsiklis, 1997).

The model represents the commercial aspects of train path allocation; the key features of the suggested model are the following:

- the train path slot catalogue contains train path slots that are conflict-free at the macroscopic level, any set of train path slots can be allocated in principle; as a consequence headway constraints between different train path slots have to be encoded in the construction of the train path catalogue and are presupposed; microscopic scheduling and verification are required to ensure
  - node capacities: it has to verified whether a macroscopic allocation is conflict-free in system nodes; node capacities or train path slot exclusion constraints could be added to the model;
  - robustness: whether a macroscopic allocation is robust is not ensured by the allocation scheme; robustness has to be either encoded in the train path slot catalogue or verified by verification at the microscopic level;
- maximum earlier departure and minimum later arrival constraints per request allow for different service levels;
- minimum dwell times per slot and next node allow to model different durations needed for system node traversal;
- different speed characteristics in the train path catalogue and train path applications have not been taken into account; a simple approach could be to partition train path

allocation in disjoint subproblems (one per standardised profile) and jointly verified at the microscopic level (it would still have to be assumed that train path slots are conflict-free at the macroscopic level);
- the suggested optimisation objective is to minimise the sum of all allocation paths' duration, earliness and lateness; the impact of lateness could grow exponentially and not linearly, as suggested; in the theory developed in the next chapter, such objectives are possible;
- geographical constraints have not been added to the model; however, they will be used in the next section as a computational feature to keep the solution space tractable;
- monetary pricing of the allocated train paths could be based on the effective costs proportional to the sum of trip time plus lateness and earliness; other cost schemes could be introduced, but are beyond the scope of this thesis.

4 MIP formulations of the Train Path Allocation Problem

The formulation of the train path allocation problem in Section 3 was driven by the commercially relevant aspects of train path allocation. In order to keep the train path allocation problem tractable, the solution space may need to be pruned and represented in a mathematical model. In the literature, similar problems have been analysed as multicommodity flow problems (Barnhart et al., 2000; Caimi et al., 2011): the objective is to flow a set of commodities through an underlying network at minimum cost without exceeding capacities. In this terminology, train path applications are the commodities that have to flow through a network with restricted capacity. Since a slot must not be used more than once, the train path allocation problem is a multicommodity flow problem with unit capacity.

Section Outline

First, pruning will be introduced as a measure to reduce the solution space and to keep the problem tractable. Then, following (Barnhart et al., 2000), the solution space of a (pruned) train path allocation problem will be represented by a train path DAG, and two different mixed integer linear programs will be formulated to solve an origin–destination integer multicommodity flow problem based on a train path DAG. Finally, computational results of these two approaches will be presented with regard to dealing with large-scale problems.

4.1 Pruning

The solution space of a train path allocation problem can get quickly intractable, as will be seen below in Section 4.5. Therefore, it is essential to be able to **prune** the problem, i.e., to reduce its solution space. The search space can be reduced by imposing restrictions on

- dwell times,
- departure time of first slot,
- arrival time of last slot,
- geographically, system nodes to be used;

Pruning is formalised in the following definition; the notation is listed pro memoria in Table 3.

Definition 4.1 (Pruning). *Let $\pi = (S, E, V, \beta, K)$ be a train path allocation problem. Then, $(S, E, V, \beta, K, \bullet)$ is a* ***pruned train path allocation problem*** *where $\bullet$ is a set of values:*

- *lower, upper bound for train paths to depart at source system node: $\bullet\check{\delta}_k, \bullet\hat{\delta}_k \in [0, \tau[$ such that $[\bullet\check{\delta}_k, \bullet\hat{\delta}_k]_\tau \subseteq [\check{\delta}_k, \hat{\delta}_k]_\tau$;*
- *lower, upper bound for train paths to arrive at terminal system node: $\bullet\check{\alpha}_k, \bullet\hat{\alpha}_k \in [0, \tau[$ such that $[\bullet\check{\alpha}_k, \bullet\hat{\alpha}_k]_\tau \subseteq [\check{\delta}_k, \hat{\delta}_k]_\tau$;*
- *lower, upper bound for next train path slot to start to system node: $\bullet\check{\beta}_{v,s}, \bullet\hat{\beta}_{v,s} \in [0, \tau[$ such that $[\bullet\check{\beta}_{v,s}, \bullet\hat{\beta}_{v,s}]_\tau \cap [\alpha_v, \alpha_v \oplus_\tau \beta_{v,s}]_\tau = \emptyset$;*
- *system nodes allowed for application k: $\bullet S_k \subseteq S$.*

A solution candidate $p = (v_1, \ldots, v_n)$ (or feasible train path) for application $k \in K$ must then satisfy

(P1) ***requested departure and arrival locations****:*

$$s_{v_1} = s_k, \qquad t_{v_n} = t_k, \tag{14}$$

(P2) ***pruned requested departure and arrival times****:*

$$\delta_{v_1} \in [\bullet\check{\delta}_k, \bullet\hat{\delta}_k]_\tau, \qquad \alpha_{v_n} \in [\bullet\check{\alpha}_k, \bullet\hat{\alpha}_k]_\tau, \tag{15}$$

(P3) ***pruned dwell times****:*

$$\delta_{v_{i+1}} \in [\bullet\check{\beta}_{v_i,s}, \bullet\hat{\beta}_{v_i,s}]_\tau, \qquad s = t_{v_{i+1}}, \qquad \text{for all } i \in \{1, \ldots, n-1\}, \tag{16}$$

Table 3: Pro Memoria: Notation Pruned Train Path Allocation Problem

$\bullet\check{\delta}_k, \bullet\hat{\delta}_k \in \mathbb{R}$	lower, upper bound for train paths to depart at source system node
$\bullet\check{\alpha}_k, \bullet\hat{\alpha}_k \in \mathbb{R}$	lower, upper bound for train paths to arrive at terminal system node
$\bullet\check{\beta}_{v,s}, \bullet\hat{\beta}_{v,s} \in \mathbb{R}$	lower, upper bound for next train path slot to start to system node
$\bullet S_k$	system nodes allowed for request k

Source: Own representation

(P4) ***allowed system nodes****:*

$$s_{v_i}, t_{v_i} \in S_k, \qquad \textit{for all } i \in \{1, \ldots, n\}. \tag{17}$$

♣

By construction, a solution of a pruned problem is always a solution to the original problem. Therefore, pruning can be used to keep the solution space tractable. Furthermore, notice that there is a pruned problem having the same solutions for every original allocation problem by the following embedding:

- $\bullet\check{\delta}_k = \check{\delta}_k$, $\bullet\hat{\delta}_k = \hat{\alpha}_k$,
- $\bullet\check{\alpha}_k = \check{\delta}_k$, $\bullet\hat{\alpha}_k = \hat{\alpha}_k$,
- $\bullet\check{\beta}_{v,s} = \check{\beta}_{v,s}$, $\bullet\hat{\beta}_{v,s} = \tau$,
- $\bullet S_k = S$.

4.2 Multicommodity Flow

Following (Barnhart et al., 2000) and (Caimi et al., 2011), a train path allocation problem can be modelled as a multicommodity flow problem: train path applications are commodities that have to flow through a network.

Algorithm 1 constructs a directed acyclic graph (DAG) for an application in a pruned train path allocation problem; the idea is the following:

- starting at the source system node, possible train path slots are considered;
- the slots that satisfy the departure bounds are added to the graph;
- successor slots are considered along the routes allowed and added to the graph if dwell time constraints are satisfied; hence, arcs in the directed graph correspond to a slot-slot connection;

- if no successor can be found, the slot is recursively removed from the DAG;
- at the final system node, slots are added to the DAG if the arrival time constraint is respected and recursively removed if not.

