Scheduling and platforming trains at busy complex stations

Malachy Carey a,*, Sinead Carville b

a Faculty of Business and Management, University of Ulster, Newtownabbey, Northern Ireland BT37 0QB, UK
b Nortel Networks, Newtownabbey, Northern Ireland BT36 6XA, UK

Received 12 October 1998; received in revised form 12 March 2002; accepted 26 March 2002

Abstract

We consider the problem of train planning or scheduling for large, busy, complex train stations, which are common in Europe and elsewhere, though not in North America. We develop the constraints and objectives for this problem, but these are too computationally complex to solve by standard combinatorial search or integer programming methods. Also, the problem is somewhat political in nature, that is, it does not have a clear objective function because it involves multiple train operators with conflicting interests. We therefore develop scheduling heuristics analogous to those successfully adopted by train planners using “manual” methods. We tested the model and algorithms by applying to a typical large station that exhibits most of the complexities found in practice. The results compare well with those found by traditional methods, and take account of cost and preference trade-offs not handled by those methods. With successive refinements, the algorithm eventually took only a few seconds to run, the time depending on the version of the algorithm and the scheduling problem. The scheduling models and algorithms developed and tested here can be used on their own, or as key components for a more general system for train scheduling for a rail line or network.

Train scheduling for a busy station includes ensuring that there are no conflicts between several hundred trains per day going in and out of the station on intersecting paths from multiple in-lines and out-lines to multiple platforms, while ensuring that each train is allowed at least its minimum required headways, dwell time, turnaround time and trip time. This has to be done while minimizing (costs of) deviations from desired times, platforms or lines, allowing for conflicts due to through-platforms, dead-end platforms, multiple sub-platforms, and possible constraints due to infrastructure, safety or business policy.

© 2003 Elsevier Science Ltd. All rights reserved.

Keywords: Transport scheduling; Train timetabling; Train platforming; Heuristic search
1. Introduction

Most research on train planning and timetabling is concerned with scheduling trains on lines, usually single lines, and is not concerned with busy multi-platform stations with multiple in-lines and out-lines, or assumes these have unlimited capacity (Assad, 1980; Crainic, 1988; Petersen et al., 1986; Jovanovic, 1989; Jovanovic and Harker, 1991; Kraay et al., 1991; Carey, 1994a,b; Carey and Lockwood, 1995; Odijk, 1996 and a recent survey Cordeau et al., 1998). This is also true of unpublished work done by or for train or rail operators. However, in Europe busy complex rail stations are key components of the busy passenger rail networks, and are the location of most train conflicts. Since these are the most complex and parts of the network to schedule, we focus on them in the present paper.

Stations with several hundred, or over a thousand, trains per day are common throughout Europe and elsewhere. Such stations typically have multiple intersecting in-lines and out-lines, connected to multiple parallel or sequential stopping points (platforms), of different types and lengths, some being dead-end and some being one or two-way through-platforms. The trains differ in their types, speeds, desired dwell times and headways, origins and destinations, and preferred or required lines and platforms. For safety and other reasons, headways (time gaps) are required before and after each train arrival and departure at each platform and on each line. There may be other temporary or permanent constraints or preferences due to station layout, safety or signalling requirements, operating policy or marketing policy. Train planning for a large busy station includes drawing up a schedule to ensure that there are no conflicts between any trains, while ensuring that all the above requirements and constraints are satisfied and minimising any deviations, or costs of deviations, from desired or preferred times, platforms or lines for each train.

The train planning process, especially for busy passenger rail systems, is normally broken down into several stages (e.g., see Watson, 2001). The first stage consists of train operators drawing up outline or draft train plans (timetables) for running train services. These drafts are based on their estimates of travel demands between origin and destination stations, the distribution of demands throughout the day, estimates of revenues and costs, estimates of available rolling stock and crews, existing contracts and commitments, business or marketing plans, and whatever knowledge they may have of plans of competitors. The draft train plans are likely to contain various types of conflicts of train times, headways, line and platform usage, etc, as indicated in the previous paragraph above. The second stage consists of taking the draft train plans (perhaps from a number of different operators using the same lines and stations) and adjusting or revising these to eliminate all of the conflicts, while pursing cost or benefit objectives and as far as possible adhering to the preferred lines, platforms and times for each train. It is this second stage with while we are mainly concerned in the present paper. Also, we focus on a single large busy train station, with multiple platforms and multiple conflicting lines in and out of the station. In ongoing and later work we take the single-station scheduling models and algorithms from the present paper and link them together into models and algorithms for a busy rail corridor (Carey and Crawford, 2000, 2001).

The organisation of the above train planning process has been changing in recent years. In many countries there has been a trend towards separating ownership or control of the infrastructure (lines, signals, stations, etc.) from the operation and marketing of train services. In that case the initial proposed schedules are produced by separate, often competing, train operating companies (TOC’s), and are then presented as bids to a company which owns and controls the
tracks. For example, in Britain (see Harris and Godward, 1997) there are 25 major TOC’s but the lines, and access to the lines, are owned and controlled by Railtrack. The latter takes all the proposed train schedules, finds and resolves conflicts between them, and makes schedule offers based on this to the 25 companies.

Scheduling trains at busy stations is still widely done manually by experienced schedulers who over the years have developed methods for finding good or acceptable solutions for this combinatorial problem. In the algorithms that we develop here, we emulate some of these methods, e.g., dealing with one train at a time, dealing with trains in chronological order, sub-dividing into speed classes, restricting search to time windows, etc. These traditional manual methods are now often supported by computer graphical interfaces (e.g., see Rivier et al., 1984, 1987, 1990; Brooke, 1996; Hammerton, 1996) for input, output, manipulation and display of the information, particularly for lines between stations. While these are very useful and necessary, they leave the actual finding and resolving of conflicts, and other scheduling choices, to the human train planner or scheduler. Hence they leave out what we are concerned with in the present paper. While human planners can find and resolve conflicts, it is a slow ad hoc process and does not allow planners time to explore alternative options, or to make other than very simple calculations, or to perform ‘what if’ analysis in time to be of use to managers. An automated method of finding and resolving conflicts could overcome these problems: other advantages of such a system are discussed later in the introduction.

Zwaneveld et al. (1996) consider the problem of routing trains through railway stations. Given the layout of the station and a proposed timetable they determine whether there is a feasible routing for the trains to satisfy safety rules (headways) and train service or connection constraints. They formulate the problem as a node-packing problem and describe a solution procedure based on a branch-and-cut approach. Kroon et al. (1997) consider the complexity of variants of the above routing problem, formulating it as a fixed interval-scheduling problem. They show that the problem is NP-complete if each train has three or more routing possibilities, but if the train has only two routing possibilities then the problem can be solved in a time that is polynomial in the number of trains. The problem considered by Zwaneveld et al. (1996) and Kroon et al. (1997) differs from that considered in the present paper. They consider the train routing feasibility problem while we also consider train planners objectives, in particular, the various trains or delays or platform choices may have different costs or penalty function and an important part of the train planners task is to reduce or minimise these costs. A further reason for our taking a different approach is that we wish to include heuristics analogous to those used in practice by train planners and managers. One reason for this is that it makes the choices more understandable to train planners and makes it easier for them to adapt or revise the proposed solution. Also, for example in Britain, it makes it easier to gain acceptance of the rules or heuristics by the competing train operators who are affected by the scheduling choices, and easier to satisfy the train regulator that the rules are consistent and fair.

Perhaps the most obvious mathematical approach is to treat the problem as a mixed integer mathematical programming problem (MIP). That is, set out an objective function (costs to be minimised or benefits to be maximised) subject to various different types of constraints. An MIP formulation of the problem was set out in Carey and Carville (1994) and Carey (1996). To appreciate the computational problems with an MIP formulation, we out-line here the number of 0–1 integer variables for a basic formulation. Since continuous variables and (linear) constraints are
not a problem in a MIP we will not count these, though they may be used to enforce operating rules or commercial policies for train times or platforms.

In their formulations Carey and Carville let \( P \) be the number of platforms and \( T \) be the number of trains. They use a \( 0–1 \) integer variable to indicate whether train \( t \) is, or is not, assigned to platform \( p \), hence \( T \times P \) such variables. A \( 0–1 \) integer variable is used to indicate whether train \( t \) arrives before train \( t' \) arrives, hence \( T^2/2 \) such variables (if there is no restriction on the order in which the trains are allowed to arrive). Similarly, \( T^2/2 \) \( 0–1 \) integer variables are used to indicate whether \( t \) departs before \( t' \) departs, a further \( T^2/2 \) to indicate whether \( t \) arrives before \( t' \) departs, and a further \( T^2/2 \) to indicate whether \( t \) departs before \( t' \) arrives. The number of \( 0–1 \) integer variables needed above is thus \( 4T^2/2 \). However, in practice it may not be acceptable to move a train more than say six trains away from its initial draft position (slot) in the train order. For each of the above four types of \( 0–1 \) integer variables, this reduces the number of \( 0–1 \) variables from \( T^2/2 \) to \( 6T^2/2 = 3T \). The total number of integer variables above is thus \( TP + 2T^2 \), or is \( TP + 4(3T) \) if trains cannot be moved more than six slots from their draft slot. A typical medium size rail station is say Leeds, England, which has \( P = 33 \) sub-platforms and \( T = 500 \) trains per day. In that case the number of \( 0–1 \) integer variables is \( TP + 2T^2 = 500(33) + 2(500^2) = 516,500 \), which gives \( 2^{516,500} \) possible solutions to be considered. If the train order is restricted as above (to move no more than six steps from their draft slots) the number of \( 0–1 \) integer variables is reduced to \( TP + 4(3T) = 500(33) + 12(500) = 22,500 \), which gives \( 2^{22,500} \) possible solutions to be considered.

This size of \( 0–1 \) integer problem may not be solvable by integer programming methods in any acceptable time frame. This is especially true since we will normally wish to solve the problem many times, for a variety of scenarios and parameters. Also, the problem out-lined above from Carey and Carville (1994) and Carey (1996) is simpler that the general problem considered in the present paper since. For example, they do not consider main platforms divided into a sequence of sub-platforms and additional \( 0–1 \) integer variables are needed to formulate headway constraints for these.

