Available online at www.sciencedirect.com
ScienceDirect
Transportation Research Part B 41 (2007) 159–178
TRANSPORTATION RESEARCH PART B
www.elsevier.com/locate/trb

Scheduling trains on a network of busy complex stations

Malachy Carey [a,*], Ivan Crawford [b]

[a] School of Management and Economics, Queen's University, 25 University Square, Belfast, Northern Ireland BT7 1NN, United Kingdom
[b] Faculty of Informatics, University of Ulster, Northern Ireland, United Kingdom

Received 15 May 2005

Abstract

Many countries have busy rail networks with highly complex patterns of train services that require careful scheduling to fit these to the existing infrastructure, while avoiding conflicts between large numbers of trains moving at different speeds within and between multi-platform stations on conflicting lines, while satisfying other constraints and objectives. However, the construction and coordination of train schedules and plans for many rail networks is a rather slow process in which conflicts of proposed train times, lines and platforms are found and resolved 'by hand'. Even for a medium size rail network, this requires a large numbers of train schedulers or planners many months to complete, and makes it difficult or impossible to explore alternative schedules, plans, operating rules, objectives, etc. As a contribution towards more automated methods, we have developed heuristic algorithms to assist in the task of finding and resolving the conflicts in draft train schedules. We start from algorithms that schedule trains at a single train station, and extend these to handle a series of complex stations linked by multiple one-way lines in each direction, traversed by trains of differing types and speeds. To test the algorithms we applied them to scheduling trains for a busy system of 25 interconnected stations, with each station having up to 30 sub-platforms and several hundred train movements per day. We here report on the results from many hundreds of test runs. To make the tests more challenging, the algorithms start from initial draft timetables that we constructed so as to contain very large numbers of conflicts to be resolved. The algorithms, implemented in C code and run on a Pentium PC, found and resolved all conflicts very quickly. A further purpose of the algorithms is that they can be used to simulate and explore the effects of alternative draft timetable, operating policies, station layouts, and random delays or failures.

Keywords: Train scheduling; Timetabling; Network; Heuristics

1. Introduction

The train planning process is traditionally divided into several stages (e.g., see Watson, 2001) one of which consists of train operators drawing up draft train plans and timetables for running train services. These drafts typically start from current or past plans and timetables and adjust these in the light of estimates of travel

* Corresponding author. Tel.: +44 2890 245133; fax: +44 2890 975156.
E-mail address: m.carey@qub.ac.uk (M. Carey).

0191-2615/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trb.2006.02.002

demands between stations throughout the day, resources of rolling stock and crews, existing commitments, estimates of revenues and costs, business plans, etc. However, these draft train plans typically contain many conflicts or infeasibilities in train arrival and departure times, headways, station dwell times, platform usage and line usage. The next stage of the train planning process consists of taking the draft train plans (perhaps from a number of different rain operators using the same lines and stations), checking these to find possible conflicts and adjusting or revising them 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. This stage is the main concern of the present paper.

There is an additional way in which the train planning process may be sub-divided into components or stages. The rail network, for example in Britain, may be decomposed into a set of main corridors for which detailed scheduling can be undertaken largely separately. This is easier in a country such as Britain where the rail network is largely radial around major cities, and even the national train network is largely radial from London. In so far as possible, corridors are scheduled in decreasing order of importance and, when scheduling trains for a corridor, the times of the trains already scheduled for other corridors are taken as fixed. If there are substantial conflicts between the schedules for two corridors of similar importance then in practice there is often it ad hoc process of iterating or adjusting between these schedules, accompanied by negotiations with train operators. We assume this scenario in the present paper and are here concerned with scheduling for a single corridor. More specifically, we consider trains of various speeds and types travelling in both directions between a series of busy multi-platform train stations, with each station linked to neighbouring stations by one or more lines in each direction.