Algorithm 1 Train Path DAG Construction

procedure CONSTRUCTDAG($k, \mathcal{T} = (S, E), V, \beta, \bullet$) ▷ Construct Train Path DAG $\mathcal{D}_k = (V_k, A_k)$ for application k, topology $\mathcal{T}$, catalogue V, dwell times β and pruning $\bullet$.
  $V_k \leftarrow \{s_k, t_k\}$
  $A_k \leftarrow \emptyset$
  $initialSlots \leftarrow \emptyset$
  $infeasibleSlots \leftarrow \emptyset$
  $processedSlots \leftarrow \emptyset$
  for all $t \in$ GETNEXTNODES$(s_k, s_k, t_k), v \in$ GETTRAINPATHSLOTS$(s_k, t, \bullet\check{\delta}_k, \bullet\hat{\delta}_k)$ **do**
    $V_k \leftarrow V_k \cup \{v\}$
    $A_k \leftarrow A_k \cup \{(s_k, v)\}$
    $initialSlots \leftarrow initialSlots \cup \{v\}$
  end for
  CONSTRUCTDAGITER($initialSlots$)
  for all $v \in infeasibleSlots$ **do** BACKTRACK(v)
  end for
  return (V_k, A_k)
  procedure GETTRAINPATHSLOTS($s, t, \check{\delta}, \hat{\delta}$)) ▷ Get train path slots from s to a system node t starting between $\check{\delta}$ and $\hat{\delta}$
    ...
  end procedure
  procedure GETNEXTNODES(u, s, t)) ▷ Get next nodes at u on a path from s to t using nodes from $\bullet S_k$.
    ...
  end procedure
▷ To be continued in Algorithm 2

Definition 4.2. *The graph $\mathcal{D}_k = (V_k, A_k)$ constructed by Algorithm 1 is called the* ***train path DAG*** *for request $k \in K$ of the pruned train path allocation problem.* ♣

Such a DAG encodes the solution candidates of the pruned train path allocation problem in the following way:

Theorem 4.3 *A train path $p = (v_1, \ldots, v_n)$ in the macroscopic topology $\mathcal{T} = (S, E)$ is a solution candidate of the pruned allocation problem if and only if $(s_k, v_1, \ldots, v_n, t_k)$ is a path in the train path DAG $\mathcal{D}_k$.*

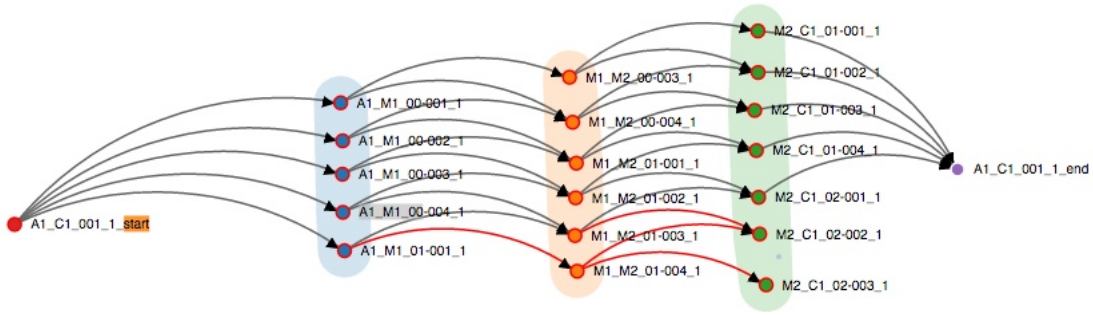
Algorithm 2 Train Path DAG Construction, continued

▷ Continuation of Algorithm 1

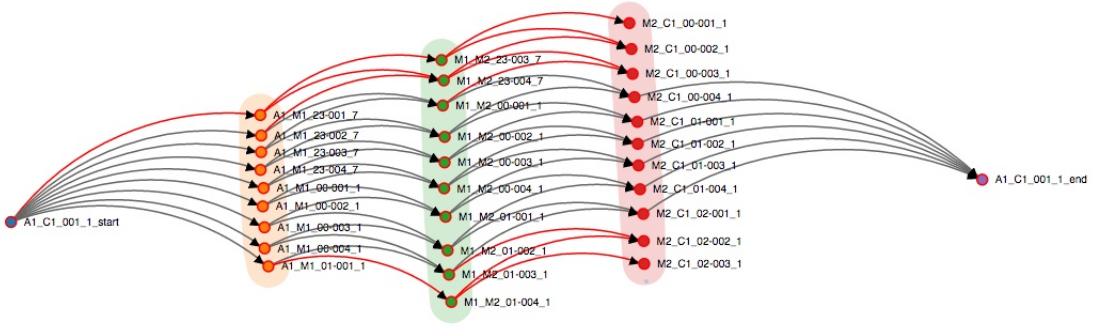
procedure CONSTRUCTDAGITER($slots$)
  $nextSlots \leftarrow \emptyset$
  for all $v \in slots \setminus (infeasibleSlots \cup processedSlots)$ **do** ▷ Slot v may be reached via different paths, but need not be reprocessed.
    $processedSlots \leftarrow processedSlots \cup \{v\}$
    if $v = t_k$ **then**
      if $\alpha_v \in [\bullet\check{\alpha}_k, \bullet\hat{\alpha}_k]_\tau$ **then**
        $V_k \leftarrow V_k \cup \{v\}$
        $A_k \leftarrow A_k \cup \{(v, t_k)\}$
      else
        $infeasibleSlots \leftarrow infeasibleSlots \cup \{v\}$
      end if
    else if $\alpha_v \notin [\bullet\check{\delta}_k, \bullet\hat{\alpha}_k]_\tau$ **then**
      $infeasibleSlots \leftarrow infeasibleSlots \cup \{v\}$
    else
      for all $t \in$ GETNEXTNODES(t_v, s_k, t_k), $w \in$ GETTRAINPATHSLOTS$(t_v, t, \bullet\check{\beta}_{v,s}, \bullet\hat{\beta}_{v,s})$ **do**
        $V_k \leftarrow V_k \cup \{w\}$
        $A_k \leftarrow A_k \cup \{(v, w)\}$
        $nextSlots \leftarrow nextSlots \cup \{w\}$
      end for
    end if
  end for
  if $nextSlots \neq \emptyset$ **then**
    CONSTRUCTDAGITER($nextSlots$)
  end if
end procedure
procedure BACKTRACK(v) ▷ If s is a leaf, recursively remove from DAG.
  if $\delta^+(v) = \emptyset$ **then** ▷ No outgoing edges, i.e., leaf?
    for all $w \in \delta^-(v)$ **do** ▷ Get parent slots in the DAG.
      $V_k \leftarrow V_k \setminus \{v\}$
      $A_k \leftarrow A_k \setminus \{(w, v)\}$
      BACKTRACK(w)
    end for
  end if
end procedure
end procedure

Figure 13: Train Path DAGs for Different Scenarios in Topology II; red arcs are those recursively removed from the DAG; vertices are grouped by train path section and vertically alined from earliest to latest from top to bottom. The labels of the intermediate nodes follow the pattern
<slot source>_<slot target>_<slot dep. hour>_<slot nb within hour>_<slot day>
and the the dummy nodes the pattern
*<appl. source >_<appl. target>_<appl. nb within day>_<appl. day>*_[start|end].

(a) Train Path DAG for Scenario I (without Earlier Departure)



(b) Train Path DAG for Scenario II (with Earlier Ddeparture)



Source: Own representation

In contrast to (Caimi et al., 2011), a train path DAG is not a resource tree, where every vertex has only one parent; since for a request k, the path leading to a slot v is irrelevant for the continuation of the train path; computation space and construction time can be saved by using this compact representation.

Notice that unit capacity in a train path DAG is virtually associated with vertices (which correspond to train path slots) and not with the arcs as in the standard multicommodity flow setting (Barnhart et al., 2000; Caimi et al., 2011) (which correspond to connections between two consecutive train path slots). However, a train path DAG can be expanded to comply with the original setting of (Barnhart et al., 2000), where capacities are associated with the arcs of the DAG. The exposition can be found in Appendix C.

