École polytechnique fédérale de Lausanne

Timetable Saturation in Practice with OR Methods

Matthias Hellwig

m.hellwig@sma-partner.com

optimising railways

sma

About SMA

- consulting and software company for railway systems
- established in Zurich in 1987, approximately 80 employees
- headquarters in Zurich, branch offices Lausanne, Frankfurt and Paris
- consulting portfolio

  Service offer, Production, Operations, Capacity, Demand and Franchises & Tenders
- software area
  timetabling tool Viriato and running-time calculator ZLR software systems, which support all aspects of railway system planning
- SMA works for train operators, infrastructure managers, public authorities and rolling stock manufacturers

About Matthias Hellwig

- master’s degree in computer science at TU Dortmund
- doctorate from the Humboldt University of Berlin in efficient algorithms
- software engineer before joining SMA in 2016
- since then Research Manager
  - responsible for the development and implementation of algorithms
  - the management of relationships with external research partners
  - PO for optimization interfaces

Agenda

- Introduction
  - Project Motivation and Intuitive Problem Setting
  - Rail Network Capacity / Infrastructure Model
  - The Problem Setting in the Project
- Algorithm
  - A Simplified MIP for Modelling Train Network Capacity
  - Overall Methodology
  - Performance Considerations
- Practical Aspects
  - Project Risks
  - Integration into a Software Tool
  - Testing / Bug Fixing
- Summary

Why do we need OR in Practice?

Management of customer, a European infrastructure manager, wants to know KPIs for evaluating their network capacity to make good strategic decisions.

“**Operations research** […] is a discipline that deals with the application of advanced analytical methods to help make better decisions.”

(from wikipedia.org / informs.org)

Management wants to answer questions like:

- Where do we have enough capacity in our network?
- Where do we need to build new tracks?
- Which parts of the network are affected by timetable changes?

Network Capacity

There isn't a unique definition accepted by all railway companies, but exist a lot of capacity notions.

- "Capacity as such does not exist. Railway infrastructure capacity depends on the way it is utilised."
- "A unique, true definition of capacity is impossible."

  (Both + figure from: UIC-406, 1st edition 2004)

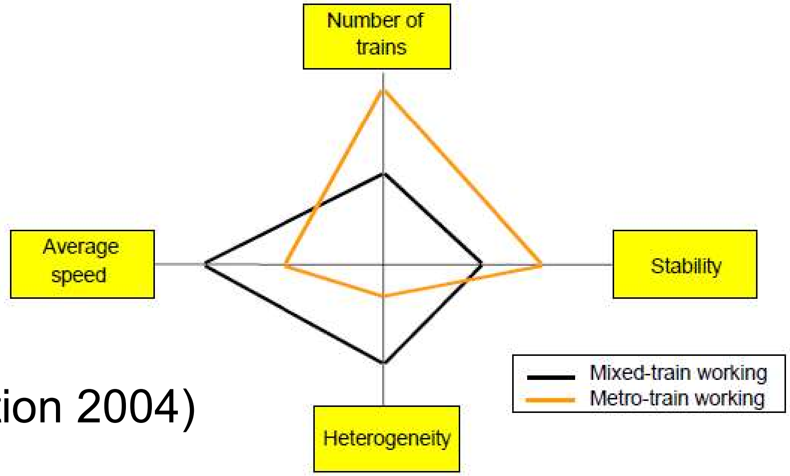


Fig. 1 - Capacity balance

Basic distinction in the literature:

existing infrastructure only vs. taking timetable into account

→ Which one(s) suit(s) best to the customer's need?

The Project

Consulting project:

find a capacity definitions suitable to the customer's needs.

Requirements:

- easy to understand
- should somehow relate to a timetable operable in principle
- as easy as possible to compute

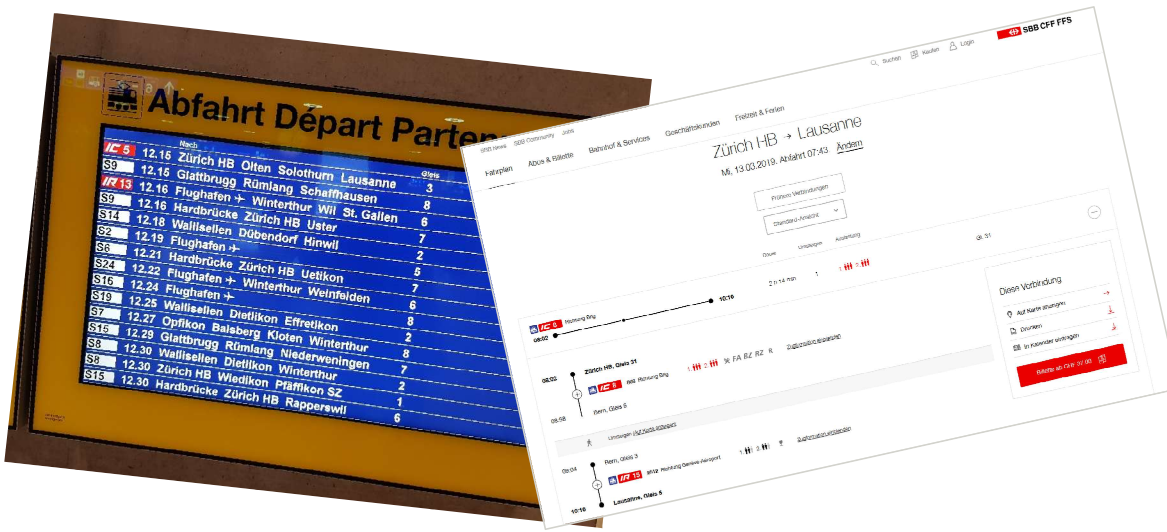
Software project:

- implement the definition(s)
- to calculate the residual network capacity automatically
- using methods from OR

