Calculation of Realistic Railway Station Capacity by Platforming Feasibility Checks

Peter Sels *
CIM/T&I, K. U. Leuven, Belgium

Bertrand Waquet
ICTRA, B-Holding, Belgium

Thijs Dewilde
CIM/T&I, K. U. Leuven, Belgium

Dirk Cattrysse
CIM/T&I, K. U. Leuven, Belgium

Pieter Vansteenwegen
Department of Industrial Management, Ghent University, Belgium

ABSTRACT

A theoretically calculated station capacity very often appears to be based on an unrealistic train set. By considering the actual, current, as well as future train set scenarios with increasing traffic, we determine the realistic station capacity of the railway stations in Belgium. Our module is able to deal with real-life station and train sets in a few minutes and is now fully integrated in the software of Infrabel, the Belgian Railway Infrastructure Management Company.

Keywords: *Train Station Capacity, Platforming, Mixed Integer Linear Programming*

INTRODUCTION

Infrastructure Management Companies need to be able to check if their stations can manage the increasing train traffic. They do this by looking at the proposed timetable and checking whether all trains can be routed to and from a platform within the times dictated by the timetable. Note that this capacity check is done after the time table construction. Therefore, we suppose in this research that the platform times are fixed.

Up to now, at Infrabel, this check was done manually, without use of optimization tools. Because of the combinatorial complexity of this problem, for a human, it is a very hard or even impossible job to perform without errors, let alone optimally. Cases of wrongly considering two routes as being in conflict as well as cases of wrongly considering two routes as not being in conflict have been noted. This resulted in respectively under- and overestimation of capacity in these cases. To more correctly estimate station capacity, the presented research and development was carried out.

In the first section we will present the platforming optimization problem. In the second section we will present some model aspects and in the last, we will draw some conclusions and hint at some further work.

*Corresponding author - Logically Yours BVBA, Plankenbergstraat 112 bus L7, Belgium, +32 486 95 67 97, sels.peter@gmail.com

PROBLEM AND MODEL

Literature Review

We refer to Caprara et al. [1] and [2] for an extensive respectively short Train Platforming Problem (TPP) research literature survey. Contrary to some other TPP research, as done by Zwaneveld et al. [3, 4] and Kroon et al.[5], in our problem, where the timetable is given, we assume the *platform times* (train arrival and train departure time) of both train sets to be *fixed*. We also assume that there is at most one routing between each line and platform. This is almost the same problem as considered by De Luca Cardillo and Mione [6] and Billonnet [7], so we compare our results with theirs.

Infrastructure Modeling

A station will be represented by a graph with edges and vertices. Train stations typically have from 1 to around 20 platforms. These are in general parallel. These platforms form the center of the station. On both platform ends, we have some straight tracks, called lines in this abstract, that continue straightly from or to the platforms. Other tracks cross these straight lines. The collection of these lines and tracks is called the routing grid. We have a routing grid on each side of the station. The goal of these routing grids is to be able to connect from the platforms to a set of lines going to or coming from other stations. The crossing tracks allow for a train coming from one station to end up on more than one possible platform in the next station. This redundancy in routings allows more flexibility in assigning trains to platforms, which should increase practical station routing capacity. Of course they also introduce restrictions, since crossing routings, also called dependent routings, cannot have two trains on them at the same time. Moreover, a safety buffer needs to be inserted between them.

So, as for infrastructure, we define a set P of platforms p, a set R of routings r, their relational mapping $r2p : R \to P : r \mapsto p = r2p_r$ and a set L of lines l.

Traffic Modeling

Next we have the train traffic which we can structure by inspecting the cases that occur.

a train can arrive from one side of the station, stop at a certain platform and leave to the other side of the station;

a train can enter and leave the station at the same side;

two trains can be merged in a station which means the two trains enter from a different routing (either on the same or a different side) and have to end up on the same platform to be coupled and leave the station at any side;

trains can be combined from more than two trains;

A train entering or leaving a station through a routing and a platform is called a movement. Each movement happens on a particular routing. An occupation is a collection of movements that share the same platform. Each movement will have a different routing, but all the movements within one occupation will pass the same platform. So each movement is associated with a routing and each occupation with a platform.