4.3 Arc-Node Formulation

The choice of a particular feasible path for an application k can be expressed by assigning binary flow values x_a^k to each arc $a \in A_k$ in the expanded train path DAG such that each feasible train path corresponds to a feasible $s_k - t_k$ unit flow in the DAG $\mathcal{D}_k = (V_k, A_k)$,

$$x_a^k = \begin{cases} 1 & \text{if } k \text{ uses } a, \\ 0 & \text{else.} \end{cases}, \qquad \forall k \in K, \quad \forall a \in A_k, \tag{18}$$

Then,

$$\min \sum_{k \in K} \sum_{a \in A_k} c_a^k x_a^k \tag{19}$$

subject to

$$\sum_{k \in K} \sum_{w \in \delta_k^+(v)} x_{(v,w)}^k \leq 1, \qquad \forall v \in \bigcup_{k \in K} V_k, \tag{20}$$

$$\sum_{w \in \delta_k^+(v)}^{k} x_{(v,w)}^k - \sum_{u \in \delta_k^-(v)} x_{(u,v)}^k = \begin{cases} 1 & \text{if } v = s_k, \\ -1 & \text{if } v = t_k, \\ 0 & \text{else} \end{cases}, \qquad \forall k \in K, \quad \forall v \in V_k, \tag{21}$$

$$x_a^k \in \{0, 1\} \qquad \forall a \in A_k, \quad \forall k \in K \tag{22}$$

where $\delta_k^+(v) = \{w \in V_k : (v, w) \in A_k\}$ and $\delta_k^-(v) = \{u \in V_k : (u, v) \in A_k\}$, and the weights c_a^k are associated with the arcs of the train path DAG as follows:

- if $a = (s_k, v)$: $c_a^k = (\Delta_\tau^-(\delta_v, \delta_k, \alpha_k)) \oplus_\tau (\alpha_v \ominus_\tau \delta_v)$ (earliness, plus duration of slot v);
- if $a = (v, w)$: $c_a^k = (\delta_w \ominus_\tau \alpha_v) \oplus_\tau (\alpha_w \ominus_\tau \delta_w)$ (effective dwell time from v to w, plus duration of slot w);
- if $a = (v, t_k)$: $c_a^k = \Delta_\tau^+(\alpha_v, \delta_k, \alpha_k)$ (lateness).

Equation (20) ensures that unit capacity of each node (i.e., of each train path slot and dummy node s_k,t_k) is respected by all commodities (i.e., by all train path applications), and Equation (21) ensures that a flow starts at the source, is conserved along a path and ends at the sink for every commodity. Notice that an equivalent equation could be written for Equation (20) in terms of the ingoing arcs instead of the outgoing arcs, and, similarly, slot durations could be added to the costs of the arc leaving the node (instead of the arc leading to the slot).

4.4 Path-Based or Column-Generation Formulation

Alternatively, the feasible $s_k - t_k$ flows may be (partially) enumerated, which leads to a set $P(k)$ of feasible train paths. Let the set of all paths be denoted

$$P = \bigcup_{k \in K, p \in P(k)} V(p),$$

where $V(p)$ is the set of vertices in the path p in the train path DAG $\mathcal{D}_k$ for which $p \in P(k)$ (which is unique since s_k (or t_k) in the feasible flow uniquely determines the application k). A simultaneously feasible solution $F \subseteq P$ of the whole train path allocation problem must contain a solution candidate for each application k, and every slot must not be used more than once, i.e.,

$$|F \cap P(k)| = 1, \qquad \forall k \in K \tag{23}$$

$$V(p_1) \cap V(p_2) = \emptyset, \qquad \forall p_1, p_2 \in F. \tag{24}$$

This corresponds to a stable set (also called independent set) of size $|K|$ in the incidence graph of simultaneously feasible solutions $\mathcal{D}_P = (P, E(P))$,

$$E(P) = \left\{ \{p_1, p_2\} : \left\{ \begin{array}{l} \{p_1, p_2\} \subseteq p(k) \text{ for some } k \in \{k_1, k_2\}, \text{ or,} \\ V(p_1) \cap V(p_2) \neq \emptyset \end{array} \right\}, \quad p_1, p_2 \in P \right\}; \tag{25}$$

(a stable set $F \subseteq P$ is a set such that no two nodes are adjacent, i.e., $p_1, p_2 \in F \implies \{p_1, p_2\} \notin E(P)$). Therefore, the set of simultaneously feasible solutions are the stable sets

$$\mathcal{F}(\mathcal{D}_P) \subseteq \mathbb{R}^P = \left\{ y \in \mathbb{R}^P : \left\{ \begin{array}{ll} x_v + x_w \leq 1 & \{v, w\} \in E(P), \\ x_v \in \{0, 1\}, & v \in P \end{array} \right\} \right\} \subseteq \{0, 1\}^P. \tag{26}$$

Notice that

- the sets $E(P(k))$ of unique choice per commodity $k \in K$, and
- the sets $E(P(v))$ of train paths containing the same node $v \in \bigcup_{k \in K} V_k$,

are cliques (where $P(v) = \{p : v \in V(p)\}$, $v \in \bigcup_{k \in K, p \in P(k)} V(p)$); (a set $C \subseteq P$ is a clique if $\{p_1, p_2\} \in E(P)$ for all $p_1, p_2 \in P, p_1 \neq p_2$).

This problem can be encoded as a MIP problem as follows: define the variables

$$y_p^k = \left\{ \begin{array}{ll} 1 & \text{if } k \text{ chooses } p, \\ 0 & \text{else,} \end{array} \right\} \qquad \forall k \in K, \quad \forall p \in P(k). \tag{27}$$

Then,

$$\min \sum_{k \in K} \sum_{p \in P(k)} c_p^k \cdot y_p^k \tag{28}$$

subject to

$$\sum_{k \in K} \sum_{p \in P(k): v \in V(p)} y_p^k \leq 1, \qquad \forall v \in \bigcup_{k \in K} V_k \tag{29}$$

$$\sum_{p \in P(k)} y_p^k = 1, \qquad \forall k \in K, \quad \forall p \in P(k), \tag{30}$$

$$y_p^k \in \{0, 1\} \qquad \forall k \in K, \quad \forall p \in P(k). \tag{31}$$

Here, the conflict constraints Equation (29) ensures that every node (i.e., train path slot or dummy node s_k, t_k) is used at most once by any commodity k (i.e., train path application), and the choice constraints Equation (30) ensures that exactly one train path per train path application is chosen.

4.5 Theoretical and Computational Comparison of the Arc-Node and the Path-Based Formulation

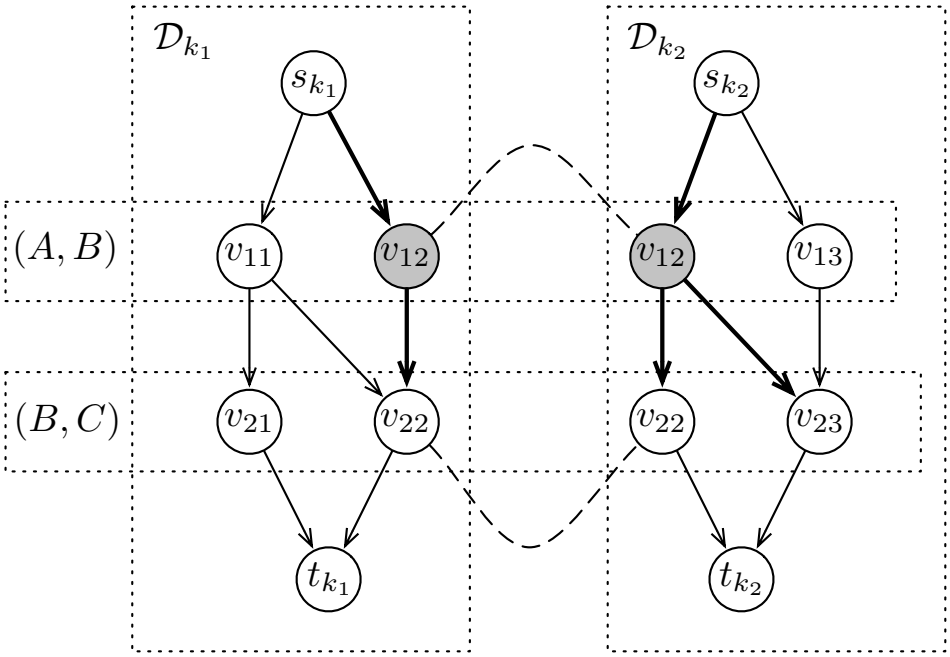
In order to compare the two formulations, consider the following Example 4.4.