today's topic

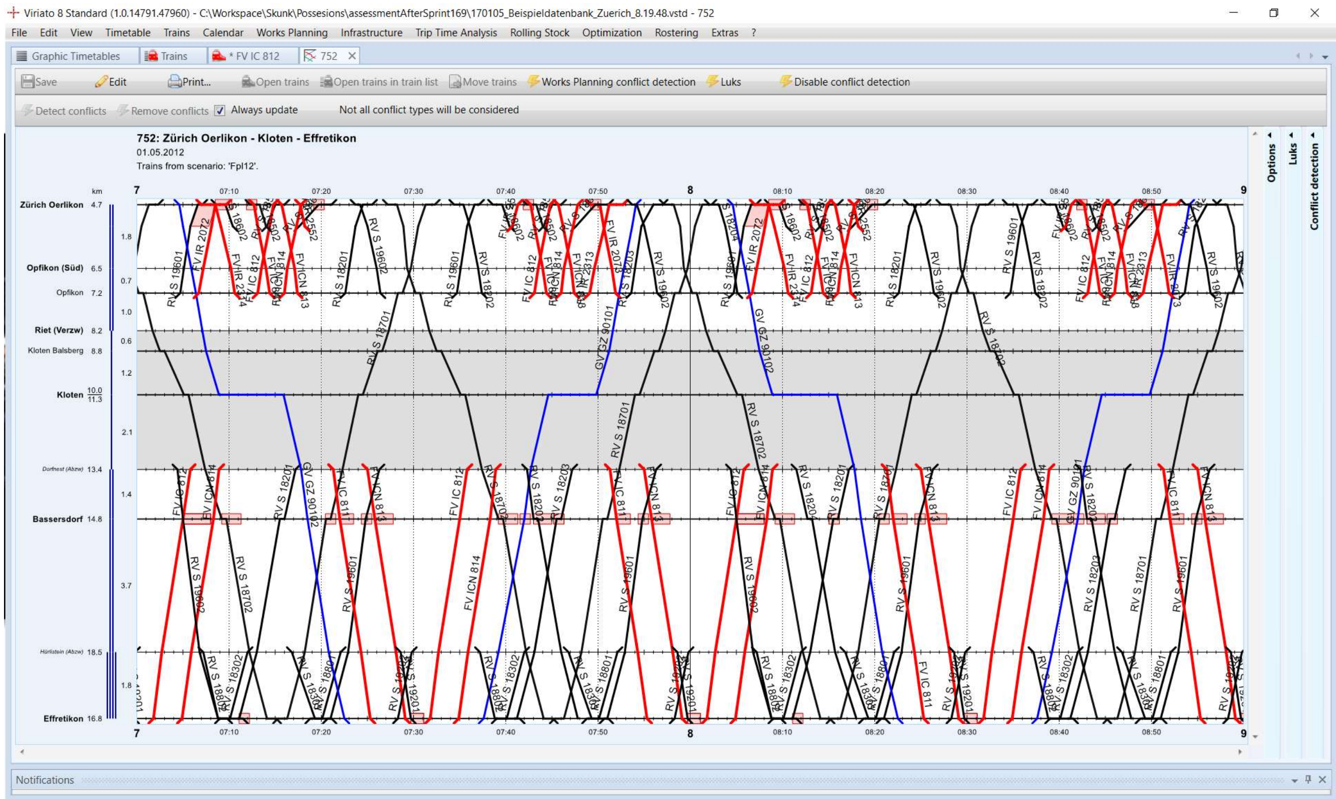
TYPES OF TIMETABLES

Timetables



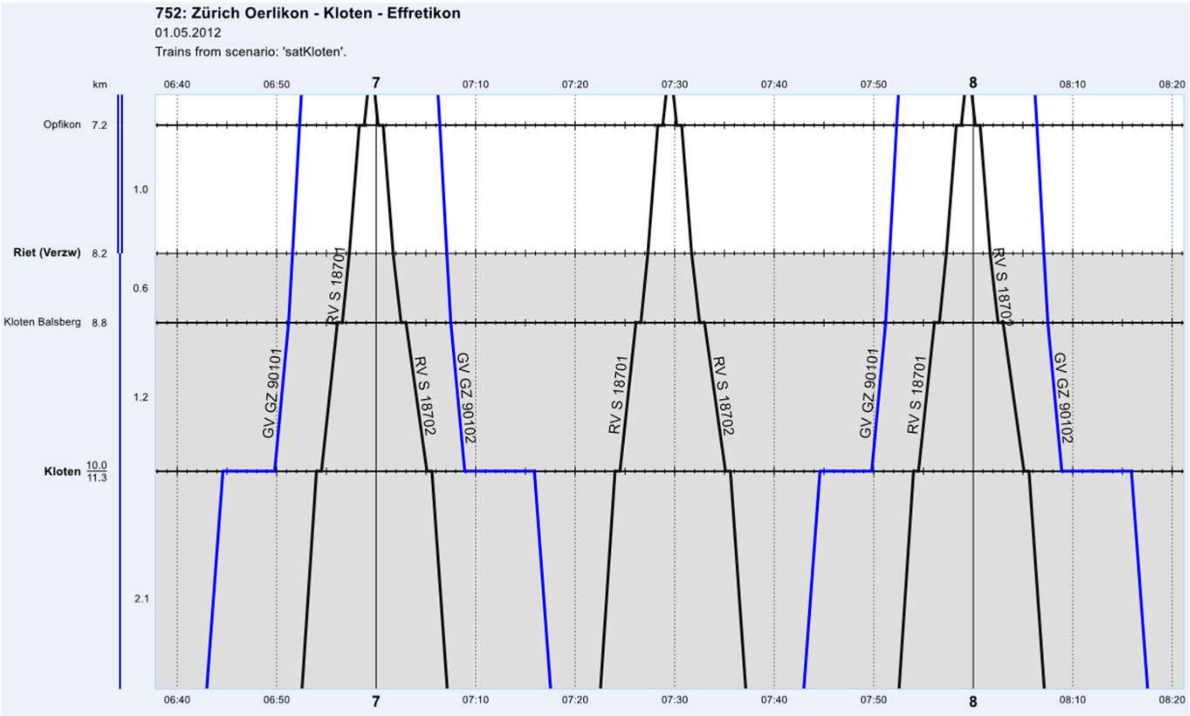
– there are different types of timetables

Timetables



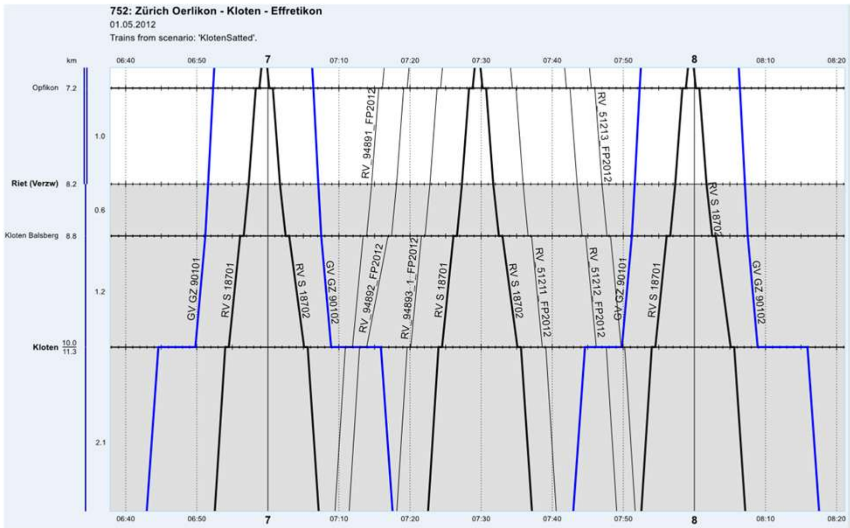
– During this talk

Intuitive Problem Setting



- want to determine the network's capacity
- How many commercially interesting train paths can we insert so as to saturate the given timetable?

Intuitive Problem Setting

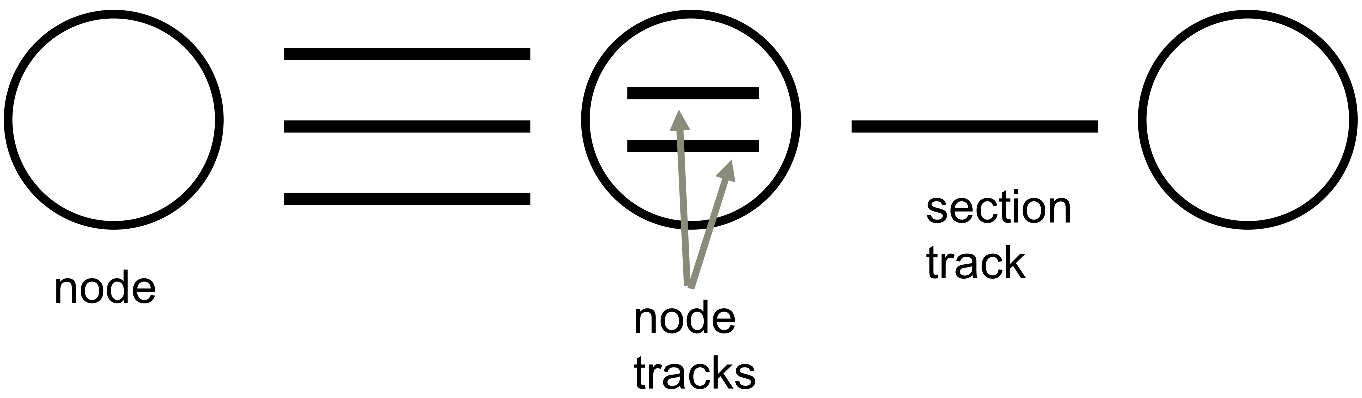


- What does it mean to saturate a timetable?
- What trains should be used for saturation?
- What is the infrastructure model?
- What are the constraints determining feasible solutions? (i.e. allowed timetables)