A major component of the train scheduling process in many countries is the scheduling of trains (assigning times and platforms) at individual busy stations. At such stations, train planners have to ensure that there are no conflicts between several hundred trains per day going in and out of the station on intersecting paths to lines and platforms, while ensuring that each train has sufficient allowance for headway, dwell time, travel time, etc. Headways are time gaps between trains and dwell time is the time that a train is stopped at a platform: both of these times are made up of various components explained later. All this has to be planned well in advance so that it can be published in advance in timetables for the travelling public. Computers are typically used, for example in Britain, to assist in the input, output and display of data and schedules, but at present generally do not perform the difficult tasks of finding or resolving train conflicts or making trade-offs between different objectives.

In earlier papers Carey and Carville (2000, 2003) developed and tested models and algorithms for scheduling trains for a single busy complex station. They used scheduling rules or heuristics analogous to those used in practice by expert train planners using 'manual' methods. Their heuristics have the advantage that they are based on those already found to perform well in practice, can easily be extended to handle additional information or objectives, make the scheduling choices more understandable and acceptable to train planners, and make it easier for the latter to adapt or revise the proposed solutions (schedules). These heuristics are also intended to make it easier to justify the scheduling choices to the competing train operators, and to the rail regulator who, in Britain, expects the rules to be understandable, consistent and fair. Here we focus on extending this work to a rail corridor. The above use of heuristic rules and algorithms is one of the features that distinguishes the present work from other work on train scheduling, but there are also other differences, as briefly outlined below. There is not a very large amount of research on train scheduling for busy timetabled trains, but see a survey by Cordeau et al. (1998).

Scheduling for a rail corridor has been investigated by earlier authors, including Petersen et al. (1986), Jovanovic (1989), Jovanovic and Harker (1991) and Kraay et al. (1991). Their concern is with a different scenario, namely the train meet-pass problem for a corridor of the type prevalent in North America. They therefore assume a two-way track used by trains in both directions, with trains passing at meet-pass points, which are double-line segments located at intervals along the track, say every dozen miles. They do not consider multi-platform stations along the line since these are almost non-existent in North America. In contrast, we are concerned with a rail system that is normal throughout Europe, hence we assume busy complex multi-platform stations with at least one one-way line in each direction between stations.

The concern of the present paper is also different from that in Carey (1994a,b) and Carey and Lockwood (1995). They too consider a rail corridor, but they assume that the stations along the corridor have unlimited

capacity so that finding time slots and platforms at these is not a problem. They adopt a mathematical programming branch-and-bound approach, with heuristics to guide the branching and bounding, to restrict the search domain and speed up the search. Here we adopt a heuristic approach without mathematical programming.

Yet another approach is taken by Zwaneveld et al. (1996) which considers the routing of trains through railway stations, given a proposed timetable. They determine whether there is a feasible routing to satisfy various constraints, formulating the problem as a node-packing problem. Kroon et al. (1997) consider complexity of variants of the above routing problem, formulating it as a fixed interval scheduling problem. The problem considered in both papers differs from that in the present paper. They consider the routing feasibility problem while we are also concerned with train planners objectives, in particular, the costs or penalties associated with scheduled delays and with choice of trains, platform or lines. Others have considered train services that adhere to a periodic pattern, e.g., Odijk (1996). However, the present paper, like Carey and Carville (2000, 2003), is concerned with rail systems where there may be little or no regular time pattern. Such regularity is often not possible, for example in Britain, due to different speed trains, very unequal distances between stations, trains skipping some stations, more trains needed at certain (peak) times, train time adjustments to avoid headway or dwell time conflicts, etc.

What we present in this paper is certainly not ready for use as a commercial system. For that, much more work would be needed, and the models and algorithms would need to be integrated with various interfaces and databases, including databases for the railway layout and infrastructure, minimum travel times between stations or junctions, minimum required headways and dwell times, etc. Here we are concerned only with developing and testing methods and algorithms for finding and resolving trains conflicts so as to satisfy various constraints and objectives.

1.1. A corridor with multiple lines and multiple busy complex stations

Consider a rail corridor consisting of a series of connected stations $1, 2, \ldots, N$, with each station s connected to the station $s + 1$ after it by one or more one-way lines in each direction. The one-way line from s to $s + 1$ is referred to as an out-line for station s and an in-line for station $s + 1$. For ease of reference, we can assume that each station $s + 1$ is due East of station s as in Fig. A.2 in Appendix A. Trains cannot overtake and pass each other on the one-way lines: they can overtake only at stations. Incidentally, the entire corridor described here is sometimes referred to as a "line", but this should not cause any confusion as it occurs only in phrases such as, the London to Brighton line, or the East Coast main line.