Example 4.4 (Multicommodity). Consider Figure 14: The two applications k_1 and k_2 request a train path from A to C. The chosen pruning results in both DAGs having the same shape.
The arc-node model contains $2 \cdot 7 = 14$ variables according to Equation (18), one for each arc in the DAGs, whereas the path-based model contains $3 + 3 = 6$ variables according to Equation (27) (as many as train paths).
There are three flows in each DAG. Consider for instance the first DAG $\mathcal{D}_{k_1}$: assume that all slots along a train path section have the same duration and that the sequence of slots along (A, B) is v_{11}, v_{12}, v_{13} and, along (B, C) the sequence is v_{21}, v_{22}, v_{23} within $[\check{\delta}_{k_1}, \hat{\alpha}_{k_2}]_\tau$; if the train of application k_1 gets v_{11}, then it could go on with v_{21} or wait at system node B for v_{22}. The two

DAGs $\mathcal{D}_{k_i} = (V_{k_i}, A_{k_i})$, $i = 1, 2$, share two train path slots $\{v_{12}, v_{22}\} = V_{k_1} \cap V_{k_2}$. However, a train path slot must not be allocated more than once. To ensure this, the two models contains 10 unit capacity constraints according to Equation (20) and 10 conflict constraints according to Equation (29), respectively, one for each slot in the DAGs and the dummy nodes (obviously, the constraints that contain only one term such as those for the dummy nodes could be omitted). Consider the grey-shaded slot v_{12}: In the arc-node model, the unit capacity constraint for this slot contains a term for each application whose DAG contains the slot, i.e., 2 in this example; in the path-based model, the conflict constraint for this slot contains a term for each solution candidate passing through it in any application's train path DAG $\mathcal{D}_{k_i}, i = 1, 2$, i.e., $1 + 2 = 3$ in this example.
In the arc-node model, there is a flow conservation constraint according to Equation (21) for all nodes of all DAGs; consider again v_{12} and the the fat arcs in Figure 14: there are two flow conservation constraints for v_{12}, one for both applications, i.e., $2 \cdot 6 = 12$ in the example. Each of these constraint has as many terms as parent and child nodes, in total two terms arc in all DAGs, i.e., $2 \cdot 7 = 14$ in total in the example. In the path-based model, there is one choice constraint according to Equation (30) for each application, i.e., 2 in the example. Both these constraints contain as many terms as solution candidates in the respective DAG, i.e., $2 \cdot 3 = 6$ in total in the example. ♣

Theoretical and computational results on the structure of the MIP formulations are shown in Table 4 and Table 5, respectively. The theoretical considerations show that the size of the arc-node formulation is essentially given by the size of the train path DAGs; the number of constraints is the total number of nodes and the number of variables is the total number of arcs. In the path-based formulation, the number of variables is the total number of enumerated solution candidates, and the number of constraints of order the total number of slots plus the number of train path applications. The computational results show that the naive approach of enumerating all solution candidates in the pruned train path DAGs leads to a large number of variables and a huge number of terms. On the other hand, the path-based approach has considerably fewer constraints. Hence, for even larger problems than the ones considered, it may be worthwhile to enumerate only a few selected solution candidates and to add more at need in a branch-and-price-and-cut approach (Barnhart et al., 2000); to develop such an approach is beyond the scope of this thesis. The numbers of the real-world data set are also illustrated in Figure 15: the (full-blown) constraints-variables matrix in the arc-node model is almost quadratic, whereas, in the path-based model, its height is smaller and its width much larger. Experiments have shown that, if the pruning and the partial enumeration in the path-based model are not strict enough, the number of terms in the conflict terms can grow very quickly and exceed the available memory, while the same pruned problem is still tractable using the arc-node representation.

Figure 14: Example of a Simple Multicommodity Flow Problem of two Train Path Applications. Two train path DAGs for applications k_1 and k_2 are shown. Train path slots which appear in both train path DAGs are linked by a dashed line. Horizontal dashed boxes are drawn around slots along the same directed train path sections (A, B) and (B, C).



Source: Own representation

Table 4: Theoretical Characteristics of the two MIP formulations in Terms of the Train Path DAG $\mathcal{D}_k = (V_k, A_k)$.

number of		arc-node		path-based
variables	(18)	$\sum_{k\in K} \lvert A_k \rvert$	(27)	$\sum_{k\in K} \lvert P(k) \rvert$
unit capacity / conflict constraints	(20)	$\lvert \bigcup_{k\in K} V_k \rvert \leq \sum_{k\in K} \lvert V_k \rvert$	(29)	$\lvert \bigcup_{k\in K} V_k \rvert \leq \sum_{k\in K} \lvert V_k \rvert$
flow conservation / choice constraints	(21)	$\sum_{k\in K} \lvert V_k \rvert$	(30)	$\lvert K \rvert$
unit capacity / conflict terms		$\sum_{k\in K} \lvert A_k \rvert$		$\sum_{k\in K} \sum_{p\in P(k)} \lvert V(p) \rvert$
flow conservation / choice terms		$\sum_{k\in K} \lvert A_k \rvert$		$\sum_{k\in K} \lvert P(k) \rvert$

Source: Own representation

Table 5: Computational Results of the two MIP formulations of the Synthetic Sample Scenario IV of Example 3.13 and the Real-World Data Set. The table shows the size of the initial pruned model. The global (hard) bounds (in minutes) are
- synthetic data set: $\delta^- = 0$, $\beta = 5$ and $\alpha^+ = 60$, and
- real-world data: $\delta^- = 60$, $\beta = 10$ and $\alpha^+ = 60$.

	arc-node	path-based
Scenario IV in Sample Topology III (bottleneck with claws, large)		
Number of feasible applications:	1'344	
Number of variables	120'960	6'720
Number of constraints	150'528	34'944
Number of unit capacity / conflict constraints	33'600	33'600
Number of terms	356'160	120'960
Number of terms in unit capacity/ conflict constraints	114'240	114'240
real-world data set		
Number of feasible applications:	4'218	
Number of variables	61'895	160'364
Number of constraints	61'036	19'225
Number of unit capacity / conflict constraints	15'007	15'007
Number of terms	171'602	1'780'940
Number of terms in unit capacity/ choice constraints	47'812	160'364

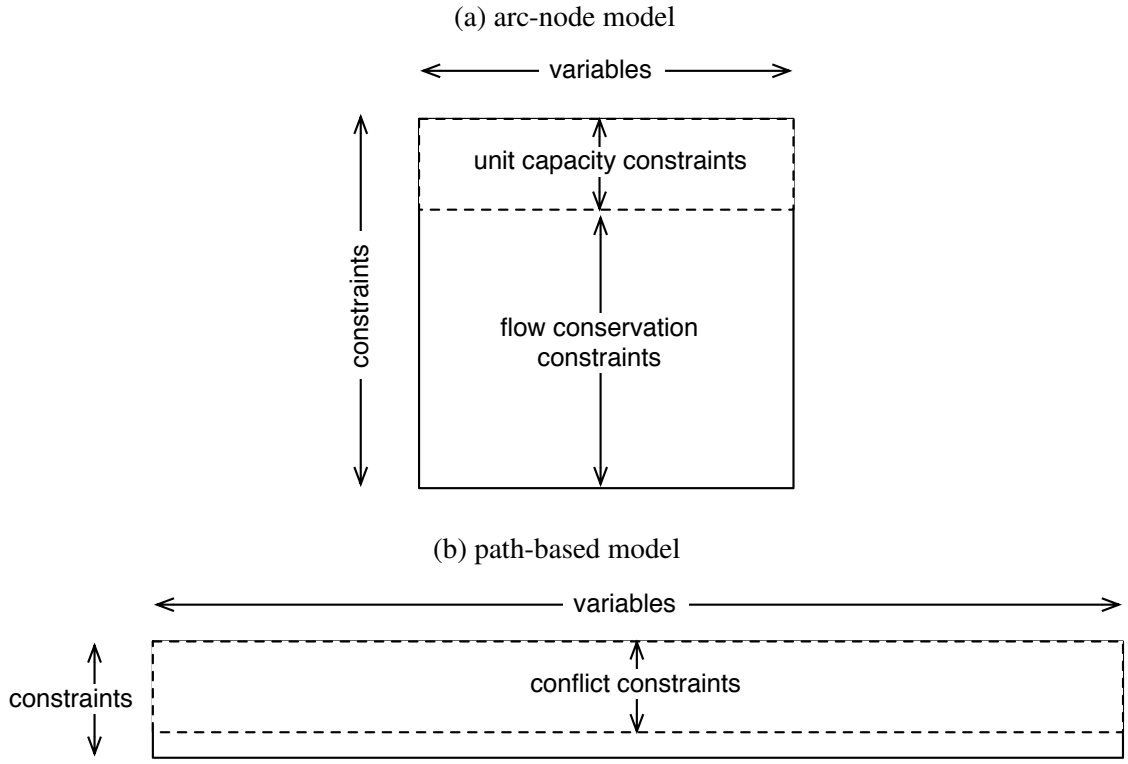
Source: Own representation

Section Synopsis and Discussion

The construction of train path DAGs allows to represent the solution candidates of a train path application in a compact way: a solution candidate is any path in the DAG from the unique root node (i.e., a node without parents) to the unique leaf node (i.e., a node without children). From this representation, two MIP formulations can be derived:

- an arc-node formulation, which encodes flow conservation constraints and works well for the size of problems at hand;
- a path-based formulation, which is based on a (partial) enumeration of train paths and which does not work well in larger problems if the selection of enumerated solution

Figure 15: Graphical Representation of the Computational Results of the Real-World Data Set of Table 5: the numbers of variables and constraints are drawn proportionally against the number of constraints.