Train Network (I)

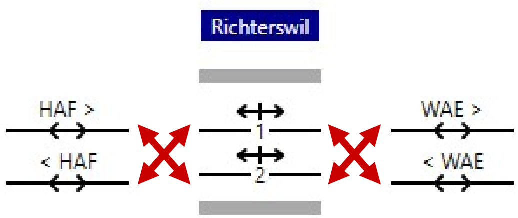
Simplified Mesoscopic Model (= almost macroscopic)

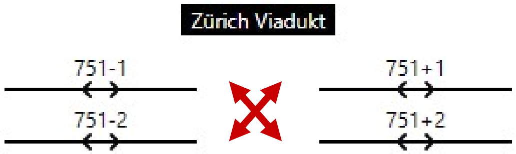
- we consider a network consisting of nodes and sections
- two adjacent nodes are connected with at least one section track
- nodes can have node tracks



Train Network (II)

Types of Nodes





Stations

– number of station tracks ≥ 1

– in all stations having at most 8 node tracks

– capacity of other stations ignored

– all routes possible 

– other station properties ignored

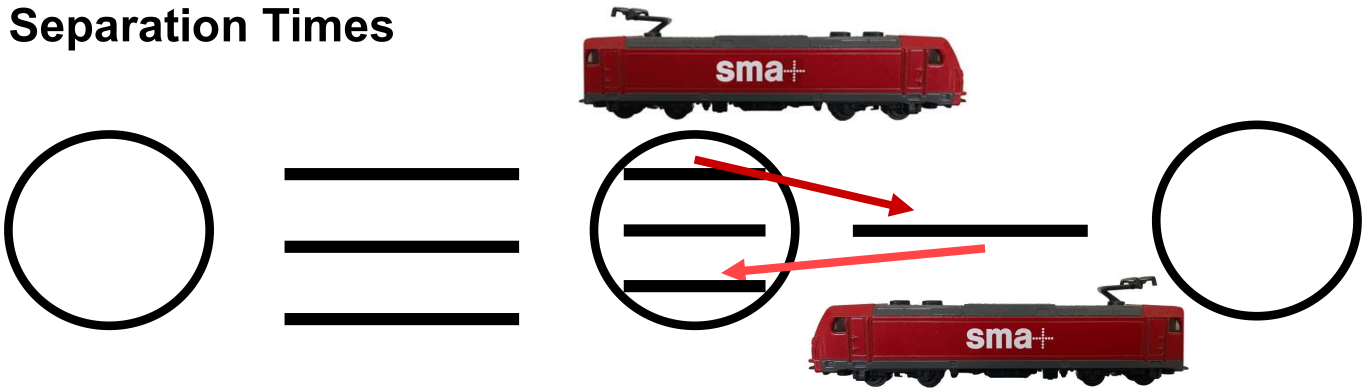
Junctions

– all routes through junctions assumed to be driveable

– assume trains have conflicts if and only if they use common section track in opposite direction

Conflict Model (I)

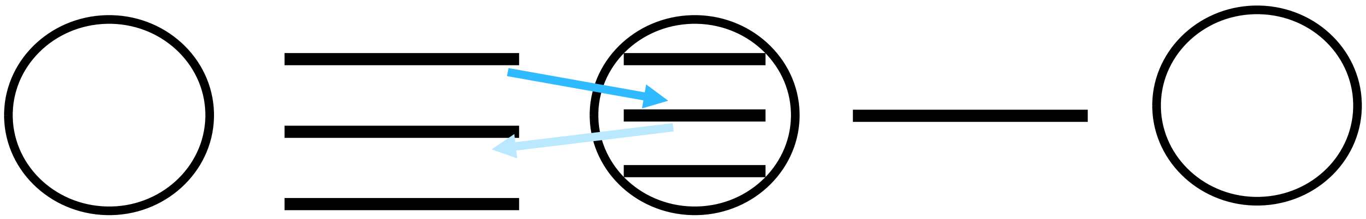
Separation Times



- two types of separation times
  - reoccupation time for reusing a section track in opposite direction (duration depends on if train stops or not) →
  - reoccupation time for a node track

Conflict Model (II)

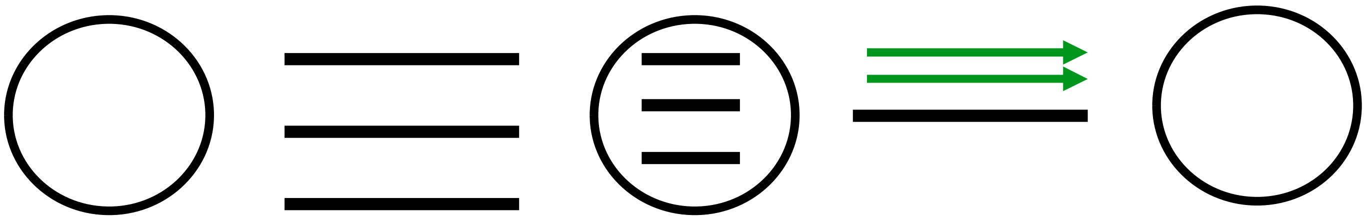
Separation Times



- two types of separation times
  - reoccupation time for reusing a section track in opposite direction (duration depends on if train stops or not)
  - reoccupation time for a node track 

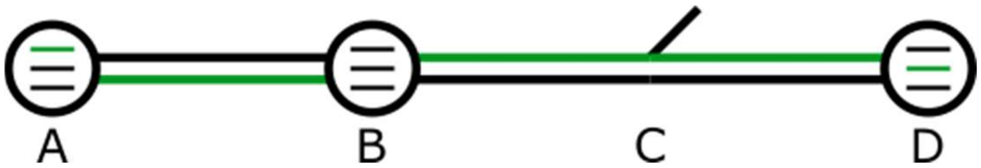
Conflict Model (III)

Headway Times



- headway times
  (depend on section track and on train types)

Trains Runs



Characteristics

- run along a sequence of nodes (“train path nodes”)
- travel on section / node tracks
- minimum running time between two adjacent train path nodes
- minimum stopping time at each train path node (dwell time)

Types of Trains In Problem

Trains from timetable ("**timetable trains**")

- arrival and departure times cannot be adapted (=> fixed)
- station tracks may be assigned

Trains for saturation ("**template trains**")

- start time to be chosen, can be delayed
- have minimum running and stopping times
- station track needs to be assigned

Both: No changes of train path node sequence or section tracks!