In the simplest case, trains travel from station 1 to station N or from N to 1, stopping at all stations on the way. However, the model also allows the following.

(i) Trains may originate or terminate at any station s, not just at stations 1 or N.
(ii) Trains may join or depart from the corridor at points between stations. (In that case the headways (time gaps) needed between trains when joining the corridor depend on the train speeds and on the distance to the next station.)
(iii) A train may pass through some stations without stopping. (These 'through trains' are treated in the same way as stopping trains, though the minimum required headways may be different.)

1.2. A single-station model and Algorithm A1

As already noted, Carey and Carville (2000, 2003) developed and tested an Algorithm A1 for scheduling trains at a single station. In the present paper we start from that algorithm and extend it to deal with a train corridor containing multiple stations and lines as above. For a single station the Carey and Carville algorithm operates as follows. First make a list of feasible platforms for each train t. (A platform is feasible if it has the appropriate electrification, watering, length, ownership, or other facilities for that train, and is connected to the in-line and out-line to be used by train t.) To find the best platform for the 'current' train t, make a trial assignment of train t to each feasible platform p in turn. Making a trial assignment to platform p involves

finding, and resolving, all headway conflicts between t and all other already scheduled trains t', that would be incurred if we assigned train t to platform p. For each conflict that is found, resolve it by delaying ('knocking-on') the arrival and/or dwell or departure times of one of the conflicting trains, delaying it just enough to satisfy the headway requirements that are prespecified by the train operator for those train types, platforms, etc. Note that having found and resolved all the conflicts that would be incurred in assigning train t to a platform p, the algorithm calculates the total cost of these conflict resolutions at platform p, then *cancels* all of these conflict resolutions before moving on to try the next platform, and the next, and so on. In the end, the algorithm retains the conflict resolutions only for the platform p that is eventually chosen as the best (least cost) platform for train t.

When making a 'trial' assignment of train t to a platform p, the Algorithm A1 checks for several types of conflict and resolves each of these. For example, it checks for headway conflicts between each pair of arriving trains and, if necessary, it adjusts the arrival time of one of them, to ensure a minimum headway appropriate for the train types, line, platform and train order. Similarly, it checks for, and resolves, conflicts between pairs of departing trains, and between arriving trains and preceding departing trains, and between departing trains and preceding arriving trains. It also checks whether the proposed platform is currently occupied. All of these checks are complicated by the existence of sub-platforms. Main platforms may be divided into a series of sub-platforms each of which can hold a train. Trains going to or from a sub-platform may have to pass through other sub-platforms, and can do so only if these are currently empty.

Following the above conflict checks and resolutions, the algorithm calculates which platform p would yield the lowest delay or costs for train t, assigns train t to this platform (i.e., records the revised/updated trial values of arrival, dwell and departure times for train t at platform p), and adds train t to the list of scheduled trains.

There are two further features that we should note here concerning the above algorithm and the rest of this paper. First, when we refer to train delays, unless stated otherwise, we mean scheduled delay, that is, waiting times included in the timetable in order to construct a feasible timetable, to ensure required headways are satisfied, or to delay a train in favour of a train with higher benefit or priority. We do not mean the random train delays that are incurred during daily train operations. Second, the trip times (running times) between scheduled stops (usually stations) are the trip times that are used by train operators in constructing train timetables. They can contain a percentage time allowance (running time supplements) to allow for random train delays that occur in daily operations. In some countries, train operators also insert up to a few minutes running time supplements at certain points in long train routes. These can easily be treated as part of the trip times when constructing the timetables.