Source: Own representation

candidates is not well controlled; however, for even larger problems, it may be worthwhile to apply a branch-and-price-and-cut approach (Barnhart et al., 2000).

The theory developed in this section easily allows for other cost functions $c : (p, k) \mapsto c_p^k$ than the linear one suggested:

- in the arc-node formulation (19), the cost of a feasible train path is summed over costs associated with arcs in the train path DAG, i.e., with slot–slot, source–slot and sink–slot connections,
- in the path-based formulation (28), no restriction is imposed at all on the structure of the cost function $c : (p, k) \mapsto c_p^k$.

It is beyond the scope of this thesis to discuss such alternative objective functions.

5 A Worfklow for the Train Path Allocation Problem

With the two MIP formulations of the train path allocation problem at hand, what are the activities to solve the train path allocation problem? Which activities can readily be implemented in software and where is human interaction necessary at the present stage of research?

Section Outline

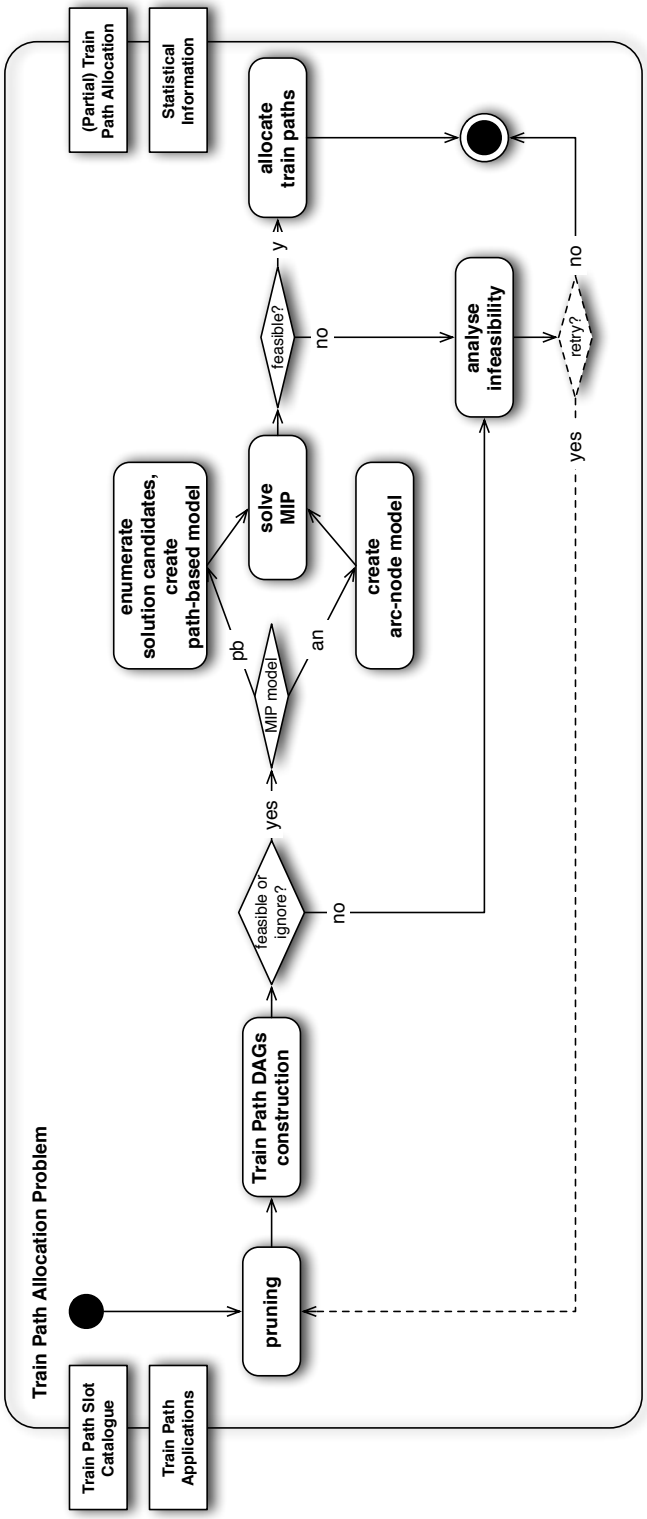
First, a semi-automatic workflow will be presented that will allow an operations research expert to solve a train path allocation problem. Furthermore, a train path application problem can be infeasible because individual train path applications are infeasible or because it is not possible to find an allocation satisfying *all* applications. Infeasible applications are detected when their DAG is constructed; an IIS analysis allows to find a set of dead-locked slots and applications. The reporting of these issues is part of the output of the train path allocation workflow.

5.1 A Semi-Automatic Workflow for the Train Path Allocation Problem

Figure 16 gives a semi-automatic workflow for the train path allocation problem; a command line tool has been implemented with the following features:

- *input data*: the train path catalogue and the train path applications are stored in a spreadsheet file (whose sheet and column layout can be configured in the tool); the train path catalogue and the train path applications are periodical referring to the same planning horizon and have a periodicity over the week; these applications and slots are "flattened" into simple periodical slots and applications, one per (requested) service day (see above Figure 10(c));
- *hard bounds*: the applications' maximum earlier departure and maximum later arrival and the catalogue's dwell times are specified globally; they are not specified in the spreadsheet, but are given as global parameter via a command line option; they were not present in the real-world data set;
- *subproblem choice*: a subset of unsatisfied train path applications can be chosen by a pattern specifying a set of periodicities to select the corresponding train path applications;
- *initial pruning*: the initial pruning parameters are set to default values; in case of infeasibility with these default values, the initial pruning parameters are determined from

Figure 16: A Semi-Automatic Workflow for the Train Path Allocation Problem (UML 2.0 Activity Diagram). The dashed parts are not implemented and intervention by an operations research expert is required.



Source: Own representation

an earliest path heuristic within the requested bounds and global minimum dwell time or within the requested bounds enlarged by the global maximum earlier departure and maximum later arrival bounds; geographical constraints have not been implemented;

- *partial allocation*: instead of reporting infeasibility, infeasible applications may optionally be removed from the allocation problem when their DAG is constructed and found infeasible;
- *MIP model*: whether the path-based or the arc-node model should be used, can be specified by a command line option; the model is passed to a Gurobi solver (Gurobi Optimization, Inc., 2015);
- *allocation*: train path slots which have been allocated and train path applications which have been satisfied are simply marked in the spreadsheet file and will be filtered out if the file is taken again as input when the next subproblem is dealt with;
- *pruning*: the choice of new pruning parameters in case of infeasibility has not yet been implemented (dashed line in Figure 16); this will be the topic of the in Subsection 5.2;
- *statistical information*: in the current state of the implementation, statistical information has to be interpreted by an operations research expert:
  - *computational information*: web pages displaying the train path DAGs (with those nodes and arcs that are recursively removed) are produced and lists containing information on the models and DAGs (such as number of constraints, number of arcs, initial pruning parameters per system node) are compiled; further information (such as computation times) can be found in the log file;
  - *infeasibility*: a so-called IIS analysis is performed and the result is displayed (see Section 5.2);
  - *commercial information* (such as comparing individual best-cost against the effective cost of the found in the overall allocation) is not yet compiled in the implementation.

For more details, refer to the command line options listed in Appendix B and the list of used technologies in Appendix A.

5.2 Identifying the Source of Infeasibility of a Train Path Allocation Problem

Although all individual applications may be feasible (this is the case if there is at least one path from s_k to t_k in the corresponding train path DAG), the allocation problem of finding an allocation for all applications may still not be feasible. In the case of infeasibility, how can useful information be derived from the model in order to proceed in the assignment process? This is the situation of the dashed arrow at the bottom of Figure 16.

An approach comes from the observation that the allocation problem is infeasible if a single train path slot is used in all solution candidates of several requests; and, similarly, if two train paths slots are used in every combination of solution candidates of three requests. These observations motivate the following definition (SAS Institute Inc., 2009).