So we can now define a set O of occupations. Each occupation $o \in O$ has a set M_o of movements m. Each movement m is of type IN or OUT. We define $M_{IN} \subset M$ as the set of IN movements and $M_{OUT} \subset M$ as the set of OUT movements. It follows that $M_{IN} \cup M_{OUT} = M$. A movement has a single constant time $t_{o,m}$. For an IN movement, this is the time the middle of the train arrives at the middle of the platform, according to the planning. For a train that stops at the current station, this is the time its doors can open and the train is ready to let passenger go and receive new ones. For a train that passes the current station, this time is just the time its middle point passes the middle of the platform. For an OUT movement, in the case of a stopping train, it is the departing time. An occupation will always have at least one IN and at least one OUT movement. For a passing train the times of the IN and OUT movements are the same. For a stopping train the OUT movement time will be larger than the IN movement time, by the dwell duration, which is typically a minute or two. Each movement also needs to respect a fixed, known line where its train enters or leaves the station. Optionally, for any movement, platform and/or routings can be fixed. The task of the optimizer module is to choose platform and routing wisely so that no train pair conflicts arise.

Feasibility

The allocation problem can now be defined as trying to find an assignment for every movement to one routing and for each occupation to a platform.

Of course the routings that are chosen for a movement must connect to the platform chosen for the occupation the movement belongs to.

A further constraint will be that, if we call the *routings* and the *platforms* the *resources*, no single resource can have more than one allocation to it at any time.

On top of this, some routings are dependent, which means that they have a sub-resource, like a switch or other routing part in common. We will also require that between two dependent routings, only one can be used at any time.

Optimality

Next to the real platforms and routings, we introduced a fictive platform and associated fictive routings to allow the model to always be feasible. This fictive platform will have all trains assigned to it that cannot be assigned to real platforms. Now the problem, technically, becomes an optimization instead of a feasibility problem.

To define optimality, we construct a cost function. It consists of two types of cost terms. The first corresponds to a cost for changing a platform for already planned trains and its allocations. The second type is associated with the concept of fictive platform and associated fictive routings. These are introduced to handle all traffic that cannot be mapped on the existing platforms or routings. These fictive resources can only be used at high cost. The total cost function will be minimized.

If keeping the already planned trains on the current platforms is no longer a goal, planning becomes easier. This will be especially beneficial if the original planning was not yet done with station capacity optimization in mind. The user of the optimizer can control its behavior in this sense.

Time Definitions

Figure 1 shows a train passing its IN-routing, platform and OUT-routing. The time constants and variables indicated on it will be used in the model.

The times a train has to arrive and depart on a platform p are given as $t_{p,arr}$ and $t_{p,dep}$ in the occupations IN and OUT movements respectively.

We follow the train in Figure 1 in increasing time and space dimensions, as it goes from the IN routing over the platform to the OUT routing. The top half of this figure shows two curved lines. The leftmost curve describes the position of the head of the train, evolving along the time and space axes. The rightmost curve shows the tail of the train. The vertical distance between them, along the space axis, represents the length of the train which is of course constant. The curved parts represent the deceleration and acceleration before and after the dwell time at a stations platform for a stopping train. However, our model will also be valid for passing trains.

The bottom half of Figure 1 is in fact a vertical projection of its top half. This leaves the time dimension only for the different resources, being two routings and one platform. By doing this projection we get occupation intervals for each resource. This is the way resources are assigned in practice.

Note that there is a time overlap between each pair of subsequent resources the train travels through. This is due to the head of the train being on the next resource, while the tail of the train is still on the current resource.

$tparr_{o,m}$ is the time where the middle of the train *arr*ives at the middle of the *p*latform and marks the end of the IN movement. $tpdep_{o,m}$ is the time where the middle of the train is still at the middle of the *p*latform and marks the beginning of the OUT movement. From these two times, and by choosing IN- respectively OUT routings, we will subtract respectively add routing durations and calculate derived times outwardly.

