An object-oriented approach to the train platforming problem

M. Galaverna\textsuperscript{a}, A. Imovilli\textsuperscript{b} & G. Sciutto\textsuperscript{a}

\textsuperscript{a} Dipartimento di Ingegneria Elettrica, Università degli Studi di Genova, Italy
\textsuperscript{b} Istituto di Transporti, Università degli Studi di Genova, Italy

Abstract

This paper proposes a computer-based solution to train platforming problem. Firstly, train platforming problem is introduced as for maximisation of station traffic capacity. Then, the paper describes the software tool especially designed by the authors in order to perform optimised train platforming automatically. Finally, an application of above mentioned software tool to case studies is presented.

1 Introduction

The carrying capacity of a station is defined as the maximum number of trains that can be platformed in a unit time, without irregularity in train running, i.e. on the hypothesis that no train has to stop because of a red signal, waiting for an available station track. The unit time usually assumed is a hour, or a day; when main stations are concerned, carrying capacity per hour is considered by preference, as rush hours and off-peak hours occur alternately [1].

Station carrying capacity plays a leading role in carrying capacity of a railway system as a whole; unfortunately, it is difficult to evaluate station carrying capacity by means of equations, since it depends upon numerous deterministic and non-deterministic parameters.

The carrying capacity of a station equipped with N platforms is defined as

\[ C = \frac{60 \, k \, N \, T}{t_{tot}} \]  \hspace{1cm} (1)

where

- \( T \) = unit time, expressed in hours;
- \( k \) = empirical coefficient, from 0 to 1, which takes into account interferences among train routes; less effective interlocking systems result in lower values of \( k \);
Evaluation of $t_{tot}$ can be performed through the following equation:

$$t_{tot} = t_1 + t_2 + t_3 + t_s + t_{par}$$

where

- $t_1$ = time necessary to set up an entry route for a train;
- $t_2$ = time interval which must occur after the route has been prepared, before the train has arrived at entry signal (this time interval is necessary in order to prevent avoidable slowing down);
- $t_3$ = time required to cover the section from entry signal to the stopping point in station;
- $t_s$ = dwelling time (zero, if a train has not to stop);
- $t_{par}$ = time required to cover the getting out route (route setting up for leaving trains is contained in $t_s$).

More complex formulas, by taking into account different categories and priorities of trains, allow one to evaluate residual station carrying capacity, that is the maximum number of trains that can be platformed in addition to those already scheduled, before that station carrying capacity is saturated.

Recently, an alternative approach to evaluating the carrying capacity of a station has become available, that is computer simulation of train movement. Computer simulation provides a more accurate evaluation than analytical approach can do, as uncertainty about coefficient $k$ is overcome: infect, it is possible to build an accurate computer model of stations, by taking into account peculiarities of stations, such as the length of platforms and the number of switches. The methodology based on computer simulation consists in reproducing a succession of situations, characterised by so an increasing number of circulating trains that saturation of station carrying capacity is reached. By doing so, carrying capacity can be found as the maximum number of trains that turned out to be likely to be platformed, before the forming of a train queue.

Theoretical carrying capacity can be exploited to an extent that depends on criteria by which train platforming is performed. Train platforming problem must be faced whenever timetable is changed, that is twice a year on Italian Railways (FS). Train platforming consists in the assignment of trains to station tracks [2].

2 The train platforming problem

On Italian Railways, train platforming is currently performed by special personnel, without computer aid. For main stations, train platforming problem is very difficult, as many constraints must be taken into account. For this reason, train-to-track assignment is usually performed by introducing alterations into pre-existent assignment plan; so, this action usually turns out in inserting new trains in a platforming plan. Since current approach to train
platforming problem is based on repeated alterations of pre-existent assignment plan, the result is often non-optimal, that is, the carrying capacity of a station is not always employed at best.

A train timetable, if correctly conceived, is never in contrast with station carrying capacity. As a consequence, train platforming is devised in such a way that no train, if it arrives on time, has to stop at entry signal because no entry route is available. Nevertheless, train platforming plan must be robust, that is, it should be possible to manage incidental irregularity in train running by introducing few alterations in the platforming plan. Examples of events which may cause an irregularity are train delays, failure events, track maintenance intervention which make a portion of station unusable, running of conditional trains.

According to aforesaid topics, it is possible to envisage train platforming problem in two ways, that is, in a static way and a dynamic way. The former corresponds to the method currently followed by the staff that makes out the platforming plan; according to this method, it is usually assumed that no irregularity of rail traffic may occur, and the robustness of platforming plan is the only precaution taken against irregularity in rail traffic. The latter corresponds to the process carried out by a station inspector whenever it is impossible to follow the scheduled platforming plan, because of unexpected events; in such a situation, the station inspector has to decide about train platforming on real time, and it is hard to achieve an optimal decision.

The ideal solution is a software tool able to suggest the optimised train platforming on real time, and to decide on the optimal sequence of train entries. The present work proposes a software tool able to perform optimised train platforming off-line; the realisation of a software module able to suggest optimised train platforming on-line represents a future extension of this research activity.

3 Operational constraints

The main difficulty in resolving train platforming problem is the treatment of constraints. Operational constraints can be classified in two categories:

- constraints relevant to trains;
- constraints imposed by interlocking.