Definition 5.1 (Irreducible Infeasible Set (IIS)). *An irreducible infeasible set (IIS) is an infeasible subset of constraints that become feasible if any single constraint or variable bound is removed.* ♣

Notice that it is possible to have more than one IIS in an infeasible problem, i.e., an IIS gives a partial explanation of the problem's infeasibility and reflects the current state of the solver's solution procedure. Commercial solvers such as Gurobi (Gurobi Optimization, Inc., 2015) can provide an IIS in the case of model infeasibility.

Definition 5.2 (IIS Train Path Slots, IIS Train Path Applications). *A constraint of an IIS falls in one of three categories:*

- *it corresponds to a unit capacity constraint (20) or conflict constraint (29) of a train path slot: the corresponding train path slots are called* ***IIS slots*** *for the IIS;*
- *it corresponds to a flow constraint (21) or path choice constraint (30) of a train path application: the corresponding train path applications are called* ***IIS train path applications*** *for the IIS;*
- *it corresponds to a flow constraint (21) or path choice constraint (30) of a dummy node s_k or t_k in the train path DAG.* ♣

This definition allows to locate the source of the infeasibility.

Example 5.3 (IIS Analysis in Scenario III (Conflict)). Consider again Example 3.12. Here, the demand of 4 services per hour is larger than the available services (which is only 3 services per hour). In case of infeasibility, the automatic IIS analysis locates the source of infeasibility as shown in the output of Figure 17: the Gurobi solver detects infeasibility and gives a list of 19 constraints which form the IIS. From this list, 4 applications and 3 slots are identified from 16 flow conservation constraints and 3 unit capacity constraints. ♣

Based on an IIS analysis, a re-pruning heuristic could be developed as follows:

1. *feasibility check* on the subproblem of IIS applications: Is the (unpruned) allocation problem restricted to the IIS applications feasible? If not, the whole allocation problem is infeasible. This subproblem might be large and thus this check might not be practically tractable.

Figure 17: IIS Analysis Output for Scenario III (Conflict) in an Arc-Node MIP: the IIS contains 19 constraints, 16 of which are IIS application constraints and 3 are IIS slot constraints. They map to 4 applications and 3 slots. Interpretation: there are only 3 slots at the bottleneck train path section (M_2, M_1) per hour and direction, but 4 applications per hour and direction requiring the bottleneck.

```
Before optimization: model has 19824 constraints and 16128 variables
Optimize a model with 19824 rows, 16128 columns and 44352 nonzeros
Variable types: 0 continuous, 4032 integer (4032 binary)
Root relaxation: infeasible, 2801 iterations, 0.02 seconds
    Nodes | Current Node | Objective Bounds | Work
 Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/ode Time
     0 0 infeasible 0 − infeasible − − 0s
Explored 0 nodes (2801 simplex iterations) in 0.37 seconds
The model is infeasible; computing IIS
Computing Irreducible Inconsistent Subsystem (IIS)...
      Constraints Bounds Runtime
     Min Max Min Max
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
        0 19824 0 32256 0s
IIS computed: 19 constraints, 0 bounds
IIS runtime: 0.25 seconds
The following constraint(s) cannot be satisfied:
Found IIS constraint fc|D1_B1_005_4|D1_B1_005_4_end
Found IIS constraint fc|D1_B1_005_4|M1_B1_05−002_4
Found IIS constraint fc|D1_B1_005_4|M1_B1_06−001_4
Found IIS constraint fc|D1_B1_005_4|M1_B1_05−003_4
[...]
Found IIS constraint ucc|M2_M1_05−002_4
Found IIS constraint ucc|M2_M1_04−003_4
Found IIS constraint ucc|M2_M1_05−001_4
Found 3 IIS slots and 4 IIS train path applications:
 IIS slot #M2_M1_05−001_4 [Do 05:00:00,Do 05:20:00], (M2, M1)
 IIS slot #M2_M1_04−003_4 [Do 04:40:00,Do 05:00:00], (M2, M1)
 IIS slot #M2_M1_05−002_4 [Do 05:20:00,Do 05:40:00], (M2, M1)
 IIS train path application #C1_A1_005_4 [Do 04:00:00,Do 05:30:00], (C1, A1)
 IIS train path application #D1_A1_005_4 [Do 04:00:00,Do 05:30:00], (D1, A1)
 IIS train path application #C1_B1_005_4 [Do 04:00:00,Do 05:30:00], (C1, B1)
 IIS train path application #D1_B1_005_4 [Do 04:00:00,Do 05:30:00], (D1, B1)
```

2. *relaxation of pruning*: increase the maximum dwell time, later arrival and earlier departure step-wise for all applications whose DAG contains an IIS slot until new slots have been added at some of the train path sections of the IIS slots.
3. *warm start*: solve the updated problem; if it is infeasible, go back to the first step and loop until either an infeasible subset of applications is found or the problem is solved.

Feedback to planners could be based on the following ingredients:

1. list of infeasible train path allocations: the train path allocations need to be checked against the train path catalogue;
2. list of IIS applications and slots identified recursively by removing one of the IIS application constraints until the remaining subproblem problem is feasible: in this case, the applications need to be modified or rejected or the catalogue needs to be modified.

Section Synopsis and Discussion

In case of infeasibility, an IIS analysis allows to identify the source of infeasibility in terms of the train path catalogue and train path applications. However, at the present stage of development, this information has to be compiled and interpreted by an operations research expert to make it accessible to domain-expert in timetable planning or to applicants.

6 Conclusion

6.1 Short Summary

In this thesis, a mathematical formulation of the train path allocation problem has been proposed and two MIP formulations have been given and discussed and proven to be applicable to the size of a real-world data set. The pivotal element is the concept of train path DAG, which allows to represent the possible train paths of a train path application of a (pruned) allocation problem. Furthermore, a tool has been implemented which allows to deal with train path allocation in an interactive, iterative and open approach and is able to visualise topologies, train path DAGs and lists with pruning parameters for infeasibility analysis.

6.2 Further Research

Although a first step has been made, a lot of theoretical, algorithmic and commercial questions remain open in the proposed model and algorithms:

- *cost functions and service levels*: instead of a linear cost function as suggested, other cost functions could be implemented, e.g., earliness and lateness could be penalised by an exponential function and they could be weighted (see service levels below);
- *interfacing*: for requests coming from outside or going into the defined macroscopic topology, interface points and times could be calculated;
- *microscopic scheduling*, in particular in system nodes: it should be verified whether the allocations are conflict-free at the microscopic level and whether they are robust;
- *improved pruning*: in the case of infeasibility, an automatic choice of pruning parameters should be implemented;
- modelling: system node *capacity constraints* and *crossing constraints* at system nodes could be added to the model;
- different *standardised speed levels* could be defined in the train path catalogue and the train path applications;
- *train path allocation with periodicity* should be worked out in detail and implemented;
- *train categories*: different train categories reflecting different speed characteristics are not reflected in the train path slot catalogue; as suggested above, a simple approach would be to design the train path slot catalogue (still conflict-free at the macroscopic level) with a slot set for each category;
- *exclusive alternatives in the train path slot catalogue*: for instance, in order to model alternatives for different train categories or different directions on the same tracks, conflict-freeness at the macroscopic level would have to be dropped and exclusion-constraints would have to be introduced in the model; an appealing feature of the current approach is that such constraints are not necessary;
- *hierarchical decomposition*: a subproblem could be split into subproblems which are solved sequentially in a divide-and-conquer approach, for instance based on periodicity (see Appendix D) or on service levels; decomposition could be further refined using:
  - *automatic application splitting*: in the case of infeasible repeated periodical applications, an automatic split of the application into two independent applications could be implemented;
  - *warm start of iterations*: allocated train path slots from previous subproblems could be taken as flexible and used as initial solutions (such that the overall solution quality may be improved);
- commercial definition of *service levels* with corresponding slot bounds, cost functions,

and monetary pricing;
- the *strength of the MIP formulation* should be analysed;
- a *column generation approach with pricing* could be applied in order to deal with large-scale models.

7 References

Barnhart, C., C. A. Hane and P. H. Vance (2000) Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems, *Operations Research*, **48** (2) 318–326.

Beck, M. J. (2015) *Industrialisierung der Fahrplanung bei der DB Netz AG*, DB Netz AG, http://www.it15rail.ch/.

Bertsimas, D. and J. Tsitsiklis (1997) *Introduction to Linear Optimization*, 1st edn., Athena Scientific, ISBN 1886529191.