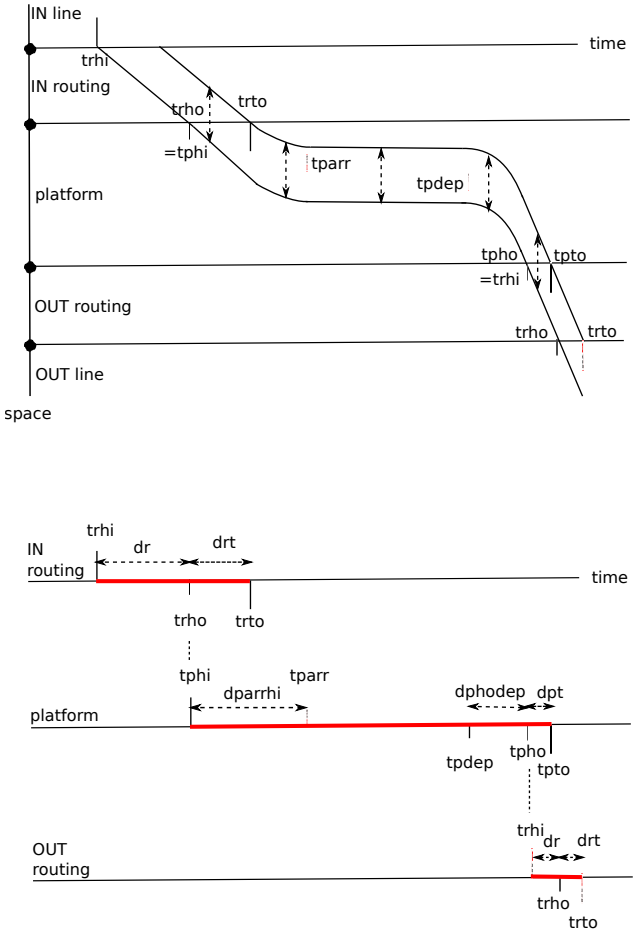

Fig. 1: Occupation Times and Durations

Mixed Integer Linear Programming Model

Based on the notation introduced above, a MILP model is constructed describing the objective function and taking into account all constraints. The central decision variables are boolean matrices $op_{o,p}$, which maps each occupation to a platform and $mr_{o,m,r}$, which maps each occupations' movement to a routing. Starting from the fixed platform times, we calculate outwardly, the upper and lower interval bounds for all possible occupations and movements. Worst case interval bounds are used to avoid formulating conflict constrains between actual interval pairs that can never overlap in time. We define boolean ordering variables *bef*ore and *aft*er, between each pair of occupations and also as their disjunctions, the *sep*arated variables. If two occupations are on the same platform, and can potentially overlap, they are explicitly time separated. This is written as $op_{o_0,p_0} \wedge op_{o_1,p_1} \implies o0sepo1_{o_0,o_1}$. Similarly, movements on common routings are time separated as, $mr_{o_0,m_0,r_0} \wedge mr_{o_1,m_1,r_1} \implies m0sepm1_{m_0,m_1}$. The constraints $\forall o \in O : \forall m \in M_o : \forall r \in R : mr_{o,m,r} \implies op_{m2o_m,r2p_r}$ nicely describe the compatibility between movements and their routings.

RESULTS

The optimizer was developed in C++ in 2 months, runs on OS X, Linux and Windows and is easily switchable to use any of the solvers mentioned in table 1. It reads infrastructure and train files, solves the problem and writes out solution files which are used by the calling software. It also writes out a graph containing all resources, being routings and platforms on the vertical axis and time on the horizontal axis, which displays all usage time intervals. Both the original and optimized allocation are given in the same figure so that the user can most easily see which trains changed platform.

Table 2 gives some results of optimizer runs on 5 stations with their respective morning peak traffic. MILP matrices that describe the problem get relatively big but the most recent release of the Cplex solver is able so solve each problem to optimality within 9 minutes.

SW HW	SW (Solver)	HW
a	Cplex v12.2	Apple 2.8 GHz
b	Gurobi v3.0.1	Apple 2.8 GHz
c	Cplex v11.2	HP 1.99 GHz
d	Xpress v7.0.2	HP 1.99 GHz

Table 1: Software (Solver) and Hardware Configurations Apple runs OS X 64 bit. HP runs Windows XP-32 bit. Both have an Intel Core 2 Duo Processor.

CONCLUSIONS

A method and tool for choosing a platform for a train set with fixed platform arrival and departure times was developed. It handles train splits and merges.