The former include assignment constraints and condition constraints. Assignment constraints force a train to be assigned to one or some pre-determined tracks; examples of such constraints are those imposed by the discharge of goods or mail, and by previous heating of coaches. Condition constraints are imposed by trains that connect; these trains should be platformed in adjacent tracks, in order to limit the distance that passengers have to cover, and to prevent passengers from crossing the tracks.

Constraints imposed by interlocking prevent simultaneous utilisation of switches or portions of track by more than one train. These constraints can be
modelling by imposing mutual exclusion in utilisation of particular "resources".

![Flow-chart of platforming algorithm](image_url)

**Figure 1: Flow-chart of platforming algorithm.**

### 4 Description of the software tool

This section describes the software tool developed by the authors to solve train platforming problem for a given station. The algorithm is shown in Figure 1. Input data include train timetable, the track plan of the station, the constraints imposed by interlocking, information about trains, such as their category, composition, and length. Firstly, the program looks for the largest time interval between two consecutive trains, in order to set the starting point for train platforming. Then, for each train included in timetable, the program
tries to assign the platform in such a way that the highest number of shunting movement be possible, in order to maximise the carrying capacity of the station.

During the execution, the program updates and stores the state of every switch and track circuit of the station, by simulating train movement in a very accurate manner. For each train, the program finds the platforms that are available in the requested time interval; then, the program enumerates the routes which link up the selected platforms with the origin or destination line. The selected routes are compared and listed according to the criterion chosen by the user. Possible ordering criteria are the shortest way, the highest speed, the lowest number of interlocked switches. According to the last criterion, the program calculates, for each route, the value of a cost function defined as:

\[ f_i = \sum_{j=1}^{N} c_j \]  

where

- \( N = \) number of switches included in i-th route;
- \( c_j = \) number of routes which include j-th switch

and searches for optimal route through the minimisation of the above-mentioned cost function. However, all the optimisation criteria aim at minimising time intervals required for train movements in station, in order to release platforms as soon as possible and make them available for other trains.

Then, the program checks all the switches included in the selected route, through the calculation of the instants of occupation and clearing of every track circuit; by doing so, the program makes sure that the constraints put by the interlocking system are respected. In this procedure, train movements are simulated either through the laws of accelerated and decelerated motion, for trains that have to stop in station, or by assuming constant speed running for trains that have not to stop. After an available route has been found, its elements (i.e., switches and track circuits) are marked as occupied; then, the program processes the next train, and repeats the above-mentioned procedures for this train. In case the program does not succeed in finding an available platform for a train, because of a conflict among routes, a backtracking procedure is performed, that is, the previous trains are platformed in tracks different from those previously assigned, in order to remove the conflict (see Fig. 2).

Time necessary to shunting work is taken into account by the program; to this end, for trains which originate in the considered station, the platform is assumed engaged 15 or 20 minutes longer, before train arrival or after train departure. Trains whose rolling-stock is afterwards used for other trains are temporarily replaced to secondary tracks, in order to make platforms available for other arriving trains; even in this case, time necessary to shunting work is calculated on the basis of the above-mentioned standard time intervals.
5 Software tool features

The main purpose of the software tool development was to realise a software tool that can be expanded later, so we decided to use an object oriented programming language, the C++. The main feature of C++ is to combine the C programming language feature with the new object oriented characteristics (classes, methods, virtual functions, polymorphism, inheritance). In this manner we have obtained good results of software engineering.

The software tool runs on Personal Computer (i.e. with 386 or 486 microprocessor and 4 Mbytes of RAM) with the DOS Operating System. We have used the Borlandc C++ 3.0 Compiler so it shall be possible to expand the software tool (using the Object Windows Library OWL) for run with an advanced Graphic User Interface (GUI) such as Microsoft Windows GUI.

The three main classes that we have realised in the software design phase are:
- Track Circuit Class;
- Switch Class;
- Train Class.
They can be contained in high level classes such as:
- Entry route Class;
- Departure route Class.

At last the Station Class is the container of all previous classes and it realise (with its methods) the logic connection of all parts.

In the software design we have decided to subdivide the software tools in three different computer programs:
- the input program realises a pre-processing on the input files data and produces some intermediate files;
- the data processing resolves the platforming problem;
- the output program realises a post-processing of the platforming problem results and produces a graphic output.

6 Case studies and extensions

The software tool described in the previous paragraph has been tested for Genova Brignole and Sestri Levante stations, which are placed on the main line Turin-Rome. Genova Brignole station has 19 tracks, 90 entry routes, 80 departure routes, and 145 track circuits; average traffic intensity amounts to 368 trains per day. Sestri Levante station has 8 tracks, one of which used by goods train only, and 3 holding sidings; there are 48 track circuits, 34 entry routes, and 36 departure routes; average traffic intensity amounts to 222 trains per day.

The platform plan resulting from the application of the software tool to these stations has been considered as feasible by the staff that at present performs train platforming without any computer aid.

The research work on the identification of parameters suitable for providing a measure of the quality of a platforming plan is still being carried out. Other future extensions of this research work concern the modelling of terminal stations, the increase of efficiency of the backtracking algorithm, and the realisation of a version suitable for on-line platforming.

7 Conclusions

This paper has presented a computer program able to solve train platforming problem for through stations. First, the weight of train platforming in determining the carrying capacity of a station has been pointed out. Then, the paper describes the computer-based approach to train platforming problem followed by the authors, some applications to real case studies, and the future extensions of this research work.

8 References