Insertion Sequence

Train insertion sequence models priorities:

- sequence of pairs $(T_1, n_1), \ldots, (T_k, n_k)$ with
- template trains $T_1, \ldots, T_k$
- n_i multiplicities

Train insertion sequence

$$\overbrace{T_1, \ldots, T_1}^{n_1}, \overbrace{T_2, \ldots, T_2}^{n_2}, \ldots, \overbrace{T_k, \ldots, T_k}^{n_k}$$

Precise Problem Setting

Input:

- train network with separation and headway times
- timetable (set of timetable trains)
- sequence of template trains
- time window

Goal:

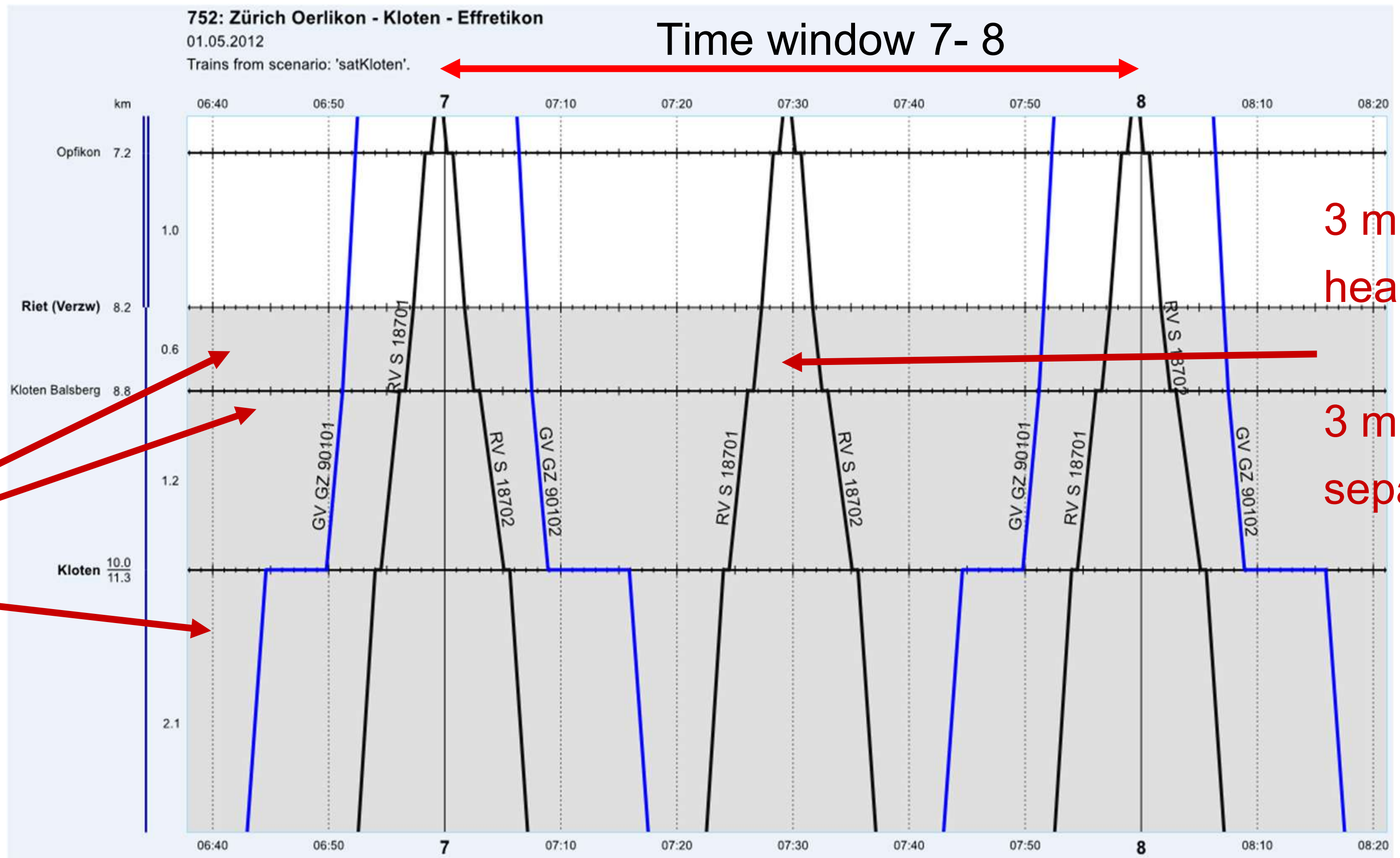
Maximize the number of trains inserted *feasibly* into the timetable in the given time window according to the template train sequence.