Station #P,#R,#O	SW HW	#Constraints	#Variables	Time (h,m,s)
Bergen 7,128,178	a	103362	33473	8m36s
	b	109955	37175	n.a. in 23h
	c	103365	33474	n.a. in 2.5h
	d	109955	37175	OM at 54m
Brugge 10,198,68	a	34384	10958	28s
	b	35627	11717	20s
	c	34384	10958	7s
	d	35627	11717	26s
Denderleeuw 9,206,211	a	154553	47187	1m25s
	b	159226	50074	21m44s
	c	154553	47187	10m5s
	d	159226	50074	2m16s
Leuven 14,324,256	a	105709	36242	8m14s
	b	113119	40676	40m41s
	c	105709	36242	5m5s
	d	113119	40676	1h32m8s
Mechelen 10,170,121	a	9959	4756	0.32s
	b	12201	6292	4.43s
	c	9959	4756	1.32s
	d	12201	6293	2.2s

Table 2: Optimization Execution Times
#P = number of real platforms. #R = number of real routings. #O = number of occupations, initial and supplementary together. Cplex Matrix dimensions are already (slightly) reduced ones. n.a. = not available (not enough patience limit). OM = Out of Memory.

For each train, a preferred platform can be set.

For realistic input size, being a realistic station and a few morning peak hours worth of train traffic, with the fastest MILP solver, the solver response time is from immediate to about 9 minutes, which makes it a very practical tool. The low solver times are due to the fixing of platform departure and arrival times together with a filter technique where we avoid adding constraints that describe conflict avoidance of pairs of trains that, independent of routing and platform choices, can be deduced to never arise.

On average, Cplex has the lowest solution times.

De Luca Cardillo and Mione [6] use a graph coloring algorithm and heuristics to find a good feasible solution, but optimality is not guaranteed. Of course, they achieve lower calculation times, often below a second.

Billonnet [7] uses a similar MILP model and also achieves optimal solutions at solver times very similar to ours, also up to 10 minutes. However, there are some differences. First, our problems are based on real world operational data, while Billonnet generates random instances. Second, Billonnets set of same-time train pairs is fixed before the MILP phase is started. In our model, routing allocation affects this conflict set via the different routing durations, so this effect has to be incorporated in the MILP model. This results in about 10 times the number of variables and about 10 to 100 times the number of constraints. Third, thanks to this fixed conflict set, Billonnet could use clique-based cuts, which lower solver times. Fourth, we cover the full search space, while for his fastest results, Billonnet used a restricted one (ILP2). Fifth, Billonnet, in 2003, used slower software (XA solver) and hardware.

POSSIBLE EXTENSIONS

Currently, our model only allows one train on a routing at a time. Even though this is safe, in a real station, liberation points exist, which allow a second train to pass onto the same routing as the first, as soon as the first train has passed the liberation point. Modeling this too would improve the maximum capacity of the model and get it closer to reality. However, in practice, railway companies consider liberation points as a method to be used in real time to solve cases of unexpected peaks in traffic, but they don't want the planning already to rely on it.

Our model is usable in other contexts than just capacity estimation. Indeed, for example, merely changing the goal function could result in a system also trying to lower transfer passenger walking distance.

REFERENCES

[1] Caprara, A. and Kroon, L. 1 2007 Transportation, Handbooks in Operations Research and Management Science

[2] Caprara, A. and Galli, L. (2007) *Proceedings of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems* , 49–61.

[3] Zwaneveld, P. J. e. a. (1996) Routing Trains through a Railway Station: Model Formulation and Algoritm *Transportation Science* **30(1)**, 181–194.

[4] Zwaneveld, P. J. (1997) Railway Planning and Allocation of Passenger Lines. Ph. D. Thesis Rotterdam School of Management.

[5] Kroon, L. and Romeijn, H. (1997) Scheduling and Platforming Trains at Busy Complex Stations *EJOR* **98(1)**, 485–498.

[6] De Luca Cardillo, D. and Mione, N. (1999) k L-List tau Colouring of Graphs *EJOR* **106(1)**, 160–164.

[7] Billonnet, A. (2003) Using Integer Programming to Solve the Train Platforming Problem *Transportation Science* **37(1)**, 213–222.