2. A multi-station multi-line model and algorithms

Here we extend the single-station scheduling Algorithm A1 from Carey and Carville (2003) so as to find and resolve conflicts on lines between stations. We assume that there is one or more one-way lines, in each direction, between stations. The methods used depend on the number of lines, and on whether the choice of line for each train is prespecified or is chosen within the algorithm, for example, to minimize delays or costs. We initially assume a single one-way line, in each direction, between each pair of stations, i.e., from s to $s+1$. We introduce multiple lines in Sections 2.1 and 2.2. We assume throughout that the various trains t have different speeds profiles and hence take different amounts of time to travel between stations, and that these inter-station travel times are known.

If a train departs from a station shortly after a slower train on the same one-way line it may catch up with the latter before the next station or scheduled stopping point, hence become delayed behind it. To avoid such a line conflict we need to enforce a time gap or headway between the train departure times. For example, if the train ahead takes x minutes more than the current train to travel from station s to $s+1$ then an extra headway of x minutes is needed before letting the current train depart. To ensure this let

$$T_{ts,s+1} = \text{the fixed trip time of train } t \text{ from station } s \text{ to } s+1.$$

Then, to avoid train speed conflicts, the headway required between t and an *earlier* train t' is

$$H^{(1)}_{t'ts,s+1} = \max\{0,(T_{t's,s+1} - T_{ts,s+1})\}.$$

If train t is slower than the preceding train t', then $(T_{t's,s+1} - T_{ts,s+1})$ is negative and the above minimum headway reduces to 0. If train t or t' does not travel all the way from s to $s+1$, then the above headway still applies, but the trip times $T_{t's,s+1}$ used in defining the headways $H^{(1)}_{t'ts,s+1}$ are then redefined as the trip times from station s up to the point at which trains t and t' stop sharing the same line.

The above headway $H^{(1)}_{t'ts,s+1}$ only ensures that a train does not catch up with the train ahead on the line. An additional minimum safety headway is needed to keep the trains apart: let this be $H^{(2)}_{t'ts,s+1}$. Also, an additional headway may be needed for marketing or other policy reasons, e.g., to have intercity trains at least 10 min apart: let this be $H^{(3)}_{t'ts,s+1}$. Combining these headways gives

$$H_{t'ts,s+1} = H^{(1)}_{t'ts,s+1} + \max\{H^{(2)}_{t'ts,s+1}, H^{(3)}_{t'ts,s+1}\}. \tag{1}$$

To enforce the above headways on the line from s to $s+1$ we adapt the algorithm that is used in Section 3.2 of Carey and Carville (2003) when considering a single station. Their algorithm included checking the headways between train t and all other trains departing from all platforms in the time window around train t, and adjusting departure time of train t if a headway conflict is found. Since they were concerned with scheduling trains only at a single station, their train "departure headways" did not explicitly include the inter-station line headways $H_{t'ts,s+1}$ which we introduced above. However, their scheduling algorithm will still continue to work if we include additional components (such as $H_{t'ts,s+1}$ above) in the headways required between trains departing from a station. That is because the minimum/required headways are simply data items in the algorithm and the algorithm does not depend on the value of the data items. In Carey and Carville, the minimum departure headways needed in the immediate exit from the station (to avoid conflicts between trains on lines that are crossing or merging in the exit area) is:

> $H(O_{t'}, p', O_t, p)^{dd}$, the headway required between two trains t and t' departing on conflicting (crossing) paths out of a station, where O_t and $O_{t'}$ are the out-lines used by these trains and p and p' are the platforms from which they depart.
> [The "*dd*" superscript is used to indicate headway between two departing trains, to distinguish it from headways required between arriving trains, or between arriving and departing trains.]

To combine the required intra-station headway $H(O_{t'},p',O_t,p)^{dd}$ and required inter-station headway $H_{t'ts,s+1}$ write

$$H(O_{t'},p',O_t,p,s,s+1)^{dd} = \max\{H(O_{t'},p',O_t,p)^{dd}, H_{t'ts,s+1}\}. \tag{2}$$

Note that we do not sum these two headway requirements. To enforce both types of headways simultaneously, replace the headway $H(O_{t'},p',O_t,p)^{dd}$ in Carey and Carville with the above combined headway.