Feasibility: Respecting separation times and headway times

Priorities: Can happen we cannot insert any T_1 but template trains $T_i, i > 1$

INPUT

Problem Setting

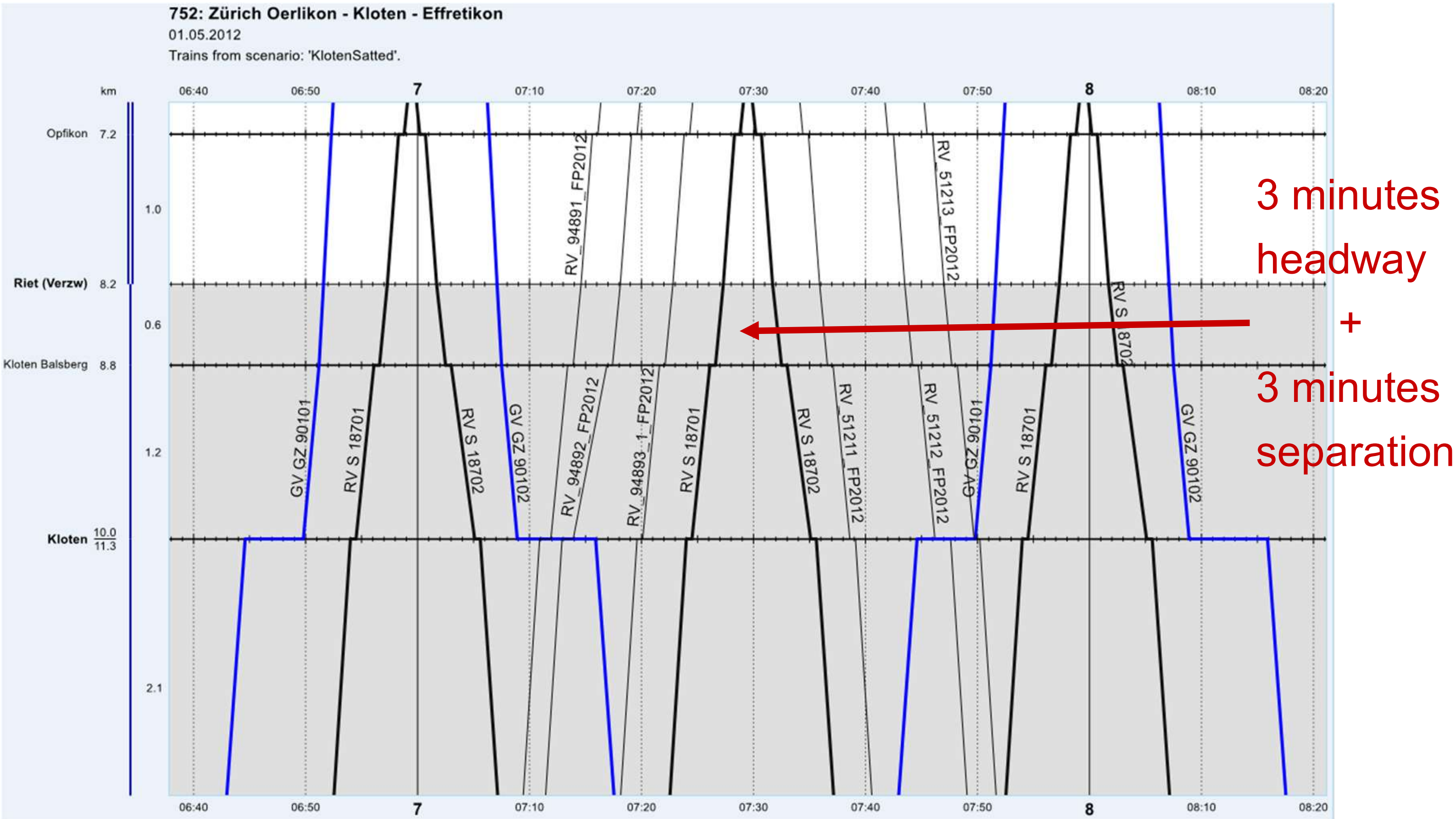


3 minutes headway
+
3 minutes separation

Single
Track section

SOLUTION

Problem Setting



Agenda

- Introduction
  - Project Motivation and Intuitive Problem Setting
  - Rail Network Capacity / Infrastructure Model
  - The Problem Setting in the Project
- Algorithm
  - A Simplified MIP for Modelling Train Network Capacity
  - Overall Methodology
  - Performance Considerations
- Practical Aspects
  - Project Risks
  - Integration into a Software Tool
  - Testing / Bug Fixing
- Summary

Modelling a Timetable as A MILP

Zurich	dep 8:00
Zurich Viaduct	arr 8:01.1 / dep 8:01.1
Zurich Wipkingen	arr 8:01.8 / dep 8:02.3
Zurich Oerlikon	arr 8:05.2 / …
…	
Schaffhausen	arr 8:44.0

Events: arrivals or departures

- basic ideas:
  - model arrival and departure events as variables
  - the variable value indicates the time when respective event takes place

Modelling a Timetable as A MILP

Zurich	dep 8:00
Zurich Viaduct	arr 8:01.1 / dep 8:01.1
Zurich Wipkingen	arr 8:01.8 / dep 8:02.3
Zurich Oerlikon	arr 8:05.2 / …
…	
Schaffhausen	arr 8:44.0

Activities:

Driving or stopping

– basic ideas:
  – model activities as relations (inequalities or equalities)
  – impose restrictions on the event times

How to Model Times in a MILP?

- available variable types of a MIP solver are from $\mathbb{B}, \mathbb{Z}, \mathbb{N}, \mathbb{Q}$
- running times of solver (=time to solve a problem) depend on
  - the size of coefficients in constraint matrix
  - types of problem variables
  - (and much more) …

Any ideas?

How to Model Times in a MILP?

0 E

Model Start Time
(e.g. 12:35)

Model End Time
(e.g 14:20)

Arr 12:45 → ???

- All event times modelled as rational variables:
- 12:45 is 10 mins after model start time
- precision? In our case: 6 s (time granularity in Viriato)
- therefore 12:45 → 100 (in model time)
- time window size [12:35, 14:20] = 105 mins → E = 1050

Running and Stopping Times

for all trains t and all train path nodes n

Stopping Time Equation

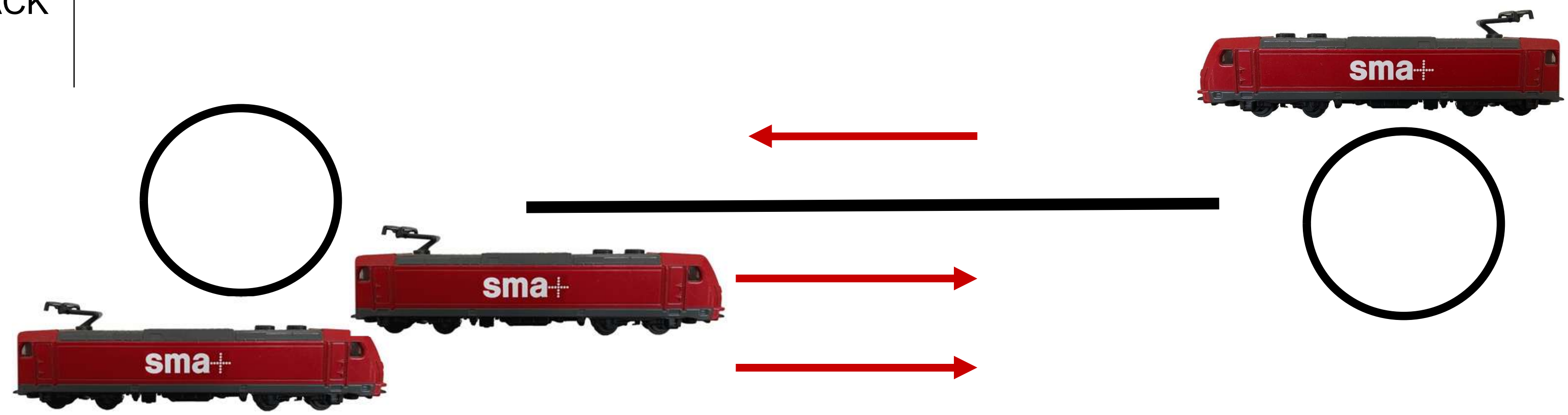
$$dep_{n,t} = arr_{n,t} + \overbrace{minstop_{n,t}}^{constant!} + \ addstop_{n,t}$$

Running Time Equation

$$arr_{n+1,t} = dep_{n,t} + \overbrace{minrun_{n+1,t}}^{constant!} + \ addrun_{n+1,t}$$

addrun / addstop are decision variables for template trains, and constant for timetable trains

Modelling Section Track Capacity

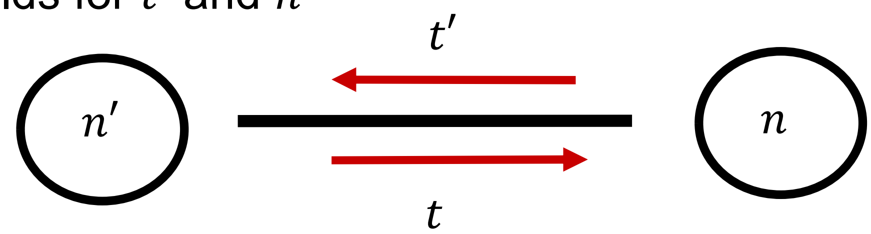


Basic requirements to trains if travelling on same section track:

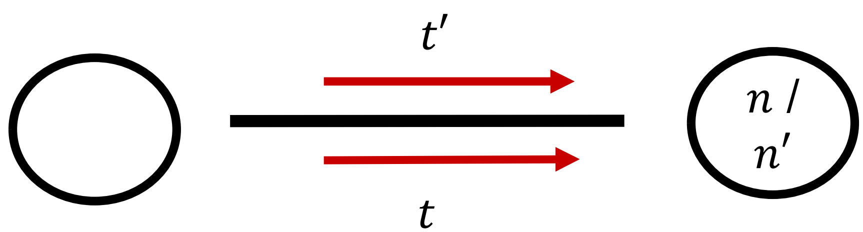
- trains cannot overtake and not cross on a section track (otherwise there would be a node somewhere)
- can be travelling into the same direction or in opposite directions
- if travelling into same direction: headway time should apply
- if travelling into opposite direction: separation time should apply