Partly because of the above, rather than using integer programming algorithms or packages, we used heuristics analogous to those used by train planners in practice. These take advantage of the structure of the problem to reduce the search space, and use knowledge of the problem and the data to look for the most promising solutions first. For example, we consider one train at a time and find and resolve all conflicts for that train before considering the next train. We can of course return later to that train to reconsider it. Also, when considering a train, we consider assigning it to each platform in turn, to find the best platform, and in doing this we consider platforms in descending order of preference. Similarly, we use heuristics to more quickly find which trains conflict with train \( t \) at any platform \( p \). Comparing these heuristics with MIP and branch-and-bound algorithms, their main benefit is that they use knowledge of the problem to focus on the most profitable branches to search.

There are also other reasons for using heuristics analogous to those used in existing manual methods. First, the train planners’ manual heuristics embody knowledge as to what produces good or acceptable solutions. Second, to have confidence in computer based algorithms, train planners and operators have told us they would first wish to see that these could produce results similar to those obtained from the existing manual methods. After that, they would be open to further improvements, extensions, etc. Third, the choices made by the algorithms will have to be ‘explainable’ in terms that are acceptable to train planners, operators and regulators. The existing heuristic methods are explainable.
Since the basic version of our heuristic algorithms are designed to produce results similar to those from existing manual methods, it may be wondered why not simply retain the latter. However, there are several advantages of formal computer based algorithms over manual methods, even if there were no improvement in the value of the objective function compared to manual methods. First, the algorithms formalise what was a largely unwritten, and somewhat ad hoc, method that varied from one train planner to the next, and thus helps provide consistency. The regulatory framework now expects consistency and fairness in the treatment of trains belonging to the various train operators. Second, the existing manual methods take several weeks to generate a schedule of train times and platform assignments for a set of stations. As a result, there is no time to consider alternative patterns of train services, alternative operating policies for headways, dwell times, etc. In a computer based algorithm, all of these can potentially be considered in minutes. Third, a computer-based algorithm can be extended to include additional information, search calculations, comparisons, etc. that train planners would like to include but are not practicable in manual methods. Finally, an important advantage is that the single-station algorithms in this paper can be used as a component in more general models, as outlined in the final section of the paper.

Over the past several years a number of general heuristic methods for solving IP or MIP or combinatorial problems have been developed, and in a range of contexts they have often been found to be very effective and provide good or acceptable solutions. These general heuristic methods are genetic algorithm, tabu search, and simulated annealing (Davis, 1987; Goldberg, 1989; Reeves, 1993; Glover and Laguna, 1997; Osman and Kelly, 1997). We have not adopted these approaches here, since we found that our own problem specific heuristics were sufficient for the examples to which we have applied them. However, the specific heuristics used in this paper could also be useful in developing algorithms based on genetic algorithms, tabu search or simulated annealing. This is because the performances of these latter heuristics depends very much on using search strategies tailored to the structure of the problem in hand, and is often greatly improved by incorporating heuristic that have already been found effective in previous methods.

What we present in this paper is not ready for use as a commercial system. For that, the present models and algorithms would need to be linked or integrated with a graphical user interface, an infrastructure data base, a trains data base, an information system which stores timetable and resource information, and which feeds results to and from similar databases for other parts of the network. However, all of these are developed, or being developed, for rail companies by commercial IT companies. Hence we concentrate here on the problem of finding and resolving conflicts, while satisfying other constraints and taking account of objectives.

There is a feature of our problem formulation that is perhaps worth noting here. The costs used in comparing scheduling options consist of costs or penalties for deviations from preferred train times and costs or penalties for choosing less preferred platforms on sub-platforms or lines. There are difficult issues involved in adding up these different types of costs or penalties. We get round this by introducing “lexicographic” cost functions or decision rules, which also have other advantages that we note, including consistency with current practice and traditional manual methods.

In Section 2 we introduce and discuss the variables, parameters, costs, objectives and constraints for the station scheduling problem and formalise these by introducing appropriate notation. In Section 3 we develop an algorithmic approach to generating a station schedule, that is, a
scheduled arrival time, departure time and platform allocation for each train. We discuss and formalise methods for finding and resolving conflicts and enforcing headway constraints, and bring these together in a formal algorithm. In Section 4 we apply the model and algorithms to a busy complex station and discuss the computational results. The data used in our computational examples is given in Appendix A. In Sections 4.4 and 5 we discuss further development and uses of this work.

2. The platforming problem: conflicts, headways and costs

2.1. Train platforming context and notation

We introduce some basic notation here and further notation will be introduced when it is first needed throughout Sections 2 and 3. For reference, all notation is summarised again in Appendix A.

2.1.1. Subscripts

Let \( t \) and \( t_0 \) denote trains. In our scheduling algorithms we let \( t \) be the current train and \( t_0 \) be the next train to be checked for conflicts with train \( t \). Let \( p \) denotes a platform at a station. There may be any number of platforms, from one to dozens. Since terminology may differ in different countries, we note that a train “platform” usually refers to a raised strip alongside a train track that is used for passengers alighting or boarding a train or for loading or unloading freight, etc. However, for train scheduling purposes, and in this paper, we let platform denote a segment of track on which a train stops at a station and in which only one train at a time is permitted.

2.1.2. Data

When constructing a detailed plan of train times and platforms, the starting point is usually an out-line plan of desired arrival, departure and dwell times. For example, the train planners may wish to keep much of the timetable the same as the previous year, while changing the times of some services and adding or deleting some services. Even when constructing a completely new timetable, planners draft an outline of train services (arrival and departure times), based on revenue considerations, before getting down to the detailed finding and resolving of headway and platform conflicts. Let,

\[ A_{t}^{\text{des}}, D_{t}^{\text{des}} \text{ and } W_{t}^{\text{des}}: \text{the desired arrival, departure and dwell times respectively for train } t. \]

The desired times may sometimes depend on the platform used, for example, if a platform is preferred by many trains, so that there is competition for time at that platform, then a smaller minimum dwell time may be allowed at that platform. In that case add a \( p \) subscript, thus \( A_{tp}^{\text{des}} \), etc.

\( I_{t}, O_{t}: \text{line on which train } t \text{ arrives (in-line) and departs (out-line) respectively.} \)

\text{Upper bounds on arrival, departure and dwell times:} \text{ There are normally upper bounds on the acceptable arrival and departure time for each train at a station. When planning train times for a single train station these upper bounds are usually fairly tight. If a train cannot be slotted in near to its earliest desired times at the station, then the train planners may have to go back to “stage one” of the planning process, as described above. It is not always be possible to find a feasible time for a train within its acceptable upper bound, since the requirement for headways and dwell}
times means there is a limit to the number of trains that can be slotted in at a station within any given time period. This limit has already been reached at peak times or busy parts of the rail network in many countries. Let
\[ A^U_t, D^U_t, W^U_t: \] the upper bound on acceptable arrival departure and dwell times respectively for train \( t \).

If conflict free train times \( (A_t, D_t, W_t) \) cannot be found within these bounds then in the present model and algorithm we provided two options:

(a) drop this train altogether from the set of trains to be scheduled, or;
(b) ignore the upper bound and find the earliest best (lowest cost) times and platform for the train beyond the “upper” bound.

An advantage of option (b) is that at least it provides the train operator with some information as to how soon the train could be slotted in, which may assist them in designing or proposing alternative services. A disadvantage of option (b) is that, by slotting in some trains at times that we know will not be acceptable, we may be tying up time slots that could be used by other trains, hence pushing these trains into less desirable slots or even pushing them beyond their upper bound times.

Despite the above objection, we used option (b) in most of the computer runs and tests that we performed with our scheduling algorithms. Our main reason for doing this was that it provided a more difficult scheduling problem for the algorithm to handle. Option (b) will of course tend to provide worse results than option (a) in terms of performance measures discussed later, namely deviations from desired times and platforms.

2.1.3. Variables

At the detailed timetabling and platforming stage (which is the concern of this paper), many of the initial draft or desired train times may have to be adjusted, for example, because some platforms are already occupied, or train times and paths conflict, or some trains have a higher priority or cost than others, etc.

\( A_t, D_t \) and \( W_t: \) the arrival, departure and dwell times respectively for train \( t \), obtained from the scheduling algorithm set out in this paper.

\( p_t: \) platform to which train \( t \) is assigned.

\( T_p: \) set of trains already assigned to platform \( p \).

\( T_p \) and \( p_t \) includes any assignments fixed in advance and any that have been assigned (fixed) by the platforming algorithm up to the current stage of the algorithm.

2.1.4. Temporary variables

In the course of searching for the best feasible train times \( A_t, D_t \) and \( W_t \), we temporarily assign each train \( t \) to various platforms \( p \). To do this, let

\( A^*_t, D^*_t \) and \( W^*_t: \) trial arrival, departure and dwell times of train \( t \) when (temporarily) assigned to platform \( p \).

These trial times (for non-fixed trains) may be changed several times in the course of the algorithm. We could also refer to these times as draft, interim, or incumbent times.
Notation for computation: In the computational algorithms in Section 3 below, the temporary variables \( A^t_p, D^t_p, W^t_p \) are used when finding the best platform for the current train \( t \). Having found this, we move on to the next train, say \( t+1 \), and do not need these temporary variables again for train \( t \). Since we need these variables only for the current train, we need only store \( A^t_p, D^t_p \) and \( W^t_p \). If there are several hundred trains, this reduces storage space for these arrays by several hundred times. However, for expositional purposes in this paper we leave the \( t \) subscript in place. Similarly, in Section 3, for computations we could do without the \( t' \) subscript on \( B^t_{i_0} \) and \( B^t_{i_2} \).