Bertsimas, D. and R. Weismantel (2005) *Optimization over integers.*, Athena Scientific, ISBN 978-0-97591-462-5.

Bucher, T. (2014) *NeTS-AVIS Tagesfahrplan BV4b Handbuch Besteller Version i41*, Schweizerische Bundesbahnen SBB AG. https://www.sbb.ch/content/sbb/de/desktop/sbb-konzern/sbb-als-geschaeftspartner/zugang-zum-bahn-netz/onestopshop/systeme/nets/_jcr_content/relatedPar/relateddownloadlist/downloadList/benutzerhandbuch_.spooler.download.pdf, accessed 2015/11/02.

Bussieck, M. R., T. Winter and U. T. Zimmermann (1997) Discrete optimization in public rail transport, *Mathematical Programming*, **79** (1) 415–444, ISSN 1436-4646.

Caimi, G., F. Chudak, M. Fuchsberger, M. Laumanns and R. Zenklusen (2011) A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling, *Transportation Science*, **45** (2) 212–227.

Caimi, G. C. (2009) Algorithmic decision suport for train scheduling in a large and highly utilised railway network, Ph.D. Thesis, Eidgenössische Technische Hochschule ETH Zürich, Nr. 18581, 2009.

Council of the European Communities (1991) *Council Directive on the development of the Community's railways (91 /440 /EEC)*. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31991L0440:EN:PDF, accessed 2015/11/23.

Council of the European Communities (2001) *Directive 2001/14/EC on the allocation of railway infrastructure capacity and the levying of charges for the use of railway infrastructure and safety certification*. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2001:075:0029:0046:EN:PDF, accessed 2015/11/23.

European Parliament and Council of the European Communities (2010) *Regulation (EU) No 913/2010 concerning a European rail network for competitive freight*. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:276:0022:0032:en:PDF, accessed 2015/11/23.

Federal Office of Transport FOT (2016) *Glossar*. http://www.bav.admin.ch/glossar/index.html?action=character&character=all&descr=true&lang=en, accessed 2016/01/01.

Gurobi Optimization, Inc. (2015) *Gurobi Optimizer Reference Manual. Version 6.5.*, Gurobi Optimization, Inc. http://www.gurobi.com/documentation/6.5/refman.pdf, accessed 2015/11/23.

Herrigel-Wiedersheim, S. (2015) Algorithmic decision support for the construction of periodic railway timetables, Ph.D. Thesis, ETH-Zürich. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 22548, http://e-collection.library.ethz.ch/eserv/eth:47610/eth-47610-02.pdf.

Opitz, J. (2009) Automatische erzeugung und optimierung von taktfahrplänen in schienenverkehrsnetzen., Ph.D. Thesis, TU Dresden.

SAS Institute Inc. (2009) *SAS/OR® 9.2 User's Guide Mathematical Programming*, SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513. https://support.sas.com/documentation/cdl/en/ormpug/59679/PDF/default/ormpug.pdf, accessed 2015/11/23.

SBB (2009) *NeTS 1.9 Benutzerhandbuch NeTS-AVIS EVU. Version 1.0.*, Schweizerische Bundesbahnen SBB AG. http://www.sbb.ch/content/dam/sbb/de/pdf/sbb-konzern/sbb-als-geschaeftspartner/angebote-fuer-evus/onestopshop/systeme/Benutzerhandbuch_01_090623i18.pdf, accessed 2016/01/28.

SBB (2016) *NeTS - Netzweites Trassen-System*, Schweizerische Bundesbahnen SBB AG. http://www.sbb.ch/sbb-konzern/sbb-als-geschaeftspartnerin/angebote-fuer-evus/onestopshop/systeme/nets.html, accessed 2016/01/28.

Serafini, P. and W. Ukovic (1989) A mathematical model for periodic scheduling problems, *SIAM J. Disc. Mat*, **2** (4) 550–581.

SMA and Partners Ltd. (2015) Homepage sma and partners ltd., `http://www.sma-partner.ch/en/`, accessed 2015/11/23.

Trasse Schweiz AG (2016) *Glossary*. `http://trasse.ch/en/faq`, accessed 2016/01/01.

Tzieropoulos, P. and D. Emery (2009) How regular is a regular-interval timetable? theoretical foundations and assessment methodology, paper presented at the *Ôø°STRC Swiss Transport Research Conference*. `http://www.strc.ch/conferences/2009/Tzieropoulos.pdf`.

Weidmann, U. (2011) *Bahninfrastrukturen*, `http://www.ivt.ethz.ch/oev/publications/skripte`.

Weiß, R. (2015) *Industrialisierte Kapazitätsoptimierung mit einem vorkonstruierten Trassenkatalog*, DB Netz AG, `http://www.it15rail.ch/`.

Wikipedia (2015) *Online Algorithm*. `https://en.wikipedia.org/wiki/Online_algorithm`, accessed 2016/01/25.

Acknowledgement

The author would like to thank:

- SMA and in particular Dr. A. Fertis for providing such an interesting topic for this DAS thesis and for his support and discussions with him during this thesis;
- the anonymous European Railway Company for providing the real-world data set;
- the author's employer SBB for supporting his DAS studies;
- Dr. R. Bürgy for pointing out the paper on the multicommodity flow problem, his valuable comments on the multicommodity flow formulations and for proofreading;
- Prof. Dr. U. Weidmann for kindly accepting this thesis.

A Used Technologies

The technologies used in the implementation are listed in Table 6. The 3rd-party Maven Dependencies and their License are listed in Table 7,

Table 6: Used Technologies

Technology	Description
MacBook Pro, i5 2.53 GHz, 8 GB RAM	Hardware
Gurobi Optimizer 6.5.0	Linear programming solver
IDEA IntelliJ (Community Edition) 14	Java IDE
Oracle JDK 1.7.0_79 incl. VisualVM	Java Development Kit, VisualVM for performance analysis
Apache Maven 3.3.3	Build tool
SonarQube 5.1	Codequality analysis
d3js 3.5.6	Graphs and matrices
Datatables 1.10, Bootstrap 3.3	Data tables
OmniGraffle Pro 4.2.1	Vector graphics
TeXShop 3.18	Typesetting
LaTeXiT 2.5.2	Mathematical formulae in vector graphics

Source: Own representation

Table 7: Used 3rd-Party Libraries

GroupId	ArtifactId	Version	Type	License
commons-cli	commons-cli	1.3	jar	Apache License 2.0
commons-io	commons-io	2.4	jar	Apache License 2.0
com.google.guava	guava	19.0	jar	Apache License 2.0
joda-time	joda-time	2.8.1	jar	Apache License 2.0
junit	junit	4.12	jar	Eclipse Public License 1.0
org.apache.commons	commons-lang3	3.4	jar	Apache License 2.0
org.apache.logging.log4j	log4j-api	2.3	jar	Apache License 2.0
org.apache.logging.log4j	log4j-core	2.3	jar	Apache License 2.0
org.codehaus.plexus	plexus-utils	3.0.20	jar	Apache License 2.0
org.joda	joda-convert	1.7	jar	Apache License 2.0
org.json	json	20140107	jar	The JSON License
org.mockito	mockito-all	1.9.5	jar	The MIT License

Source: Own representation

B Command Line Interface

The command line options of the implementation are listed in Table 8.

C Train Path DAG Expansion

The standard formulation of multicommodity flow problems is to impose capacity constraints on the arcs instead of nodes (Barnhart et al., 2000). The following construction shows that a train path DAG can be expanded to a new DAG whose arcs have unit capacity, complying with the original setting. An example of such an expansion is shown in Figure 18: the nodes (apart from the source and target dummy nodes) correspond to train path slots or slot-slot-connections; on the other hand, the arcs have no real-world semantics in this representation.