Definition of Ordering Variables

- assume train t travels on $track$ before it arrives at train path node n
- and same holds for t' and n'
- Case 1:



- Case 2:

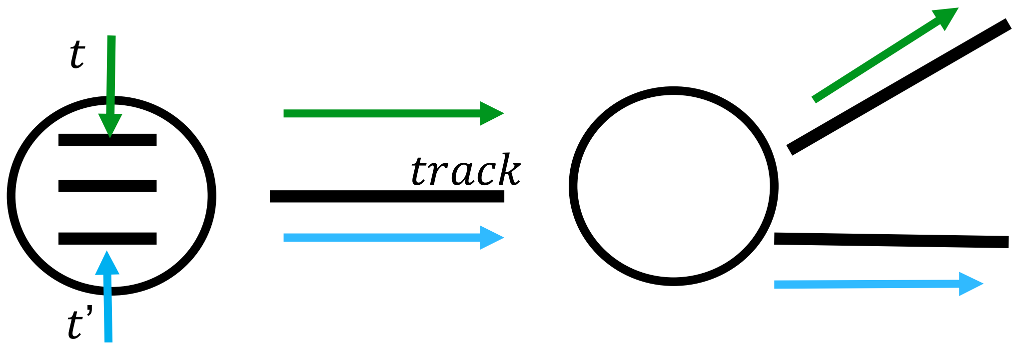


- let $ord_{t,n,t',n'}$ be an indicator variable saying that train t travels before train t' on $track$

For sake of simplicity of notation in the following:

$$ord_{t,t',track}$$

Headway Times (Basic Idea)

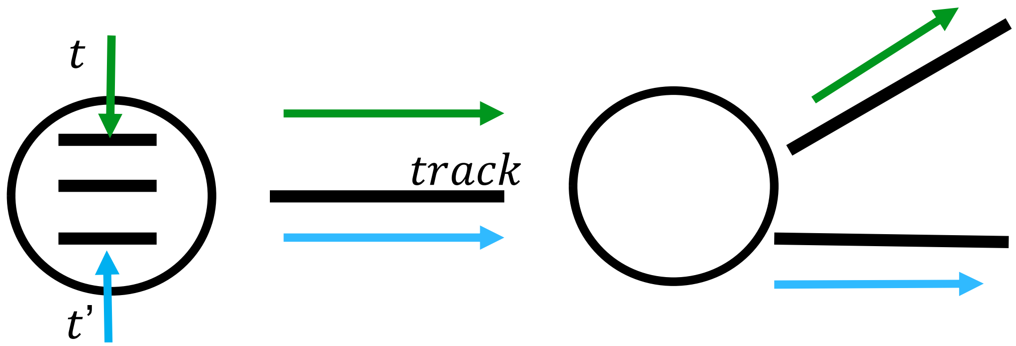


– for all $track$ visited by both trains on their respective path same direction

$$arr_{t,n} + \overbrace{hwy_{track,t,t'}}^{constant!} \leq arr_{t',n'}$$

– if train t' travels not before train t (i.e. t travels before t') on $track$ then the arrival time of train t' at its next train path node is at least the arrival time of train t at its next node plus headway time

Headway Times



– for all $track$ visited by both trains on their respective path same direction

$$arr_{t,n} + \overbrace{hwy_{track,t,t'}}^{constant!} \leq arr_{t',n'} + \mathrm{M}\, ord_{t't,track}$$

– if and only if train t' travels not before train t (i.e. t travels before t') on $track$ then the arrival time of train t' at its next train path node is at least the arrival time of train t at its next node plus headway time

M is a sufficiently large constant (which M suffices?)

Headway Times

- Analogously: for departure events

$$dep_{t,n} + \overbrace{hwy_{track,t,t'}}^{constant!} \leq dep_{t',n'} + \mathrm{M}\, ord_{t',t,track}$$

→ Add analogous inequalities for the reverse order of t and t', i.e. same inequalities with t and t' interchanged

- Clearly, want to add $ord_{t,t',track} = 1 - ord_{t',t,track}$, too

Travelling in opposite direction:

- separate departure and arrival events of two trains at same section track with a separation time where the exact separation time value depends on whether the preceding train has a stop or not

Node Track Selection

- If node track capacities are considered then the algorithm should determine the track a train occupies

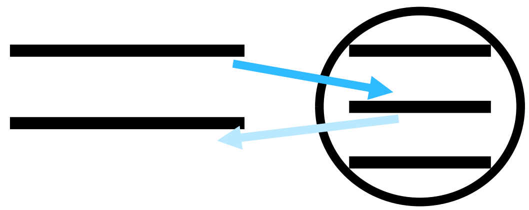
Node Track Selection

$$\sum occ_{n,t,track} = 1$$

with binary variables $occ_{n,t,track}$

Modelling Station Track Capacity

– A separation time between t and t' applies if both trains use the same node track



– If t' is on $track$ before t, then the arrival time of t is at least the departure time of t' plus the separation time

Separation Time for Node Tracks

$$dep_{t',n} + sep_{t',\mathrm{t}} \leq arr_{t,n} + M\,(1 - (occ_{n,t,track} \land occ_{n',t',track})) + M(1 - ord_{t't,track})$$

Objective

Simply the sum of the travel times of the trains

$$min \sum (dep_{t,n(t)} - arr_{t,0})$$

because that «looks good»!

Algorithm

$output = \emptyset$

foreach $train$ in $\overbrace{T_1, \dots, T_1}^{n_1}, \overbrace{T_2, \dots, T_2}^{n_2}, \dots, \overbrace{T_k, \dots, T_k}^{n_k}$

try to insert all trains $output \cup \{train\}$ at same time feasibly into the given timetable using MILP

if insertion successful $output := output \cup \{train\}$

(else $output$ remains unchanged)

return $output$

PERFORMANCE CONSIDERATIONS

Make it Work in Practice

- have (almost) all building blocks
- but: we need to make a working software
- instance sizes? Running time of the algorithm?
- ~8000 trains travelling per day in the Belgian network

  → won't work (see next slide)

Ideas to make it work?

Make it Work in Practice

- observation in practice: there are a lot of variables and constraints in our model for practical problem instances

  (~300 template trains ~ 14 mio. variables)
- already the model creation costs too much time (minutes per one MIP)

Therefore overall idea:

- do not model aspects (hopefully) not relevant for practice

→ saves **for sure** model generation and solution processing time

→ saves **potentially** model solving time (here it does)

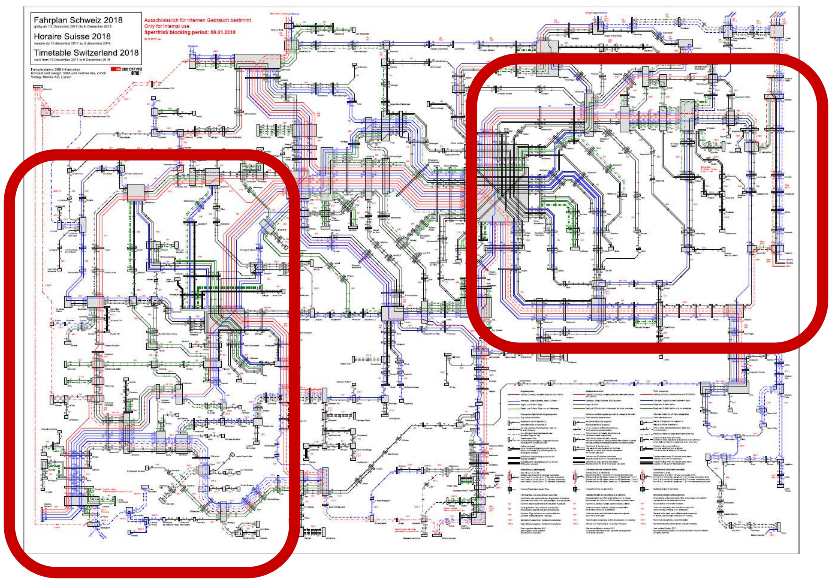
→ What has the largest impact?