2.2. Platform feasibility

Trains have various characteristics and requirements which must match those of the station platform. For example, a train can use a platform only if the platform is connected to the line on which the train arrives (the in-line) and the line on which it exits (the out-line). The platform must be long enough to accommodate the train, and if the train is electric the platform must be electrified. If the train requires watering or buffet car restocking these must be available at the platform. To represent these requirements, introduce a feasibility indicator, 

\[ F^t_p: \text{TRUE if platform } p \text{ is feasible for train } t, \text{otherwise FALSE.} \]

2.3. Platform desirability costs

Even if several platforms are feasible for train \( t \) some platforms are more desirable than others. For example, some train businesses may customise some platforms for their own use, or may prefer platforms near to ticket sales points, the station entrance, car parks, other public transport, platforms used by connecting services, station shops or catering. To take account of this, let 

\[ C^u_t: \text{the undesirability of platform } p \text{ to train } t. \]

For example, rank the platforms in order of desirability for train \( t \) and set \( C^u_t \) equal to this rank order. Thus the platforms for train \( t \) in descending preference order may be 3, 9, 5, 12, 2, \ldots, hence \( C^u_{i_3} = 1, C^u_{i_9} = 2, \text{etc.} \) If there is a tie for say platforms 5 and 12 then \( C^u_{i_5} = C^u_{i_{12}} = 3. \)

2.4. Platform obstruction costs

At a station there may be a number of multi-train platforms, that is, platforms each of which can accommodate more than one train at a time. This is accomplished by dividing a platform into two or more sub-platforms and treating each as a separate platform which can accommodate a single train. However, if a train is present at one of these sub-platforms it may block or obstruct trains wishing to use other sub-platforms of the multi-platform. To deter this we introduce, 

\[ C^o_{tp}: \text{a sub-platform obstruction penalty or cost due sending train } t \text{ to sub-platform } p. \]

This penalty or cost is based on some prior estimate of the likelihood that if train \( t \) uses this sub-platform it would block some other train. It may block another train later in the scheduling process, or it may block a train on-the-day due to on-the-day random deviations from the scheduled arrival, dwell or departure times. Since the likelihoods of these events are difficult to estimate we can use a simple ad hoc penalty. For example, rank the sub-platforms in increasing order from ‘furthest in’ to ‘furthest out’. Those further out are likely to block more arriving and departing trains hence assign them a higher obstruction cost \( C^o_{tp} \). For example, for a terminal
platform consisting of say three sub-platforms, let the obstruction costs be 0, 1, 2, starting from the furthest in. And for a ‘through’ platform consisting of say three sub-platforms, let the obstruction costs be 1, 0, 1, starting from either end. Alternatively, if more trains arrive from the right than the left, let the sub-platform obstruction costs be say 0, 1, 2, starting from the left.

2.5. Time adjustment costs

In constructing a timetable we first attempt to assign each train to its most desired arrival and departure times $A_{t}^{ed}$ and $D_{t}^{ed}$. However, this may not be possible. For example, train times and paths may conflict, or all feasible platforms may already be occupied, or other trains may have a higher priority or cost, etc. In view of this, train arrival and departure times are adjusted to new times $A_{t}$ and $D_{t}$. The deviations from the desired times are $(A_{t}^{ed} - A_{t})$ and $(D_{t}^{ed} - D_{t})$. These deviations usually impose some cost on train operators and users, since the desired times were based on estimates of revenue, convenience, and efficient use of personnel, rolling stock, etc. The cost of these time adjustments or deviations can be written as

$$c_{a}^{t} = c_{a}^{t}(A_{t}^{ed} - A_{t}) \quad \text{and} \quad c_{d}^{t} = c_{d}^{t}(D_{t}^{ed} - D_{t}) \quad \text{and} \quad c_{w}^{t} = c_{w}^{t}(W_{t} - W_{t}^{ed})$$

Since the dwell time $W_{t} = (D_{t} - A_{t})$, we have to be careful to avoid double counting costs, e.g., if the arrival time is fixed an increase in departure time also increases dwell time. However, dwell time can impose a separate cost, e.g., for some trains an increase in both the arrival and departure time may impose a cost only if the dwell time increases.

In searching for the ‘best’ (least cost) conflict-free arrival, departure and dwell times, we temporarily assign each train $t$ to various platforms on a trial basis. The trial arrival and departure times for train $t$ at platform $p$ are $A_{tp}^{*}$ and $D_{tp}^{*}$. Hence the time adjustment cost associated with these trial times can be written as

$$c_{a}^{tp} = c_{a}^{tp}(A_{t} - A_{tp}^{*}) \quad \text{and} \quad c_{d}^{tp} = c_{d}^{tp}(D_{t} - D_{tp}^{*}) \quad \text{and} \quad c_{w}^{tp} = c_{w}^{tp}(W_{t} - W_{tp}^{*})$$

These costs functions can often treated as linear, if only because in practice they are likely to be rough estimates. They would be non-linear and increasing for large deviations from desired times, but these are ruled out by upper bounds on the permitted train times. Linear cost functions can be written as

$$c_{a}^{tp} = (A_{t} - A_{tp}^{*})C_{a}^{tp} \quad \text{and} \quad c_{d}^{tp} = (D_{t} - D_{tp}^{*})C_{d}^{tp} \quad \text{and} \quad c_{w}^{tp} = (W_{t} - W_{tp}^{*})C_{w}^{tp}$$

where $C_{a}^{tp}$, $C_{d}^{tp}$ and $C_{w}^{tp}$ are constant costs per minute of deviation. The $p$ subscript on these will usually not be needed.

It is important to note that the scheduling methods in this paper can easily handle time adjustment cost functions which are as complex and non-linear as we wish as these make a negligible difference to the calculations. They would of course increase the possibility of finding local non-optimal solutions, but that is inherent in the problem rather than in the method.

Summing the above costs gives the total time adjustment or knock-on cost,

$$c_{k}^{tp} = c_{a}^{tp} + c_{d}^{tp} + c_{w}^{tp}.$$
2.6. A lexicographic cost function and decision rule

There may be number of feasible platforms for a train and we wish to choose the best (least cost) of these. Each platform has a set of three costs associated with it, a platform undesirability cost $C_{up}$ (Section 2.3), a platform obstruction cost $C_{op}$ (Section 2.4) and a time adjustment or knock-on cost $c_{kp}$ (Section 2.5). To obtain an overall cost of for train $t$ at platform $p$ we could add up these three costs. However, since data for these costs is not easy to obtain, penalties or rankings are typically used instead, and these are not easily comparable between the three types of cost. In that case, it is not very meaningful to add up the three costs. However, to find the best platform, we must combine the costs in some way, and we set out some ways here.

(i) Convert the costs to common units of time or money and take the sum or weighted sum for each platform. It seems that this is seldom done in practice and there is no readily available data on suitable conversion factors.

(ii) Use a lexicographic cost function/decision rule. We have borrowed the term lexicographic from economic theory, where a lexicographic utility (or cost) function for a set of goods means that the goods can be ranked in decreasing order of desirability and that one more unit of a good is always preferred to any amount of the next lower good in the rank order. A similar idea, sometimes referred to as “non-Archimedian” costs or penalties, is sometimes used to rank goals in the variant of linear programming called “goal programming”. For assigning train platforms a lexicographic cost function implies a lexicographic decision rule as follows. Divide the platform costs for a train into types 1, 2, 3, etc., in descending order of importance. From the available feasible platforms for train $t$, choose the one with the lowest cost of type 1. If there is a tie, then from the tied platforms choose the one with the lowest type 2 cost. If there is again a tie, choose the one with the lowest type 3 cost, and so on.

To implement this, we let time adjustment or knock-on costs be type 1, platform desirability cost be type 2, and platform occupation costs be type 3. We observed that this approximated the cost ranking used by some experienced train planners in Britain, but other rankings can easily be used.

It may be thought that using costs of types 2 and 3 only as tie breakers is giving only a trivial role to these costs. But that is not so, since cost ties are prevalent. At large stations there are often several feasible platforms for a train and the knock-on costs (type 1 costs) for most of these may be zero. In that case it is the type 2 costs which determines the choice of platform. Also, at large stations there are often a number of platforms having the same type 2 costs (platform (un)desirability cost), in which case the type 3 cost determines the choice of platform.

(iii) Advantages of a lexicographic cost function or decision rule. These can be summarised as follows:

(a) It reflects how decisions on choices are currently made in train scheduling, at least in the UK. It is widely used in practice by train planners, though we have never heard it referred to by this name. Incidentally, it also reflects decision making in a wide range of other areas in management, outside of transportation or scheduling.

(b) A lexicographic decision rules means that data is not needed for actual or for relative costs (or benefits or utilities). Such actual or relative cost data is currently not used and is not available. In view of that, using a lexicographic cost function or decision rule appears to be the only available option.
(c) Even if actual or relative cost data were provided to the train planning authority by the various train operators, the estimates from one train operator might be disputed by another. A lexicographic cost function or decision rule may be less controversial or less disputable.

(d) If acceptable actual or relative cost or penalty data should become available, then cost functions based on this can easily be substituted for the lexicographic cost functions that we are currently using. The rest of the problem definition, and the algorithms set out later below, would remain almost unchanged.

(iv) **Lexicographic cost function with cutoffs or thresholds.** The cost function (ii) can be adapted in various ways. For example, do not assign trains to a platform with a type 2 cost (undesirability cost) above a prespecified cut-off level—treat such platforms as infeasible. Another adaption of (ii) is to use the platform occupation cost only to break ties between (sub)platforms of the same multi-train platform.

We can state the lexicographic cost function (ii) more formally. For train $t$, platform $p$ has a (lexicographically) lower cost than platform $p'$ if

\[
\begin{align*}
&\text{if } [c_{kp}^k < c_{k'p'}^k] \\
&\text{or if } [(c_{kp}^k = c_{k'p'}^k) \text{ and } (C_{kp}^u < C_{k'p'}^u)] \\
&\text{or if } [(c_{kp}^k = c_{k'p'}^k) \text{ and } (C_{kp}^u = C_{k'p'}^u) \text{ and } (C_{kp}^o < C_{k'p'}^o, P_{pp'} = 1)].
\end{align*}
\]

Otherwise platform $= p'$ has the lower cost. The three costs $c_{kp}^k$, $C_{kp}^u$ and $C_{kp}^o$ are as defined in Sections 2.3–2.5, and

\[P_{pp'} = 1 \text{ if } p \text{ and } p' \text{ are sub-platforms of the same multi-train, through platform, otherwise } P_{pp'} = 0.\]

The lexicographic cost function can be referred to as $L(C_1, C_2, \ldots, C_n)$ where $C_1$ to $C_n$ are the cost types in descending order of importance.