The numerical values that we use here for the above headways, and components of headways, are the values that were used by Railtrack and later Network Rail, and published in various of their internal documents. They are also set out in an appendix of Carey and Carville (2003). The values are in minutes but the models and algorithms in this paper are compatible with using seconds or fractions of minutes.

2.1. Scheduling the current train at the current station: a simple conflict resolution method

In this section we set out how to find the 'best' time slot (departure time and dwell time) and platform for train t at station s, given that train t will exit from station s onto a line to $s+1$: we assume that the 'best' time slot and platform is that which incurs the least immediate cost or delay. In the method set out below we refer to train t station s, but to handle all the trains this will of course have to be repeated for all trains t and stations s, in a sequence order as discussed in Carey and Carville (2003).

If train t exits from station s just before, or after, another train, then the time intervals between their departure times must be sufficient to ensure the various types of headway requirements (set out in Section 1) are satisfied. In particular, as noted in Section 1, if train t takes x minutes longer than the next train t' to travel from s to $s+1$ then an additional x minutes is needed in the headway between the trains, to ensure that they do not conflict. All of the various required headway components are included in the headway term $H(O_{t'},p',O_t,p,s,s+1)^{dd}$.

In Sections 3.1–3.2 of Carey and Carville (2003), if a conflict is found between the trial departure time of the current train t (at its current trial platform) and another already scheduled train t' they resolve the conflict by delaying (knocking-on) the (trial) departure time of the current train t, not the other train t'. For expositional simplicity we follow a similar procedure in this section. However, this method may produce poor solutions if trains have significantly different speeds between stations. In Section 3.1 we will expand to a method that produces better solutions when there are differences in the train speeds: there we instead adjust the times of whichever train (t or t') causes least immediate delay or cost. From Carey and Carville (2003):

> A^*_{tp} and D^*_{tp} are the current (trial) arrival and departure times, respectively, of train t from its current trial platform p. The corresponding dwell time at the platform is then $W^*_{tp} = D^*_{tp} - A^*_{tp}$.
> $A_{t'}$ and $D_{t'}$ are the current (trial) arrival and departure times, respectively, of any other train $t' \neq t$ currently being considered for possible conflicts with train t. The corresponding dwell time of train t' is then $W_t = D_t - A_t$.

All of these times may be times from the initial draft timetable or may have been already adjusted in an earlier iteration of the algorithm for finding and resolving time conflicts.

The check for conflicts between the current train t and each other train t' depends on whether t departs before or after t', hence we have two cases (a) and (b) as follows.

(a) *Finding and resolving conflicts with preceding trains*

Check if, given their current trial departure times at station s, train t currently departs *after* train t' on the same line from station s to $s+1$. If so, check if the headway requirement between trains t' and t is violated, that is, check if the departure time of train t (i.e., D^*_{tp}) lies in the headway interval that is required after the departure time $D_{t'}$ of train t'. More formally check

$$\text{if } (D_{t'} \leqslant D^*_{tp} \leqslant D_{t'} + H(O_{t'},p',O_t,p,s,s+1)^{dd}). \tag{3}$$

If this is true, there is a headway time conflict between t' and t, which can be resolved by moving the trial departure time D^*_{tp} of train t back to the end of the headway interval after train t'. Delaying the departure time of train t of course also increases the dwell time W^*_{tp} of train t. Thus, if conflict (3) occurs, then change (increase) the trial values D^*_{tp} and W^*_{tp} as follows.

$$\text{Set } (D^*_{tp} = D_{t'} + H(O_{t'},p',O_t,p,s,s+1)^{dd}) \quad \text{and} \quad (W^*_{tp} = D^*_{tp} - A^*_{tp}). \tag{4}$$