Make it Work in Practice

Measures for Reducing the Overall Solution Time:

- cut out a relevant piece of the true timetable (→ next slide)
- no capacity of large nodes
  (induces too many variables and constrains,
  i.e. between all train pairs $t_1 \rightarrow t_2,\ t_1 \rightarrow t_3, t_2 \rightarrow t_3, \ldots$)
- Do we need an optimal solution in each step w.r.t. our objective?
  → no. Set large MIP-Gap for all but last iteration of main loop
- sometimes a solver fails to prove the infeasibility if no capacity left
  → set large solver timeout and assume infeasibility if exceeded

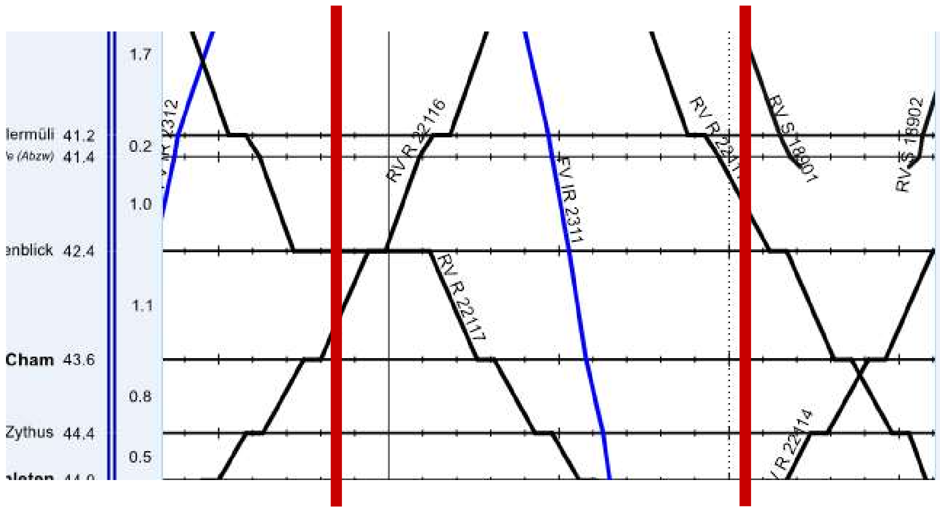
Timetable Cutting (I)



- can we decompose the network geographically?
- yes, depending on actual the instance (definition of template trains)
- typically a network can be decomposed into subnetworks
- solve the saturation problem independently in each part

  → Divide & Conquer

Timetable Cutting (II)



– the customer is happy if we are able to saturate the network during the morning rush hour
→ cut the considered time window and fix «boundary conditions»

In our case:

→ truncate minimum-running and -stopping times

→ fix order

POSSIBLE EXTENSIONS

Outlook

The model is quite simple, a lot of aspects missing as there are

- unplanned stops of request trains
- node capacities only partially considered
- routes through junctions / into (and out of) stations
- and respective separation times
- consideration of possessions

Agenda

- Introduction
  - Project Motivation and Intuitive Problem Setting
  - Rail Network Capacity / Infrastructure Model
  - The Problem Setting in the Project
- Algorithm
  - A Simplified MIP for Modelling Train Network Capacity
  - Overall Methodology
  - Performance Considerations
- Practical Aspects
  - Project Risks
  - Integration into a Software Tool
  - Testing / Bug Fixing
- Summary

sma

Project Risks

- Main project risks:
  - Does the customer accept the solution method?
  - Does the methodology scale sufficiently?

  → Can we provide the project on time and within budget?

- Solution Approaches
  - communication

    → clarify expectations and agree on a solution method before
  - prototying

    → estimate scalability of method and assess solution quality

    → pre-project?

What's more to a real-life project?

- w.r.t. customer (end user)
  - easy to define input data («usable tool»)
  - visualize results of the algorithms («understandable output»)
  - minimum protection against accidental misuse («robust tool»)
  - data shouldn't be lost if something goes wrong («stable tool»)

What's more to a real-life project?

- w.r.t. algorithm developer
  - different focuses of data types : GUI ≠ Algorithm ≠ DB
  - wants high data quality:
    complete, not self-contradictory or non-causal