2.7. **Path conflicts**

Between stations, there are usually only a few parallel one-way lines in each direction—usually one, two or three. However, the immediate approach to a (multi-platform) station is usually a complex mesh of intersecting tracks, linking the incoming and outgoing lines to various platforms. To avoid confusion here, we will refer to these routes in and out of stations as “paths”, to distinguish them from the “lines” between stations. When a train approaches a station it travels on some path linking the approach line to a platform, and when departing it uses some path linking the platform to an out-line. Usually, many of these paths conflict, by crossing each other or by sharing a piece of track. For obvious reasons, trains are not allowed on conflicting paths at the same time. The existence of path conflicts can be identified in advance from a detailed map of the track layout. To represent conflicts introduce an indicator,

\[X(l, p, l', p') = \text{TRUE if the path connecting line } l \text{ to platform } p \text{ conflicts with the path connecting line } l' \text{ to platform } p', \text{ otherwise let } X(l, p, l', p') = \text{FALSE}.\]
To determine whether there is a path conflict, we do not need to know in which direction (in or out of the station) the trains are travelling, nor the types of trains. Also, it is worth noting that there is often a choice of several paths from a line to a platform, or vice versa. However, in this case we are only concerned with whether there exist non-conflicting paths. We do not need to know about alternative possible paths, or about the detailed layout of the mesh of lines in the station mouth. If there exists any path connecting line $l_0$ to platform $p_0$ and any path connecting line $l_0'$ to platform $p_0'$ such that these two paths do not conflict, then we say there is no conflict, i.e., $X(l, p, l_0, p_0') = 0$.

To illustrate, consider two trains $t$ and $t_0$ on in-lines $l = I_t$ and $l_0 = I_{t_0}$. Then $X(l, p, l_0, p_0')$ indicates whether there is a conflict between the paths of train $t$ and $t_0$ arriving on lines $I_t$ and $I_{t_0}$ respectively and going to platforms $p$ and $p_0'$ respectively. Similarly, if $l = I_t$ and $l_0 = O_{t_0}$ then $X(l, p, l_0, p_0')$ indicates whether the path of train $t$ arriving on line $I_t$ and going to platform $p$, conflicts with the path of train $t_0$ departing on line $O_{t_0}$ from platform $p_0'$.

2.8. Minimum headway constraints

If two trains are on conflicting paths we must ensure that there is at least a required minimum headway (time interval) between them, for safety and signaling reasons. The minimum headway required depends on the types of the trains, on whether the trains are arriving or departing from the station, and on the platform and line used by each train. For example, let

$H(t, p, t', p')^{da}$: the minimum headway required between train $t$ departing from platform $p$ and the next train $t'$ arriving at platform $p'$.

The superscripts $d$ and $a$ denote departure and arrival, and the order of the superscripts (and subscripts) indicates the order of the trains, i.e., train $t$ is followed by $t'$. (Note that in defining these headways we have not introduced subscripts to denote the in-lines or out-lines used by trains $t$ and $t'$. The reason is that for an arriving train $t$ the in-line is already specified by $I_t$, and for a departing train the out-line it is specified by $O_{t_0}$.)

The headway requirement depends on whether $t$ is arriving or departing, whether $t'$ is arriving or departing, and which ($t$ or $t'$) is first. Thus, by analogy with $H(t, p, t', p')^{da}$, there are eight different possible headway requirements between trains $t$ and $t'$, i.e.,

$H(t, p, t', p')^{da}, H(t, p, t', p')^{aa}, H(t, p, t', p')^{ad}, H(t, p, t', p')^{dd}$

and these another four with the subscripts $tp'p'$ replaced by $t'p'tp$.

Of course, only at most two of these eight possible headways will actually arise for any given pair of trains, since their paths will meet or cross at most twice at a station—when arriving and/or when departing.

2.9. Platform occupation constraints for multi-platform stations

Platform occupation constraints ensure that trains do not occupy the same platform simultaneously. For platforms which only hold a single train we simply need to check that the platform is not already occupied before a train travels through it or stops at it. However, the situation is more complex for multi-train platforms. This is discussed in Section 3.3.
Multi-train platforms are platforms which are permitted to accommodate two or more trains at the same time, because the platform is long or the trains are short. A platform which can handle \( n \) trains can be thought of as \( n \) separate platforms, and in practice multi-train platforms are often marked off into separate segments, labeled say platforms 4a, 4b, etc. Trains going to or from any one of these \( n \) sub-platforms have to be checked for path conflicts and minimum headways in exactly the same way as trains to “normal” single train platforms. Hence, we can deal with multi-train platforms by extending the list of platforms to include each of the sub-platforms as a separate platform. This suffices when checking for path conflicts or headways.

However, multi-train platforms cause other problems. Sub-platforms are simply segments of a multi-train platform. Hence, when arriving or departing from a sub-platform, a train may have to pass through other sub-platforms, and for this to be possible we have to ensure these other sub-platforms are currently empty. The set of sub-platforms through which a train has to pass depends on which end of the multi-train platform the train arrives at, or departs from. We will arbitrarily let “+” denote one end of the multi-train platform, and let “−” denote the other end (e.g., at Leeds call the West end “−” and the East end “+”, at York call the South end “−” and the North end “+”). To model the movements of a train to and from multi-train platforms we introduce the following notation. Let,

\[
\begin{align*}
M_p^{−} & \text{: the set of sub-platforms which lie between sub-platform } p \text{ and the “−” end of the multi-train platform. That is, } M_p^{−} \text{ is the sub-set of sub-platforms which block arriving or departing trains travelling between sub-platform } p \text{ and the “−” end of the multi-train platform. If sub-platform } p \text{ lies at the “−” end of the multi-train platform, or if the multi-train platform is a terminal platform with access from the “+” end, the set } M_p^{−} \text{ will be empty.} \\
M_p^{+} & \text{: the set of sub-platforms which lie between sub-platform } p \text{ and the “+” end of the multi-train platform. If sub-platform } p \text{ lies at the “+” end of the multi-train platform, or if the multi-train platform is a terminal platform with access only from the “−” end, the set will be empty. Let,} \\
L^{−} & \text{: the set of lines in or out of the “−” end of the station.} \\
L^{+} & \text{: the set of lines in or out of the “+” end of the station.}
\end{align*}
\]

The set of platforms which block a train’s arrival or departure depends upon which side of the station it is traveling in. For example, if train \( t \) arrives on in-line \( I_t \in L^{−} \), or departs on out-line \( O_t \in L^{−} \), the set of sub-platforms blocking its arrival or departure at platform \( p \) is \( M_p^{−} \).

For brevity we may sometimes (loosely) refer to trains travelling in a “+” direction, as shorthand for travelling to or from platforms which lie between sub-platform \( p \) and the “+” end of the multi-train platform. Similarly, we refer to trains travelling in a “−” direction.

3. A scheduling algorithm

At busy railway stations trains arrive and depart on average every few minutes throughout the day. This is true, for example, at a dozen stations around London and at the main stations in most cities in Britain. The times of these have to be scheduled to be allowed at least minimum headways, turnaround times, and dwell times, while ensuring they all arrive and depart without
conflicting with the times or platforms of any other trains arriving, departing or waiting at the station. The data required for train platforming is set out above in the discussion of constraints and costs.

3.1. Outline of the algorithm steps

The algorithm considers one train at a time, and finds a provisional or final arrival time, departure time and platform for that train. It then adds the train to a list of temporarily or permanently ‘fixed’ (assigned) trains. As well as trains which are fixed during the course of the algorithm, some trains may have their times or platforms fixed before the algorithm begins. When all trains have been considered and scheduled, the algorithm can cycle back to reconsider previously fixed trains and seek further improvements. However, we found that the algorithm gave good results even without this.

The algorithm considers each train \( t \) separately. For each train, it considers each feasible platform and for each of these platforms it checks whether any headway or platform conflicts exist between the train \( t \) and each train already scheduled temporarily or permanently at the station. These conflicts are discussed in Section 2 above. If a conflict exists the arrival and/or departure times of train \( t \) are increased to a point where there is no longer a conflict. However, this time adjustment may introduce new conflicts with trains which did not previously conflict with train \( t \). A new set of trains may therefore have to be checked or re-checked for conflicts with train \( t \). This iteration between conflict checking and making time adjustments continues until a time slot is found where the train \( t \) can be assigned without conflicts to the proposed platform. The costs of assigning the train to this platform in this time slot are then calculated. (The costs consists of the knock-on, desirability and platform obstruction costs discussed in Sections 2.3–2.5. Ways of combining these costs are discussed further in Section 2.6). This whole process is repeated for each feasible platform for train \( t \), until a platform is found which yields the lowest costs or penalties. Train \( t \) is then assigned to that platform, and is added to the list of temporarily or permanently assigned trains. The algorithm proceeds to the next train in the list of non-fixed trains to be assigned. This constraint checking and time adjustment processes are discussed further in the following sections.

To resolve conflicts, in this paper we delay trains rather than advance them, for two related reasons.

(a) First, when scheduling trains we will usually consider them in chronological order of their draft or desired arrival/or departure times. In that case, adjusting the times of a train forward is less likely to cause further conflicts than moving it back in time. We observed that train planners using traditional manual methods also behave in this way, that is, they consider trains in chronological order and normally delay trains rather than advanced them. (Nevertheless, in further research we are investigating resolving conflict by advancing trains as well as delaying trains.)