The cost of making these changes can be written as

$$c^l_{tps,s+1} = c^l_{tps,s+1}(D^{*\Delta}_{tp}, W^{*\Delta}_{tp}). \tag{5}$$

where $D^{*\Delta}_{tp}$ and $W^{*\Delta}_{tp}$ denote the changes from the previous values of D^*_{tp} and W^*_{tp}, respectively. We have inserted an l superscript on c since we will later introduce choice of lines l and the cost can depend on the choice of line. The costs Eq. (5) is likely to be linear in the variables. For example, the cost per minute of departure delay ($D^{*\Delta}_{tp}$) may be taken as proportional to an estimate of the number of passengers most affected, that is, the passengers who join the train at station s. It may also depend on the type of train (e.g., intercity or local commuter) and on the platform. Similarly, the cost per minute of extra dwell time ($W^{*\Delta}_{tp}$) may be taken as proportional to an estimate of the number of passengers who stay on the train at station s (who do not get on or off the train).

$$\text{Add } c^l_{tps,s+1} \text{ to the cumulative cost of assigning train } t \text{ to trial platform } p. \tag{6}$$

We refer to the "cumulative cost" of assigning train t to trial platform p since, each time that we resolve a conflict by moving the times of train t, we add a corresponding amount $c^l_{tps,s+1}$ to the cost.

(b) *Finding and resolving conflicts with succeeding trains*

This is similar to (a) above, but here we check if train t currently departs *before* train t' on the same line from station s to $s+1$. If so, check if the headway requirement between trains t' and t is violated, that is, check

if the departure time of train t' (i.e., $D_{t'}$) lies in the headway interval that is required after the departure time D^*_{tp} of train t. More formally check

$$\text{if } D^*_{tp} \leqslant D_{t'} \leqslant D^*_{tp} + H(O_t,p,O_{t'},p',s,s+1)^{dd}. \tag{7}$$

If this is true, there is a headway time conflict between t' and t, which can be resolved by moving the trial departure time D^*_{tp} of train t to after that of train t', exactly as in (4)–(6). The above discussion can be summarised in the following algorithm.

Algorithm M0. To implement the above conflict checks and resolutions, to schedule train t at station s, amend Algorithm A1 of Carey and Carville (2003) as follows. Replace the departure headway checks and resolutions in A1 with (a) and (b) above. This only requires changing their $H(O_{t'},p',O_t,p)^{dd}$ to $H(O_{t'},p',O_t,p,s,s+1)^{dd}$ and adding the new cost $c^l_{tps,s+1}$ as in (5) and (6). The six other types of arrival and departure headway conflict checks and resolutions in their Algorithm A1 (Section 3.2) remain unchanged.

2.2. Introducing multiple lines, with line choice prespecified

Only trains on the same line between s and $s+1$ can have line headway conflicts hence, if there is more than one line, then when checking for line conflicts in (3) above we must check if the trains use the same line. Let

$l_{ts,s+1}$ denote the line used by train t when departing from station s 'towards' $s+1$ (train t may or may not arrive at $s+1$: it may leave the network before station $s+1$). Normally $l_{ts,s+1}$ is the out-line O_t for train t at the station.

Trains t and t' use the same line if and only if $l_{ts,s+1} = l_{t's,s+1}$, hence we insert this in the line conflict check (3), so that (3) becomes

$$\text{if } l_{ts,s+1} = l_{t's,s+1} \quad \text{and} \quad D_{t'} \leqslant D^*_{tp} \leqslant D_{t'} + H(O_{t'},p',O_t,p,s,s+1)^{dd}. \tag{3'}$$

Similarly for conflict check (7). The rest of the conflict checking and resolving procedure is otherwise unchanged from the single-line case above.

2.3. Introducing multiple lines, with choice of line

Suppose there are two or more one-way lines $l = 1,2,\ldots,$ from station s to $s+1$. The train headways and line conflicts depend on the choice of line for each train. We have to choose a line l for each train, and we assume the line is chosen so as to reduce delays or costs. A key consideration in assigning trains to lines is: if the trains on a line have the same speed, then many more trains can use the line than if they have very different speeds. For example, if a slow train is followed by a fast train, then the latter must slow down or allow a long time gap before it departs, since trains cannot overtake and pass on a single line. Because of this, there are often two or more lines in the same direction between stations and train operators often designate one of these as the 'fast' line and one as the 'slow' line.