Definition C.1. *Let $\mathcal{D}_k = (V_k, A_k)$ be train path dag for request k. The* ***expanded train path DAG*** *$\mathcal{E}(\mathcal{D}_k) = (\mathcal{E}(V_k), \mathcal{E}(A_k))$ defined as follows:*

- *Vertices $\mathcal{E}(V_k)$*
  - *$s_k, t_k \in \mathcal{E}(V_k)$*
  - *for $v \in V$: two vertices $v_{in}, v_{out} \in \mathcal{E}(V_k)$*
- *Directed edges $\mathcal{E}(A_k)$:*
  - *for $v \in V$: $(v_{in}, v_{out}) \in \mathcal{E}(A_k)$*
  - *for $(v, w) \in A_k$: $e \in \mathcal{E}(A_k)$ where*
    - *if $v, w \in V$: $e = (v_{out}, w_{in})$*
    - *if $v = s_k, w \in V$: $e = (s_k, w_{in})$*
    - *if $v \in V, w = t_k$: $e = (v_{out}, t_k)$* ♣

Example C.2 (Expanded Train Path DAG in Scenario I (without Earliness)). The expanded train path DAG of Figure 13(a) is shown in Figure 19. ♣

D Periodical Train Path Allocation with Periodicity

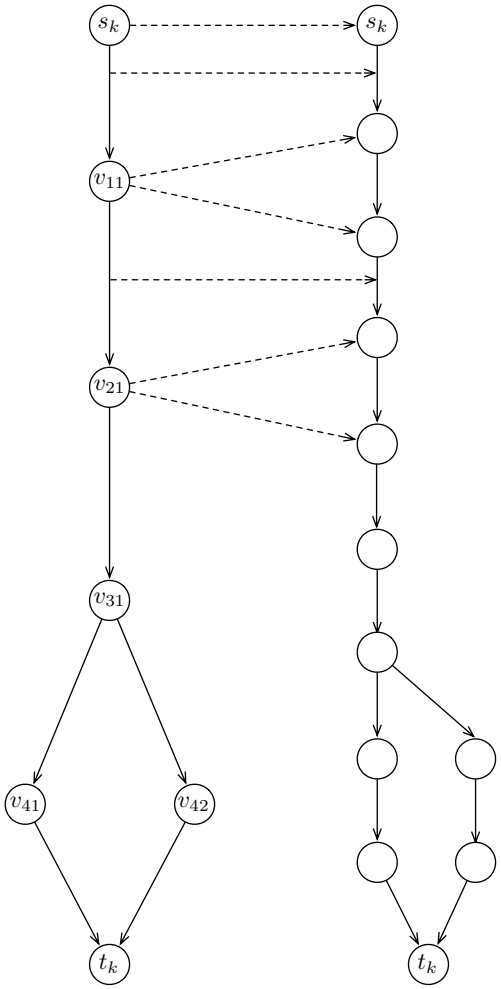
A train path slot typically operates on a specific pattern of days of service in a week, and a train path application may ask for a set of days of service. Such a periodicity is captured in the following definitions.

Table 8: Command Line Options

Command Line Option	Description
`-clean`	start with empty allocation
`-cleanoutput`	delete dthe output dir
`-confighelp`	show file format configuration options and terminate
`-file <file>`	input file
`-globalHardMaximumEarlierDeparture <nb>`	global hard maximum earlier departure (minutes) (default value: 0)
`-globalHardMaximumLaterArrival <nb>`	global hard maximum later arrival (minutes) (default value: 0)
`-globalHardMinimumDwellTime <nb>`	global hard minimum dwell time (default value: 0)
`-ignoreinfeasibleapps`	ignore infeasible train path applications and try to allocation feasible applications
`-help`	show this help message and terminate
`-output <file>`	output directory (default value: output)
`-pathbased`	use path-based model, default is arc-node model
`-properties <file>`	properties file (in ISO 8859-1 encoding)
`-requestfilter <pattern>`	Pattern of the form `01??000` (default value: `???????`)
`-showconfig`	show effective file format configuration
`-skipweboutput`	do not create html pages (may save time)

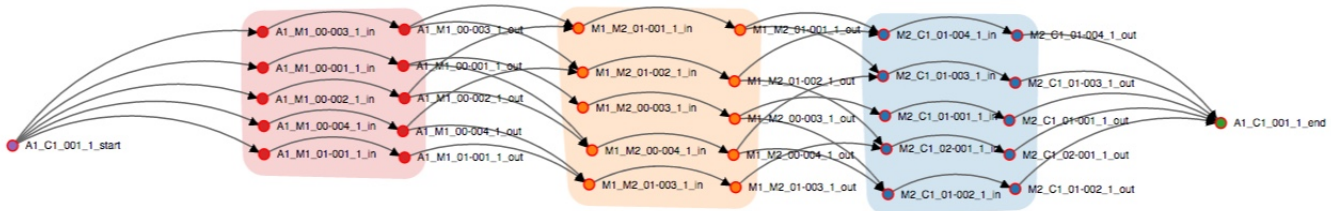
Source: Own representation

Figure 18: Illustration of Train Path DAG Expansion: dashed arrows show the expansion for some elements of the train path DAG.



Source: Own representation

Figure 19: Expanded Train Path DAGs for Scenario I (without Earliness).



Source: Own representation

Definition D.1 (Periodicity). *Let $T \in \mathbb{R}$ be a positive number and let $N \in \mathbb{N}$ be a positive integer. Let (T, N) be called* ***periodical time frame*** *representing the time frame $[0, \tau[$, $\tau = N \cdot T$. A set $\Pi \subseteq \{1, \ldots, N\}$ is called a* ***periodicity*** *in the periodical time frame (T, N)* ♣

Definition D.2 (Train Path Slot with Periodicity). *A* ***train path slot with periodicity*** *in a macroscopic railway topology $\mathcal{T} = (S, E)$, with respect to a periodical time frame (T, N) is a quintuple $v = (s_v, t_v, \delta_v, \alpha_v, \Pi_v)$ where*

- *$(s_v, t_v, \delta_v, \alpha_v)$ is a train path slot such that $\delta_v, \alpha_v \in [0, T[$,*
- *$\Pi_v \subseteq \{1, \ldots, N\}$ is a periodicity in (T, N).* ♣

In the same way, **train path applications with periodicity** can be defined.

Definition D.3 (Train Path Catalogue with Periodicities). *A* ***train path catalogue with periodicity*** *in a macroscopic railway topology $\mathcal{T} = (S, E)$ is a finite set V of train path slots with periodicity with respect to a common periodical time frame (T, N).* ♣

In this setting, three approaches are possible:

1. *flat allocation*: if different allocations on different days of service are acceptable, a simple train path slot can be introduced for all $|\Pi_v|$ days of service:

$$v_i = (s_v, t_v, [\delta_v + (i-1) \cdot T], [\alpha_v + (i-1) \cdot T \mod (N \cdot T)]), \quad \forall i \in \Pi_v, \tag{32}$$

   and the same rationale can be applied to train path applications with periodicity; then the model for the simple train path allocation case can be used as described in this thesis;
2. *penalise differences*: if different allocations on different days of service are acceptable but undesirable, then such combinations could be penalised in the objective function;
3. *hard constraints*: if different allocations on different days of service are not acceptable, the approach for simple train path allcation of this thesis has to be modified as sketched here:
   a) for all applications $k \in K$, construct the train path DAG for a prototypic day $i \in \Pi_k$ of service using a filtered catalogue $\tilde{V}^k = V_i^k \cup V_{i-1}^k \cup V_{i+1}^k$,
      - $V_i^k = \{v \in V : \Pi_v \subseteq \Pi_k\}$: V_i^k contains only those slots that are available on all requested days of service,
      - $V_{i-1}^k = \{v \in V : j - 1 \in \Pi_v \text{ for all } j \in \Pi_k\}$: V_{i-1}^k contains only those slots that are available on all days previous to a requested day of service,
      - $V_{i+1}^k = \{v \in V : j + 1 \in \Pi_v \text{ for all } j \in \Pi_k\}$: V_{i+1}^k contains only those slots that are available on all days following a requested day of service;
   b) replace v_i by v in the train path DAG;

c) introduce unit capacity and conflict constraints only for requests whose periodicity have one requested day of operation in common (i.e., for k, k' such that $\tilde{V}^k \cap \tilde{V}^{k'} \neq \emptyset$);

this approach ensures that the train path DAGs encode train paths that work on all requested days of operation and that the MIP works on the slots and applications with periodicity (the details ensuring that the same slot with periodicity can be used by several applications with disjoint periodicities are omitted here); this approach has to assume that train paths only use $\tilde{V}^k$, but it allows to extend the theory developed in this thesis to periodical train path allocation with periodicity.

Notice that periodicities are partially ordered by set inclusion. Therefore, a sequence of subproblems can be determined by topological sorting; such a hierarchical decomposition approach could allow to keep the mathematical optimisation problem tractable. Also notice that, in case of infeasibility, the periodicity could be split into two subsets, resulting in two independent applications (a semi-flat approach) and deviations between services could be penalised in the objective function. However, these ideas are not formalised nor implemented in this thesis and remain to be worked out in detail.