(b) Second, the draft or desired train times in draft train plans in Britain tend to be the earliest desired times. Even if a desired train time is in the middle of a time window of acceptable times, we can start the scheduling process with the train set to its earliest acceptable times, so that only time delays are allowed.
3.2. Finding and resolving headway conflicts

Headway constraints ensure that if a pair of trains are on conflicting paths there is a sufficient time gap between them. There are eight types of headways, as set out in Section 2.8. For each type the algorithm performs a similar set of checks for conflict and adjustments to resolve any conflicts found. First, it checks if the pair of trains \( t \) and \( t_0 \) are on conflicting paths, and if they are it checks if the headway between them is sufficient. If the headway is insufficient then the arrival and/or departure times are adjusted so that it arrives and/or departs after the conflict point, separated from it by a suitable (minimum) headway.

For example, suppose we are attempting to assign the current train \( t \) to platform \( p \). One of the possible conflicts is: if the path of train \( t_0 \) conflicts with the path of train \( t \) (i.e., if \( X(I, p, I', p') = \text{TRUE} \)) and if train \( t_0 \) arrives after the arrival time of train \( t \) but with insufficient headway between them (i.e., if \( A_{tp} < A_{tp_0} < A_{tp} + H(I, p, I', p')^{aa} \)), that is,

\[
\text{if } X(I, p, I', p') = \text{TRUE} \quad \text{and} \quad (A_{tp} < A_{tp_0} < A_{tp} + H(I, p, I', p')^{aa}) \]

If this is true (i.e., if there is a conflict), then move the arrival time for train \( t \) to after the arrival time of train \( t_0 \), including the required minimum headway \( H(I, p, I', p')^{aa} \), and set a new departure time for train \( t \) by simply adding the required dwell time to the new arrival time. That is

\[
\text{set } (A_{tp} = A_{tp_0} + H(I, p, I', p')^{aa} \text{ and } D_{tp} = A_{tp} + W_{tp}^{des})
\]

In summary, finding and resolving the above conflict can be stated as

\[
\text{if } X(I, p, I', p') = \text{TRUE} \quad \text{and} \quad (A_{tp} < A_{tp_0} < A_{tp} + H(I, p, I', p')^{aa})
\]

\[
\text{then set } (A_{tp} = A_{tp_0} + H(I, p, I', p')^{aa} \text{ and } D_{tp} = A_{tp} + W_{tp}^{des})
\]

A similar process is repeated for each of the other types of headway conflicts. There are eight possible types of headway conflict between trains \( t \) and \( t' \), as indicated by the eight types of headway in Section 2.8. For each of these the conflict check is set out in column 3 of Table 1. If this conflict check proves positive then resolve the conflict by making the changes shown in column 4.

We can reduce the number of conflict checks by ruling out some of the above conflict checks in some iterations of the algorithm (see Section 4.4).

3.3. Enforcing platform occupation constraints

3.3.1. Single-train platforms

A platform occupation conflict arises if the platform proposed for train \( t \) is already assigned to another train for all or part of the proposed dwell time of train \( t \). Obviously this can occur even if there are no arrival or departure headway conflicts with train \( t \). Suppose the proposed platform for train \( t \) can accommodate only a single train at a time. Then a platform occupation conflict between train \( t \) and another train \( t' \) can be overcome by adjusting the arrival time of \( t \) forward so that it arrives after train \( t' \) has departed, allowing a minimum headway between the two trains. However, we first require a method of checking and recording whether, at the proposed platform, there is a platform occupation conflict with train \( t' \). To record this a temporary variable \( B_t \) is used where,
B_t = TRUE if any train t’ is present at platform p for any part of the proposed time slot of train t at platform p.

To calculate B_t we have to make three separate checks:

Does t’ arrive at platform p before t arrives and depart after t arrives?

Does t’ arrive at platform p before t departs and depart after t departs?

Does t’ arrive at platform p after t arrives and depart before t departs?

More formally:

B_t = TRUE if \( p = p_t \) and \([A_r \leq A_{ip}^* \leq D_r]\) or \((A_r \leq D_{ip}^* \leq D_r)\) or \((A_{ip}^* \leq A_r \text{ and } D_r \leq D_{ip}^*)\).

Otherwise B_t = FALSE.

Note that more than one train t’ may use platform p during the proposed dwell time of train t, hence we record B_t rather than a single B. Then we can find and resolve platform occupation conflicts as follows:

for all \( t' \in T_p \), if \((B_t = TRUE)\) then set \((A_{ip}^* = D_r + H(O_{t'}, p, I, p)^{da})\) and \(D_{ip}^* = A_{ip}^* + W_{ip}^*\).

In the platforming algorithm, this temporary variable B_t has to be updated each time the arrival and/or departure times of train t is changed at the current platform p.

### 3.3.2. Multi-train platforms

For multi-train platforms the situation is more complex. In addition to the above check (that platform p is unoccupied), we must also check that the entrance and exit routes to the sub-platform p are clear for the arrival and departure of train t. If they are not clear, then the arrival and/or departure times must be adjusted to a point where the conflict no longer exists. There are
four types of conflicts, namely conflicts due to the train $t$ arriving at the “$-$” side of the station, arriving at the “$+$” side of the station, departing from the “$-$” side of the station or departing from the “$+$” side of the station. Take for example the arrival of train $t$ on an in-line $I_t$ on the “$-$” side of the station. The algorithm first checks if the in-line is on the “$-$” side of the station. If it is, then the set of platforms $M^-_p$ block access to the proposed platform $p$, and if any of these are occupied by a train $t'$ the arrival time of $t$ will have to be adjusted to a point after the departure of train $t'$, allowing a minimum headway. The departure time is adjusted by a similar amount so that the dwell time remains constant, as follows:

$$
\text{if } (I_t \in \mathbb{L}^-) \text{ then for all } p' \in M^-_p \text{ and all } t' \in T_{p'}: \text{ if } B^a_{t'} = \text{TRUE set } (A^*_{ip} = D_t + H(O_t, p', I_t, p)^{da} \text{ and } D^*_{ip} = A^*_{ip} + W_{ip}).
$$

where the indicator variable $B^a_{t'}$ is defined as

$B^a_{t'} = \text{TRUE if train } t' \text{ is present at platform } p' \in M_p \text{ at arrival time of train } t \text{ at platform } p.$

which can be computed from,

$$B^a_{t'} = \text{TRUE if } [p'_i \in M_p \text{ and } (A_{t'} \leq A^*_{ip} \leq D_{t'})], \text{ otherwise } B^a_{t'} = \text{FALSE}.$$

Repeat this process for each of the other three types of conflicts which can exist. Thus we have:

(i) if $(I_t \in \mathbb{L}^-)$ then for all $p' \in M^-_p$ and all $t' \in T_{p'}$: if $B^a_{t'} = \text{TRUE set } (A^*_{ip} = D_t + H(O_t, p', I_t, p)^{da}$

and $D^*_{ip} = A^*_{ip} + W_{ip}^{des});$

(ii) if $(I_t \in \mathbb{L}^+)$ then for all $p' \in M^+_p$ and all $t' \in T_{p'}$: if $B^a_{t'} = \text{TRUE set } (A^*_{ip} = D_t + H(O_t, p', I_t, p)^{da}$

and $D^*_{ip} = A^*_{ip} + W_{ip}^{des});$

(iii) if $(O_t \in \mathbb{L}^-)$ then for all $p' \in M^-_p$ and all $t' \in T_{p'}$: if $B^d_{t'} = \text{TRUE set } (D^*_{ip} = D_t + H(O_t, p', O_t, p)^{dd}$

and $W_{ip} = D^*_{ip} - A^*_{ip});$

(iv) if $(O_t \in \mathbb{L}^+)$ then for all $p' \in M^+_p$ and all $t' \in T_{p'}$: if $B^d_{t'} = \text{TRUE set } (D^*_{ip} = D_t + H(O_t, p', O_t, p)^{dd}$

and $W_{ip} = D^*_{ip} - A^*_{ip}).$

The indicator variable $B^d_{t'}$ is similar to $B^a_{t'}$. That is

$B^d_{t'} = \text{TRUE if train } t' \text{ is present at platform } p' \in M_p \text{ at departure time of } t \text{ from platform } p.$

which can be computed from,

$B^d_{t'} = \text{TRUE if } [p'_i \in M_p \text{ and } (A_{t'} \leq D^*_{ip} \leq D_{t'})], \text{ otherwise } B^d_{t'} = \text{FALSE}.$

3.4. The basic algorithm

The algorithm described throughout the previous section can now be set out more formally as follows. First we introduce one further item of notation.

$f_i = 1$ if the arrival and departure times of train $t$ are fixed, otherwise 0. This is initially set to 0 for all trains, unless we wish to fix the arrival and departure times of some trains in advance, in which case set $f_i = 1$ for those trains.
Algorithm A1

0. **Initialisation.** Set $t = 0$, $t' = 0$, $p = 0$. Set the draft arrival, departure and dwell times of each train $t$ to the desired times (see Section 2.1), that is, for all $t$ and $p$ set $(A^*_t = A^{des}_t, D^*_t = D^{des}_t, W^*_t = W^{des}_t)$.

1. **Introduce the next train $t$** to be considered: i.e., the next train for which we have to find the least cost feasible platform. E.g., set $t = \text{next non-fixed train } (f_i \neq 1)$ in chronological order of draft arrival times, but also see Section 4.4 for alternative rules. If $t > \text{last train } T$, go to Step 9.

2. **Introduce the next feasible platform** to be considered for train $t$.
   (a) Introduce the next platform: e.g., set $p = p + 1$, but also see Section 4.4 for alternative rules. If $p = \text{last platform } P$, go to Step 9.
   (b) If the platform is not feasible for train $t$ (if $F_{tp} \neq 1$) return to (a).

3. [Steps 3–7 are nested under Step 2 and comprise finding and resolving conflicts for train $t$ at platform $p$.]
   - Record the current times of train $t$: $A^{old}_t = A^*_t, D^{old}_t = D^*_t, W^{old}_t = W^*_t$.
   - Choose the first train $t_0$ to examine for conflicts with train $t$: e.g., set $t_0 = 0$, but also see Section 4.4 for alternative rules.