In practice various different rules are used by train planners when choosing lines for trains. We introduce two general alternate rules that are analogous to those used in practice, and are outlined as follows. (Rule 1 would be executed prior to starting the timetabling algorithm, whereas Rule 2 would be implemented during the algorithm.)

Rule 1. Suppose there are n trains travelling from station s and $s+1$ on m lines. List the trains in descending order of speeds and divide the list into m groups, having successively lower speeds. Assign the first group (fastest trains) to line 1, the second group to line 2, and so on, so that the line for each train is thus prespecified.

This reduces the problem to that in the previous section. How the trains are divided into groups depends on the spread of train speeds. For example, if the train speeds are 125, 110 and 90 miles per hour (mph) and there are only two lines, we may assign the 125 mph trains to line 1, the 90 mph trains to line 2, and the 110 mph

trains to whichever line has fewer trains. Or, if there are a relatively large number of 110 mph trains, we may assign some of them to each line. If there are many different train speeds, we could again rank the trains in descending order of speed, and divide the n trains into m different speed groups, with approximately n/m in each group. To improve the homogeneity of train speeds on each line, we may then move some trains between lines. For example, if a few trains on one line have a much higher speed than the others on the line, and are closer to speeds on the next line, then we may move them to the next line.

Rule 2. First try to schedule all of the highest speed trains at station s, then the next highest speed trains, and so on. When assigning each train t, first try to assign it to line 1, then line 2, and so on, skipping any lines which are not allowed for train t, until we find the line which causes the lowest immediate delay (or delay cost) to train t. If the delays or costs for some line are zero, we need not consider any further lines.

By "try to assign train t to line s" we mean, find the best platform p for train t, as set out in Section 2.1 above. In practice train planners seldom try more than one or two lines for each train t. First, there are usually only two lines, sometimes three and seldom more. Even if there are more, the first or second line may yield no line conflicts for train t, hence no more need to be considered. Further, some lines may be ruled out, being reserved for trains of a certain speed, type or destination. For example, line 1 may be reserved for the highest speed trains, line 2 may not allow trains above a certain speed, and line 3 may be required for trains which divert *en route* to the next station. To represent this we can let $l_{ts,s+1} \in F_{ts,s+1}$ where $F_{ts,s+1}$ denotes the set of lines which are permissible for train t departing from station s to $s+1$. A special case is when the lines for all trains are prespecified, as in Section 2.2. Even when more than one line is feasible for train t, some lines may be preferred to others. To take account of that let

C_{tl} be a cost or penalty for train t using line l.

If line l is used by train t we add the cost C_{tl} to the other costs to be considered when choosing a line, platform and time slot for train t.

3. More flexible options for resolving conflicts

At each station or stop, Algorithm M0, in Section 2, resolves time conflicts between train t and any other already scheduled trains t' by adjustment (delaying) the times of the former (train t) by the smallest amount that is necessary to avoid conflicts (and, in doing this, the algorithm tries every platform that is permitted for train t, to find the platform that requires the least time adjustments (delays) to resolve any time conflicts). Actually, in the Algorithm M0 (and the other algorithms below) the criterion used is not just the "smallest time delay" but is the "least cost time delay", since costs other than time can be included, and times at different platforms can be weighted differently. While Algorithm M0 considers adjusting only the times (and perhaps the platform) of train t, in contrast, Algorithm M1, which is introduced below, instead resolves conflicts by adjusting the times (and perhaps the platform) of the current train t or the conflicting train t' depending on which would cause the least immediate net delay or cost to resolve the conflict.

3.1. Conflicts at exits from stations or junctions and on lines between these

In Section 2.1 above we set out a method for finding a suitable platform (and time slot) for train t at station s, taking account of the headways which are required between t and other trains t' on the line from station s to $s+1$. We assumed that, in resolving any conflicts, only the times of the current train t can be adjusted, not the times of the other trains t'. This restriction may be acceptable when only small headways are involved, for example, headways $H(O_{t'},p',O_t,p)^{dd}$ needed to avoid conflicts only in the station exit. However, it can cause very large delays if applied to resolving conflicts due to differing speeds of trains on the same line. For example, instead of letting a fast train t depart just after a slow train t', net delays or costs may be greatly reduced by letting the fast train t' go first. We therefore relax the earlier assumption, and instead adjust the times of whichever train (t or t') causes least immediate delay or cost. This implies that, when considering each trial platform p for train t we also consider/try changing the departure time and dwell time for each conflicting train t', hence we will have to record these proposed changed times. To record these, let:

$D_{t'p}$ = current trial departure time of train t' when considering assigning train t to platform p.
$W_{t'p}$ = current trial dwell time of train t' when considering assigning train t to platform p.