    e.g. rounding errors leads to infeasible problems

$$arr_{n+1,t} = dep_{n,t} + minrun_{n,t}$$
$$100 \neq 58.49 + 41.52$$

→ conversion of data types and data needed: GUI ⇔ Algorithm

→ solution concept: SMA's Algorithm Platform

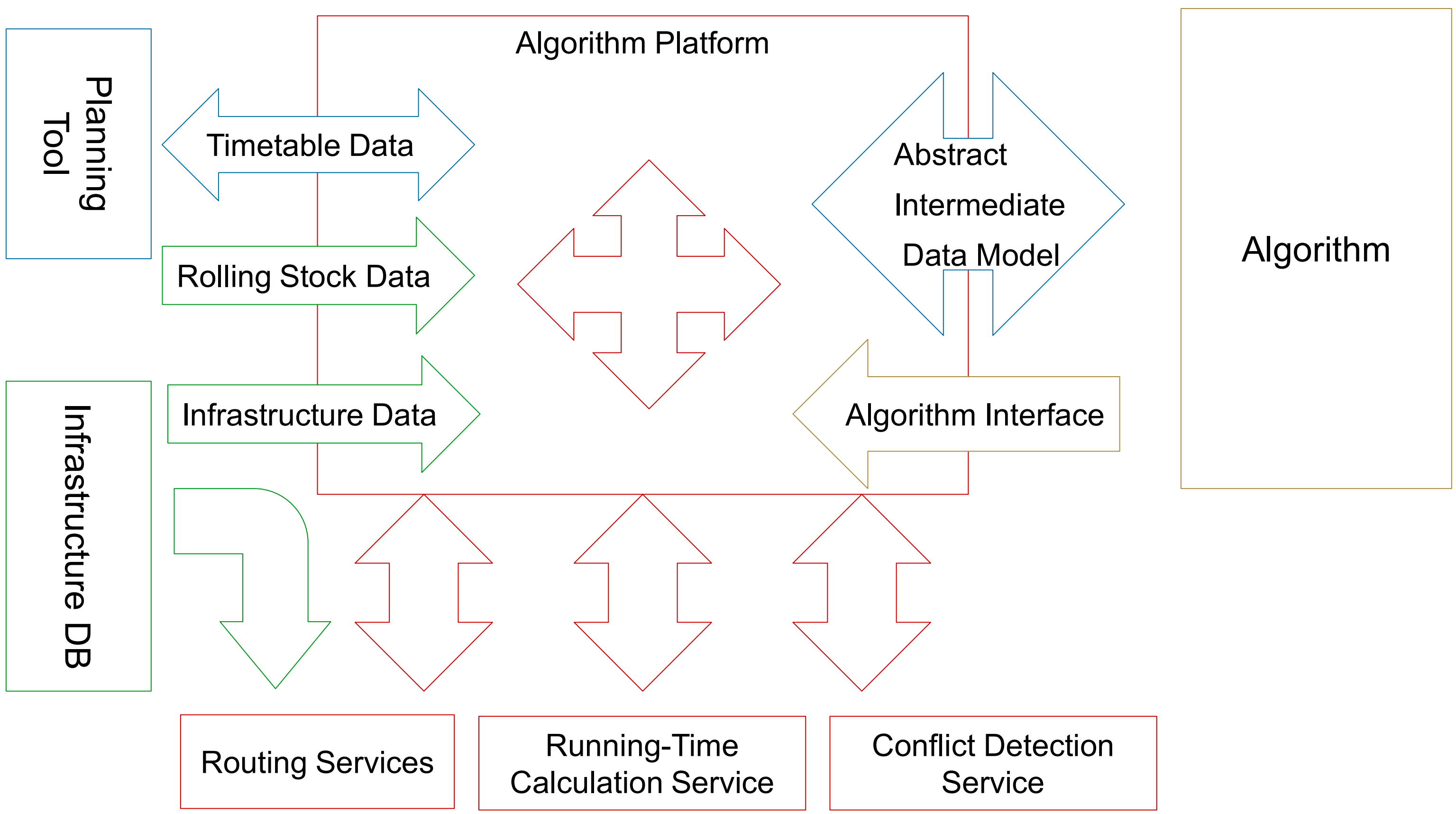
Algorithm Platform

- opens our timetabling tool Viriato for external algorithms

- provides a usable GUI for end users
  - user has a robust and rich tool to provide input data (timetables / template trains / network data)
  - user can visualize the results

- offers an algorithmic interface for an algorithm developer
  - data types tailored to algorithmic use cases
  - suitable for railway problems
  - algorithm developer can really focus on the algorithmic work (not developing the GUI / data type conversions)

Interfaces


Planning Tool
Algorithm Platform
Timetable Data
Rolling Stock Data
Abstract Intermediate Data Model
Algorithm
Infrastructure DB
Infrastructure Data
Algorithm Interface
Routing Services
Running-Time Calculation Service
Conflict Detection Service

Software Development Aspects

- development in a team: high code quality desirable
  → reviews makes code maintainable by a team
  → software should be extensible
  → challenge: deep OR know-how not widely spread in CS
- software testing
  - ensure working software (correctness and stability)
  - testing protects us from regressions
    → basic properties still work when model extended
- solver software has to be integrated (if applicable)
  → domain model has to be transformed into the mathematical model
  → needs abstraction and clear architectural structure

PRACTICAL ASPECTS

Implementing Optimization Algorithms

- MIP models are quite error-prone, i.e. sensitive to errors in input data recall rounding error problem

$$arr_{n+1,t} = dep_{n,t} \quad + minrun_{n,t}$$
$$100 \qquad \neq \quad 58.49 \; + \; 41.52$$

- Test the model aspects / properties rather than the single equation/inequality generation

  → Tests interaction of the different types of constraints «integration test» alike

- Example: How to test that our conflict model is working?

Example

Aspect: Headway Times

«Feasibility test»

- If two trains t, t' use the same a section track there is a solution of the model if $dep_t \geq dep_{t'} + hwy_{t,t'}$
  - generate the model for two trains
  - fix in addition $dep_t = dep_{t'} + hwy_{t,t'}$
  - verify there is a feasible solution

→ ensures that our model is solvable if trains have enough headway time "a feasible solution can be found"

→ should be tight to be a sensitive test

Example

Aspect: Headway Times

«Infeasibility test»

- generate the model for two trains
- fix $dep_t = dep_{t'} + hwy_{t,t'} - 0.01$
- verify there is no feasible solution

→ ensures model is infeasible if headway time is violated
"the headway constraint cannot be violated"

Both tests together ensure correctness of the model for the given aspect (of course limited to the test case)

Implementing Optimization Algorithms

Take care of the scaling of the numbers in you model!

– keep variable values as small as possible dep = 100 vs. dep = 10000
  → model solving time depends on the length of numbers
  → smallest unit in our problem is 1 (a tenth of a minute)
  → largest number: M ≅ maximum headway time plus duration of considered time window in tenth of minutes

– watch out for numerical problems
  (e.g. rounding errors, fractional numbers, quotient of largest / smallest number)
  → in our model definition: all times are integral by definition
  → typically: $M/1 \leq 2400$

Debugging Optimization Algorithms

Our algorithm fails to produce a solution or produces an obviously strange looking solution. What can we do?

IIS computation (infeasible irreducible subsystem)

– Practical instances are quite large, probably no one can debug a MIP with more than 10.000 constraints

→ make model smaller by isolating problem

In our case:

→ create same problem with less trains, e.g.
restrict size of considered time window (smaller saturation problem)

→ add artificial constraints that you expect to hold and reduce the degree of freedom (and thereby the MIP model size) to isolate the problem

Example

Could be returned by IIS computation

$$dep_{0,t} = arr_{0,t} + 10, arr_{1,t} = dep_{0,t} + 35$$

$$dep_{1,t} = arr_{1,t} + 10, arr_{2,t} = dep_{1,t} + 55$$

$$arr_{2,t} = 110, arr_{0,t} = 15$$

What is wrong?

Suppose we know (by inspecting input timetable) that there should hold:
$dep_{0,t}$ = 15

Adding constraint $dep_{0,t}$ = 15 to model and recalculating IIS yields:

$$arr_{0,t} = 15, dep_{0,t} = arr_{0,t} + 10, dep_{0,t} = 15$$

Example

$$arr_{0,t} = 15, dep_{0,t} = arr_{0,t} + 10, dep_{0,t} = 15$$

→ identified problem might be:
time $arr_{0,t} = 15$ in boundary condition wrong, therefore needs to be $arr_{0,t} = 5$

We see:

- made MIP smaller by adding additional domain knowledge
- identification of true error cause by domain knowledge
  (reason could have been $dep_{0,t} = arr_{0,t}$, i.e. wrong minstop)

→ need to have insight into the problem domain to understand bugs, too

Agenda

- Introduction
  - Project Motivation and Intuitive Problem Setting
  - Rail Network Capacity / Infrastructure Model
  - The Problem Setting in the Project
- Algorithm
  - A Simplified MIP for Modelling Train Network Capacity
  - Overall Methodology
  - Performance Considerations
- Practical Aspects
  - Project Risks
  - Integration into a Software Tool
  - Testing / Bug Fixing
- Summary

Summary

Industrial projects - also and in particular those in which OR methods play a role - consist of

- requirements engineering
- suggesting and agreeing about a potential solution method
- making simplifications / assumptions where necessary / adequate

to define the problem so that it is solvable in practice.

Benchmark your model on real instances!

A customer wants a usable and stable software at a reasonable price.

In general: A customer wants to have a working product in the first place, not a beautiful, novel and sophisticated solution method.

Summary

Mathematical models are quite prone to data errors. Important to test and clearly structure the code and to find good model formulations. Debugging can be a challenge in real models. Models should be as simple as possible as to be implementable with reasonable effort. Take extensibility into account!

Besides mathematical skills and domain know-how one needs:

- strong software development skills
- strong communication skills

to exchange with non-OR-experts: customers / software developers

Conclusion: Development and implementation of algorithms is an important part, but only a part of a real project.

Contact

SMA and Partners Ltd.

Gubelstrasse 28

8050 Zurich

Switzerland

Phone +41 44 317 50 60

info@sma-partner.com

www.sma-partner.com