4. [Steps 4–6 are nested under Step 3 and comprise finding and resolving conflicts between trains $t$ and $t_0$ at platform $p$.]
   - Introduce the next train $t_0$ to examine for conflicts with train $t$, see Section 4.4 below. E.g., if train $t_0 < \text{last train } T$, set $(t = t_0 + 1)$.

5. (a) **Find and resolve headway conflicts.** Check for headway conflicts between train $t'$ and the current trains $t$ and resolve these conflicts by incrementing the timings of train $t$. That is, perform the checks (i)–(viii) from column 3 of Table 1 and resolve any conflicts as in column 4 of that table.
   (b) **Find and resolve platform occupation conflicts.**
      (ix) Find and resolve platform occupation conflicts for single-train platforms as in Section 3.3.1.
      (x) Find and resolve platform occupation conflicts for multi-train platforms as in Section 3.3.2.

6. **Check if the arrival or departure times of train $t$ have been changed** (knocked-on). If they have not been changed, introduce the next train $t'$ to examine for conflicts with train $t$, i.e., if $(A^*_t = A^{old}_t)$ and $(D^*_t = D^{old}_t)$ then return to Step 4.
   If they have been changed then re-check for conflicts between $t$ and all other trains $t'$, i.e., if $(A^*_t > A^{old}_t)$ or $(D^*_t > D^{old}_t)$ return to Step 3.

7. **Compute the time adjustment or knock-on cost** of assigning the current train $t$ to current trial platform $p$ (see Section 2.5), e.g.,
   $c^k_{tp} = c^a_t(A^*_t - A^{des}_t) + c^d_t(D^*_t - D^{des}_t) + c^w_t(W^*_t - W^{des}_t)$.
   If platform $p < \text{last platform } P$, return to Step 2.

8. **Find the lowest cost platform** for the current train $t$. We come to Step 8 only after we have cycled through Steps 2–7 for all platforms $p$ for train $t$. In Steps 2–7, and in particular in Step 7, we have found, for each platform $p$, the knock-on delay costs $c^k_{tp}$ that would be incurred if
we assigned train \( t \) to that platform. Then, in Step 8, to find which is the best of these platforms for train \( t \), use the knock-on costs \( c_{tp}^k \) from Step 8 together with the cost data \( C_{tp}^u \) and \( C_{tp}^o \), and combine these costs using the (lexicographic) cost function from Section 2.6. Let \( \bar{p} \) denote the resulting lowest cost platform for train \( t \).

Assign train \( t \) to the lowest cost platform \( \bar{p} \). That is, set \( p_t = \bar{p} \), set arrival, departure and dwell times \( A_t = A_{tp}^* \), \( D_t = D_{tp}^* \) and \( W_t = W_{tp}^* \), and add train \( t \) to the set of fixed trains, i.e., set \( f_t = 1 \).

If train \( t < \) last train \( T \), return to Step 1.

9. End.

4. Computational tests and results, and variants/extensions of the algorithm

To test and demonstrate the above model and algorithms, we applied them to scheduling for some stations and we here report some of the results for one station. To enable this testing, we programmed the algorithms as a set of modules in the C programming language using Borland C and ran this on a Pentium PC under Windows 95.

It would be desirable to numerically compare or benchmark the heuristic algorithms in this paper against other algorithms that have been applied to this problem. However, as discussed in the introduction, there are no other special purpose algorithms available for this particular problem. An alternative would be to benchmark them against the present methods, in which the scheduling is done ‘manually’. However, this was not possible since the rail authority in Britain (Railtrack) considers the draft train plans as confidential. This is partly because they contain draft proposals for train services and timetabling from various train operators who are in competition with each other. However, the basic version of our heuristic algorithm largely consists of our formalisation of the existing manual methods that are used by train planners/schedulers in Railtrack. The manual method consists of rules and heuristics evolved over decades and we elicited them by observing and debriefing train planners over several days, spread over a few years. In that sense the present methods are benchmarked into our heuristic algorithm. This enables solutions to be obtained similar to those from existing manual methods (see Example 1) but also allows the heuristics to be extended in ways that are not practicable using manual methods. Further reasons for adopting this approach were set out in the introduction.

4.1. Data used in computational testing

We choose Leeds station (see Fig. 1) in the north of England as a test train station and the relevant detailed data for Leeds station is set out in Appendix A. We choose this station since its layout, mixture of platform types and trains types included most of the complexities which are met with in practice, thus ensuring that the scheduling algorithms would have to handle all of these features. Leeds station is accessed via six lines, four at the West end and two at the East end. At each end of the station these lines form a complex mesh of crossing lines and hence headway conflicts. There are 12 main platforms which are divided into sub-platforms so that they can, and commonly do, accommodate more than one train at the same time. There are 34 (sub)platforms in all. Seven of the main platforms are terminal platforms and five are through platforms which can
be used in either direction. The trains have minimum required dwell times ranging from a few minutes to 45 min.

Leeds is a relatively busy station with about 900 arrivals and departures/day—more than one per minute in rush hours. It has a mixture of train types and speeds both local and long distance from several TOC’s. Each of these companies is potentially in competition for train time slots and platforms at the station, and trains from each operating company have preferred or required platforms.

4.2. Results

Some of the results obtained by applying the scheduling Algorithm A1 to the above network and data are as follows.

4.2.1. Example 1

Though initial draft timetables were not available, we obtained a draft timetable in which many of the time conflicts had already been eliminated using the ‘manual’ method. This timetable is still useful for benchmarking purposes, since we can ignore the allocation of trains to platforms obtained by the manual method, use our Algorithm A1 to assign the trains to platforms and assign times for these, and compare the results with those from the manual method. When we did this we expected to find few time conflicts, since most had been removed the traditional method. However, the algorithm found a significant number of conflicts and resolved these by making small adjustments to the arrival times of 10 trains and to the dwell times of seven trains (out of 490 trains). The ‘conflicts’ consisted mainly of headways smaller than the specified minimum headways. We checked with Railtrack to see why this occurred. Apparently they had allowed the lower headways to avoid having to change certain desired train times. To reflect this, in later versions of the algorithm we introduce absolute minimum headways as well as the desired minimum head-
ways, and allow the former to be used only if this is needed to resolve a conflict or reduce further delays to an already delayed train.

Though the algorithm did not significantly change the times from the Railtrack timetable, it changed the main platform assignment for about 20% of trains, and the sub-platform assignment for about another 20%. Only a minority of these changes were due to resolving conflicts referred to above. The reasons for the other changes are of interest since they are likely to arise quite widely, not only in train or transport scheduling, when comparing results from automated scheduling with results from by-hand scheduling. The main reasons for the changes were as follows. First, the platform preferences that we obtained from Railtrack often specified groups of platforms as being equally desirable; e.g., platforms 5, 6 and 8 are equally 'most desirable' for intercity trains. In that case the Railtrack scheduler may choose say platform 6 for a train while our algorithm may choose the equally desirable platform 5. Second, the Railtrack preferences among platforms were sometimes not clear or varied among Railtrack schedulers. The platform preference data that we used were elicited from discussions with individual schedulers, and are estimates of largely unwritten, changing and somewhat discretionary rules. This is particularly true for preferences among sub-platforms of the same main platform. As a result, some of these preferences or rules are likely to differ from those actually applied by some Railtrack schedulers, hence they yield a different platform assignment.

4.2.2. Example 2

Given the above difference between our platform assignment and that chosen by Railtrack, as a further experiment we re-ran the programme while forcing the trains to go to the platform assigned to them in the Railtrack timetable. Since this means we cannot change platforms to avoid conflicts, we expect that the number and sizes of conflicts in this example will be larger than in Example 1. As expected the number of conflicts increased, and more than doubled the number of trains that had their times changed from the Railtrack draft times.

4.2.3. Example 3

When a train schedule has been adopted and published, operators try to run trains according to this timetable. However, in day-to-day train running there are exogenous delays to trains due to many causes (breakdowns, failures, personnel, passenger boarding, etc.). To emulate this, we added random disturbances to the final schedule obtained in Example 1: we choose random disturbances typical of those that occur in practice in day-to-day running of trains. Let \( S \) denote the schedule obtained in Example 1, and \( S + e \) denote this schedule with exogenous random delays or disturbances \( e \) added.

Schedule \( S + e \) may not be feasible, and is likely to cause further knock-on delays, hence further adjustments of the (arrival, dwell and departure) times may be necessary. We assume here that operators will wish to wish to minimise further deviations from the timetable, subject to other constraints and preferences. In doing this, they give different weights, preferences or costs to deviations from the timetable. To emulate that, we take \( S + e \) as the new draft or desired timetable and apply Algorithm A1 to generate a new acceptable feasible timetable.

To introduce disturbances, to the arrival time of each train we added an exogenous delay selected at random (uniformly distributed) between \(-2\) and \(20\) min. Similarly, to the dwell times of trains we added exogenous delays uniformly randomly distributed between \(-2\) and \(10\) min.
We assume that these exogenous delays cannot be eliminated, hence will be present in any rescheduling. The purpose of scheduling is then to manage or reduce the size, costs or spread of additional knock-on delays. When we applied Algorithm A1 we found that in the resulting schedule an additional 22% of trains had knock-on arrival delays, 13% had knock-on dwell delays, and 29% had knock-on departure delays. A departure delay is caused by an arrival delay or a dwell delay or both. Hence \( (22 + 13) - 29 = 6\% \) had both types of departure delay. The knock-on delays could be reduced, as mentioned above, by allowing late trains to depart after less than their normal desired dwell time. This is particularly true of the dwell and departure delays and to a lesser extent arrival delays. We explore this and other operating policy options elsewhere.

4.2.4. Further examples and experience

In ongoing work we have performed a large amount of additional testing and experimenting with the above model and algorithms. There is not space to report this here, but much of it is outlined in Sections 4.4 and 5 below and will be reported in later papers. For example, we have experimented with adapting or adding various steps or heuristics in the algorithm, and we have used it to simulate on-the-day train running. The latter involved re-running the algorithms tens of thousands of times to reschedule trains following random delays or disturbances similar to those typically found in practice. In all of this the algorithms have performed well and have found good solutions very quickly.

4.3. Analysis of the schedules produced

As with most schedulers, train planners would like to have not just a schedule but also a battery of measures of the quality of the schedule. Hence we developed an output analyser to analyse the generated schedule and print out a range of statistics which include:

**Changes from desired or draft departure, dwell or arrival times:** A list of all trains that have their arrival, dwell or departure times adjusted, and the mean, median, average deviation, standard deviation, etc., of these time adjustments. The number and percentage of trains that have arrival, dwell or departure times adjusted by more than 1, 2, 5, 10, 15, ..., min. All of these statistics are calculated separately for arrival, dwell and departure time adjustments, as well as in aggregate. Also, all of the above measures are repeated for each type or speed of train taken separately, or for trains of each TOC.

**Platform adjustments:** The number and percentage of trains which are assigned to their first preference platforms, second preference, etc. The platform penalties or costs incurred by each train.

**Platform usage:** Percentage daytime utilisation of each platform. Analysis of actual headways or time gaps between trains at each platform. That is, the mean, median, average deviation, standard deviation, 10–90 percentiles, etc., of these.

The above statistics very useful for exploring the effects of changes in scheduling constraints, data or policy. For example, exploring effects of changes in minimum headways or dwell times, or restrictions on platform usage, or say allowing ‘late’ trains to depart before the end of their minimum dwell time. They also proved very useful in diagnosing errors and in algorithm development.
4.4. Variants of the algorithm, and computing times

We experimented with several variants of the algorithm set out in Section 3. In particular we tried different variants of the rules in Steps 1–4, as discussed below. The speed with which the programme executed Algorithm A1 depended very much on the exact rules used in these steps of the algorithm.

(i) In Step 1, the rule used for choosing the next train \( t \) to consider for scheduling.

There are a few standard rules used in practice by train operators to decide the next train to assign. The most common rule is to consider the trains in chronological order of their initial desired or draft arrival or departure times. However, if there are train conflicts, the trains considered first may have an advantage. Train planners often exploit this by first scheduling all the most important trains (where importance may be defined by train speed, or expected revenues, or some other criterion), then scheduling the next most important class of trains, and so on. This process allows the more important to have first choice of the available time slots. For example, first schedule express trains, then other inter-city trains, then regional fast trains, then regional stopping trains, then freight trains.

When scheduling trains within each of the above categories, train planners usually consider the trains in chronological order of their initial desired or draft arrival or departure times. Our computational results tend to support the above rules. That is, we found that if trains have very different cost weights, say delays to intercity trains cost much more than delays to freight trains, then scheduling all intercity trains first, in chronological order of desired arrival times, produces better results (lowest weighted average delay or cost per train). On the other hand, we find that if all trains are given equal weight then considering all trains in chronological order of their draft arrival times tends to produce the best results. Also, we found that the order in which trains are considered tended to affect only the next few trains, or next several trains, and there were seldom long chains of knock-on effects. We are exploring all of these issues in later papers. There we have experimented with the above train orders, and also other orders such as random order, reverse chronological order, platform preference order, and have not found that these produce superior results.

The order in which trains are considered for scheduling can affect the algorithm computing time, however, the effect of scheduling order on the quality of the schedule (delays of costs) is much more important, so that, for this part of the algorithm, computing time is a secondary issue.

(ii) In Step 2 (a), the rule used in choosing the next platform \( p \) to consider for train \( t \).

When we tried platforms in numerical order, or in random order, the algorithm was very much slower than when we tried them in platform preference order. When we tried platforms in preference order we frequently found that the first or second platform tried had no costs or penalties, and caused no conflicts, hence no delays, so that the search for a platform could stop. In contrast, if we searched in numerical order then even if there were no conflicts at any platform, we had to keep searching until the highest preference platform was found.

(iii) In Steps 3 and 4, the rule used in choosing the next train \( t' \) to check for conflicts with train \( t \) at platform \( p \).

When searching for conflicts with train \( t \), the initial version of the algorithm checked for headway conflicts with all other trains \( t' \). In later versions we more sensibly checked for headway conflicts only with trains arriving or departing within time windows around the arrival or
departure times of train \( t \). Other trains could of course still have platform occupation conflicts with train \( t \), which we still checked. However, even platform occupation conflicts could not occur if a train arrived more than a few minutes after train \( t \) departed, or departed more than a few minutes before \( t \) arrived, hence such trains could be ignored in conflict checking.

The impact on computing times from changing the above rules (i)–(iii) was greater than might be thought. The reason is, these steps have to be repeated an enormous number of times. For example, every time a trial change of train time is made at a trial platform for train \( t \), this may put \( t \) back in conflict with some trains with which conflicts had previously been resolved. Hence if we change the time or platform of train \( t \) we have to recheck it for possible conflicts with all trains.

As we progressively improved the above search rules, the computing times fell. They started out at several hours, then reduced to a few hours, then to minutes, then down to a few seconds. There is not space to adequately discuss these and other improvements here, but it will be presented elsewhere.

5. Concluding remarks and further research

In this paper we set out and discussed the train platforming problem, that is, the problem of assigning trains to time slots at platforms, so as to satisfy various headway conflicts, platform occupation constraints, etc. We developed and tested an algorithm for solving the problem subject to the various constraints and objectives. By introducing various search strategies in the steps of the algorithm we were able to make dramatic reductions in the computing time of the algorithm, until it ran in several seconds. Also, in our algorithms, we are able to incorporate the rules, costs, preferences and trade-offs which are used by expert train planners. Further, with a computer based system we are able to do this much faster, and take account of additional options and information that it is not feasible for train planners to consider when using traditional manual methods. This research demonstrates that the heuristic search strategies which we used are suitable for solving the train scheduling problem.

In further work, we have taken the schedules produced by the algorithms of the present paper and tested their reliability and robustness (Carey and Carville, 2000). To do that, we added exogenous delays or disturbances to the arrival, dwell and departure times, the delays being drawn from distributions of exogenous random delays that typically occur in practice in day-to-day train running. These delays introduce train time and platform conflicts, hence we again used the algorithms from the present paper to find and resolve the conflicts, which typically introduces additional, knock-on, delays. By repeating this process thousands of times we simulated the patterns of delays and platform changes in day-to-day train running.

We are extending this research in several other ways, to be presented in later papers. We are introducing and testing additional rules and strategies designed to further speed up the algorithms, and produce better solutions. As noted in Section 4.4 above, the exact form of some steps in the algorithm has a very substantial effect on the computing time.

Second, though the simulations in Carey and Carville (2000) are of interest, they test the reliability and robustness of the schedule only after it is produced, and are therefore not of much direct help in designing more reliable schedules. To address the issue of designing more reliable schedules,
we are introducing additional heuristics into the algorithms of the present paper. For example, when choosing a platform for a train, if the present heuristics find little or no difference between two platforms, assign the train to the platform which is free longest. This means that if there are on-the-day delays, the train is less likely to conflict with other trains at the platform. To test the reliability of the resulting schedules we simulate running them with and without the additional reliability heuristics.

Third, we are extending the present single station models and algorithms to a series of such stations. Scheduling trains at busy stations is typically the most complex component of train pathing and scheduling, but a train schedule usually involves several such stations and the (one-way) lines between them. Between stations there may be a choice of lines and we have to ensure sufficient headways between trains travelling at different speeds on each line. In view of this, in Carey and Crawford (2000, 2001) we consider such a scenario, and use the algorithms from the present paper to schedule the trains at the stations en route.

Acknowledgements

The authors wish to thank two referees for their very helpful comments and suggestions. This research was funded by a UK Engineering and Physical Science Research Council (EPSRC) grants GR/H/48033 and GR/K/75798 which we gratefully acknowledge. The authors also wish to thank various people at the former British Rail, Railtrack and the rail industry in general for their assistance and advice.

Appendix A. Train and platform data for Leeds station

In Section 4 we tested the train scheduling algorithms from Sections 2 and 3 by applying them to timetabling and platforming the trains for Leeds station. The data for Leeds station used in these tests is set out below. This data is also set out in more detail with more explanation and interpretation in Carey (1995). The origin of most of the data is as follows: headways and dwell times from British Rail (1994a) and from some British Rail train planners, draft Working Timetables from British Rail (1994b,c), and feasible line-platform connections and line conflicts from the station diagrams in Quail Map Company (1996).

\[ p = 1, \ldots, P, \] the set of platforms. There are 12 main platforms, numbered 1–12. These are divided into sub-platforms, so that the full list of platforms is: (1A, 1B), (2A, 2B, 2C), (3, 3A, 3B), (4, 4A, 4B), (5, 5A, 5B, 5C), (6, 6A, 6B), (7, 7A, 7B), (8, 8A, 8B, 8C), (9, 9AB, 9B, 9C), (10, 10A, 10B), 11, 12. Most main platforms can used as single platforms (e.g., 5) or as sub-platforms (e.g., 5A, 5B, 5C), but some (e.g., 1A and 1B) are currently always used as sub-platforms.

\[ t = 1, \ldots, T, \] the set of trains to be scheduled. For the set of trains we used an old British Rail draft timetable for Leeds station. This has 491 trains and about 900 arrivals and departures: some early morning trains are already in place and some trains late in the day do not depart again.
[For each train, this timetable lists the station arrival and departure times (from a stationary position), the train origin and destination, the departure time from the origin, the arrival time at the destination, the platform used, and the train ID.]

\(I_t, O_t\) = line on which train \(t\) arrives (in-line) and departs (out-line) respectively.

All trains to the same destination normally use the same line out of the station, and we got a list of these line destination combinations from BR. Similarly, for all trains from the same origin. We divided all train arrivals and departures by origins and destinations, hence identified the in-lines and out-lines used. We adhered to these lines in our tests of the scheduling algorithms.

\(A_i^{\text{des}}, D_i^{\text{des}}\) and \(W_i^{\text{des}}\) = the earliest desired arrival, departure and dwell times respectively of train \(t\).

For the desired arrival and departure times we used the timetable used above in defining the train set \(t = 1, \ldots, T\). To make the task of the scheduling algorithm and programme somewhat more difficult, we added some random disturbances to these times. For the desired dwell times we used two different methods.

Method 1: set, \((\text{desired dwell time}) = (\text{desired departure time}) - (\text{desired arrival time})\).

Method 2: set, the desired dwell times equal to the minimum dwell times below.

**Minimum dwell times at station:** RR trains: 3 min (or 2 if necessary) for through trains, 10 min (or down to 4 if necessary) for terminating trains.

Intercity: 3 min for (East Coast and Cross-Country) through trains, 30 for ECML terminating trains, 30–50 min for East Coast and Cross-Country depending on destination and on train type.

Dwell times (turnaround times) for terminating trains include time needed for any cleaning, re-stocking, watering, etc.

\(X(l, p, l', p') = \text{TRUE}\) if the path connecting line \(l\) to platform \(p\) conflicts with the path connecting line \(l'\) to platform \(p'\), otherwise let \(X(l, p, l', p') = \text{FALSE}\).

To see which lines conflict, we used the same procedure used in Railtrack, and formerly in BR. That is, we have used the detailed diagram of the station and junction layout (see Fig. 1) as follows. If on this diagram it is possible to path two trains in or out of the station without a conflict then we assume there is no conflict between them. Otherwise there is a conflict. There is one line conflict which does not show up on the diagram (Fig. 1): due to platform ‘overlap’ there is a conflict between any arrival at platform 5 followed by an arrival at platform 6 (3 min headway required).

\(H(t, p, t', p')^{\text{da}}, H(t, p, t', p')^{\text{aa}}, H(t, p, t', p')^{\text{ad}}, H(t, p, t', p')^{\text{dd}}, \text{etc.}\) Headways for line conflicts. Headways are needed only for trains on lines which conflict, as indicated by \(X(l, p, l', p')\).

Headways are reduced if necessary. Headways for all other combinations of arrivals and departures: 3 min or 4 min, but can be reduced to 3, 2.5 or 2 min if really necessary.

Headways on lines in and out of the station: 3 min in all cases, except 2.5 min following short trains on lines c and d (to West).

\(f_i = 1\) if the arrival and departure times of train \(t\) are fixed in advance, otherwise 0. In most tests we let the times of all trains be non-fixed.

\(P_{pfp} = 1\) if \(p\) and \(p'\) are sub-platforms of the same multi-train, through platform, otherwise \(P_{pfp} = 0\). We constructed this from the platform data above. E.g., \(P_{5A} = 1\) if \(p\) is platform 5A and \(p'\) is platform 5B or 5C.
$C_{t}^{a}, C_{t}^{d}$ and $C_{t}^{w} = \text{the cost per minute of deviation of the arrival, departure and dwell times } A_{ip}^{*}, D_{ip}^{*}$ and $W_{ip}^{*}$ from the desired times $A_{ip}^{\text{des}}, D_{ip}^{\text{des}}$ and $W_{ip}^{\text{des}}$. (These costs are used in Section 2.5 to compute the total time adjustment costs $C_{ip}^{k}$. We initially set each of these to 1.

$C_{ip}^{0} = \text{the platform occupation cost of assigning train } t \text{ to platform } p$.

At Leeds station, some of the multi-train platforms have two sub-platforms A and B, and some have three sub-platforms A, B and C. To generate obstruction costs for these sub-platforms, we used the penalty method suggested when we defined these costs in Section 2.4. We then multiplied each of these by penalties (0, 1, 2) by $w = 2$, where $w$ is as in the data for $C_{ip}^{u}$.

$C_{ip}^{u} = \text{the undesirability cost or penalty incurred by train } t \text{ if assigned to platform } p$.

Instead of constructing a different cost $C_{ip}^{u}$ for every train $t$, we group the trains into types. If a set of trains have the same origin, destination, arrival direction, departure direction, and train operator then we assume the same cost $C_{ip}^{u}$. The costs used are shown in Carey (1995) and were based on discussions with British Rail train planners. The costs range from 0, for the most preferred platforms, through $w/6, w/3, w/2, w, 3w/2 \text{ to } 20w$. We set $w$ to 24.

To combine the costs $C_{ip}^{k}, C_{ip}^{u}$ and $C_{ip}^{0}$ we used the lexicographic cost function of Section 2.6, and for that we assumed the costs $C_{ip}^{k}, C_{ip}^{u}$ and $C_{ip}^{0}$ are in descending order of importance.

Platform feasibility $F_{ip}$: Which platforms are feasible for each train depends on whether the line on which the train arrives or departs is connected to the platform, whether the platform is long enough for the train, whether the train is electric and whether the proposed platform is electrified, whether the train needs watering and whether the proposed platform has watering facilities, and so on. This data is rather lengthy to set out, especially since some of the data depends on the specific origin or destination of the train. Hence we omit the $F_{ip}$ data here and refer to Carey (1995).

Appendix B. Notation used in the paper

For reference, this appendix brings together the notation introduced in Sections 2 and 3 and used throughout the paper.

B.1. Subscripts

t, $t'$ denote trains. In the algorithm, $t$ is the current train, and $t'$ is the next train to be checked for conflicts with train $t$.

$p$ denotes a platform at a station. There may be any number of platforms, from one to dozens.

B.2. Sets or lists, for multi-train platforms

$M_{p}^{-}$: the set of sub-platforms which lie between sub-platform $p$ and the “−” end of the multi-
train platform.

$M_{p}^{+}$: the set of sub-platforms which lie between sub-platform $p$ and the “+” end of the multi-
train platform.

$M_{p}$: union of $M_{p}^{+}$ and $M_{p}^{-}$.
$L^-$: the set of lines in or out of the “–” end of the station.
$L^+$: the set of lines in or out of the “+” end of the station.

### B.3. Data

$I_t, O_t$: line on which train $t$ arrives (in-line) and departs (out-line) respectively.

$F_{tp} = \text{TRUE}$, if platform $p$ is feasible for train $t$, otherwise $\text{FALSE}$.

$A_{t\text{des}}, D_{t\text{des}}$ and $W_{t\text{des}}$: the earliest desired arrival and departure time and the minimum dwell time for train $t$. These times may sometimes depend on the platform used, in which case, add a $p$ subscript, thus $A_{tp\text{des}}$, etc.

$A_{t\text{U}}, D_{t\text{U}}$ and $W_{t\text{U}}$: upper bounds on the scheduled arrival, departure dwell times respectively for train $t$.

$X(l, p, l', p') = \text{TRUE}$ if the path connecting line $l$ to platform $p$ conflicts with the path connecting line $l'$ to platform $p'$, otherwise let $X(l, p, l', p') = \text{FALSE}$.

$H(t, p, t', p')_{\text{da}}$: the minimum headway required between train $t$ departing from platform $p$ and the next train $t'$ arriving at platform $p'$. Seven other headways, $H(t, p, t', p')_{\text{aa}}, H(t, p, t', p')_{\text{ad}}, H(t, p, t', p')_{\text{dd}}, H(t', p, t, p)_{\text{aa}}, H(t', p, t, p)_{\text{ad}}, H(t', p, t, p)_{\text{dd}}, H(t', p, t, p)_{\text{dd}}$, are defined similarly, where superscripts $d$ and $a$ denote departure and arrival, and the order of the superscripts (and subscripts) indicates the chronological order of the trains.

$f_t = 1$ if the arrival and departure times of train $t$ are fixed, otherwise 0. Some arrival and departure times may be fixed in advance and others are fixed in the course of the algorithm.

$C_a, C_d$ and $C_w$: the cost per minute of deviation of the arrival, departure and dwell times $A_{t\text{des}}, D_{t\text{des}}$ and $W_{t\text{des}}$ of train $t$ from the desired times $A_{tp\text{des}}, D_{tp\text{des}}$ and $W_{tp\text{des}}$.

$C_{ip}$: the platform undesirability cost incurred by train $t$ if assigned to platform $p$.

$C_{op}$: the platform occupation cost of assigning train $t$ to platform $p$.

$P_{pp'} = 1$ if $p$ and $p'$ are sub-platforms of the same multi-train, through platform, otherwise $P_{pp'} = 0$ (see Section 2.6).

### B.4. Variables

$A_t, D_t, W_t$: the scheduled arrival, departure and dwell times respectively of train $t$ at the end of the scheduling algorithm.

$p_t$: platform to which train $t$ is assigned.

$T_p$: set of trains assigned to platform $p$.

$T_p$ and $p_t$ includes any assignments fixed in advance and any confirmed assignments up to any stage in a platforming algorithm.

### B.5. Temporary variables

The following temporary variables are used in the course of the algorithm.
$A_{tp}^*, D_{tp}^*, W_{tp}^*$: the trial arrival, departure and dwell times for train $t$ at the current trial platform $p$. These trial times (for non-fixed trains) may be changed several times in the course of the algorithm. We could also refer to these times as draft, interim, or incumbent times.

$B_r = \text{TRUE}$ if any train $t'$ is present at platform $p$ for any part of the proposed time slot of train $t$ at platform $p$ (see Section 3.3.1).

$B_{r}^{a} = \text{TRUE}$ if train $t'$ is present at platform $p' \in M_p$ at arrival time of train $t$ at platform $p$ (see Section 3.3.2).

$B_{r}^{d} = \text{TRUE}$ if train $t'$ is present at platform $p' \in M_p$ at departure time of $t$ from platform $p$ (see Section 3.3.2).

$c_{kp}^k$, the knock-on costs incurred by train $t$ if it is assigned to $p$ (defined in Section 2.5), e.g.,

$$c_{kp}^k = c_{A}^a(A_{tp}^* - A_{des}^*) + c_{D}^d(D_{tp}^* - D_{des}^*) + c_{W}^w(W_{tp}^* - W_{des}^*).$$

References


Carey, M., Crawford, I., 2000. Scheduling trains through a series of busy complex stations. Research Report. Faculty of Business and Management, University of Ulster, N. Ireland, BT37 0QB.

Carey, M., Crawford, I., 2001. Improving train scheduling for lines with multiple busy stations. Research Report. Faculty of Business and Management, University of Ulster, N. Ireland, BT37 0QB.