Note that in $D_{t'p}$ and $W_{t'p}$ the subscript is p not p', the platform used by train t' and, unless by coincidence, $p \neq p'$. We use p rather than p' since, as we will see, the trial departure time of train t' can depend on which platform we are considering for train t.

Then to check and resolve conflicts on lines between stations we can proceed as follows.

(a) *Finding and resolving conflicts with preceding trains*

This is the same conflict check as (3) in (a) in Section 2.1, except that $D_{t'}$ in (3) is here replaced by $D_{t'p}$. That is, check if train t currently departs *after* train t' on the same line from station s to $s+1$ and, if so, check if the headway requirement between trains t' and t is violated. Thus, following (3), check

$$\text{if } D_{t'p} \leqslant D^*_{tp} \leqslant D_{t'p} + H(O_{t'},p',O_t,p,s,s+1)^{dd}. \tag{8}$$

If this is true, there is a headway time conflict between t' and t, which can be resolved (as in (4)–(6) above) by moving the trial departure time D^*_{tp} of train t back to the end of the headway interval after train t', keeping their sequence order fixed, that is,

$$\text{set } (D^*_{tp} = D_{t'p} + H(O_{t'},p',O_t,p,s,s+1)^{dd}) \quad \text{and} \quad (W^*_{tp} = D^*_{tp} - A^*_{tp}). \tag{9}$$

This is the same as (4) in Section 2.1, except that $D_{t'}$ in (4) is here replaced by $D_{t'p}$. The cost of this change is, as in (5),

$$c^l_{tps,s+1} = c^l_{tps,s+1}(D^{*\Delta}_{tp}, W^{*\Delta}_{tp}). \tag{10}$$

Alternatively, the conflict (8) can be resolved by moving the departure time $D_{t'p}$ of train t' back in time to after the departure time of train t, which changes the train order. To do this, in (9) and (10) swap D^*_{tp} with $D_{t'p}$, etc., so that these equations become,

$$\text{set } (D_{t'p} = D^*_{tp} + H(O_t,p,O_{t'},p',s,s+1)^{dd}) \quad \text{and} \quad (W_{t'p} = D_{t'p} - A_{t'p}) \tag{11}$$

$$\text{and } c^l_{t'ps,s+1} = c^l_{t'p's,s+1}(D^{\Delta}_{t'p}, W^{\Delta}_{t'p}). \tag{12}$$

The decision rule for resolving the line headway conflict is then:

$$\begin{aligned}&\text{If } c^l_{tps,s+1} < c^l_{t'ps,s+1} \text{ then (provisionally)adjust the times of train } t,\\ &\text{i.e., execute (9), otherwise (provisionally)adjust the times of train } t',\\ &\text{i.e., execute (11).}\end{aligned} \tag{13}$$

$$\text{Add the smaller of } c^l_{tps,s+1} \text{ and } c^l_{t'ps,s+1} \text{ to the cumulative cost of assigning train } t \text{ to trial platform } p. \tag{14}$$

The train time-changes ('knock-ons') in this decision rule are provisional until it is eventually decided to which platform p to assign train t. [The eventual decision as to which platform to choose is made when all options (platforms) have been tried and all the conflict adjustment costs for each options have been computed, at which point the lowest cost option (platform) is chosen.]

(b) *Finding and resolving conflicts with succeeding trains*

This is the same conflict check as (7) in (b) in Section 2.1, except that $D_{t'}$ in (7) is here replaced by $D_{t'p}$. That is, check if train t currently departs *before* train t' on the same line from station s to $s+1$ and, if so, check if the headway requirement between trains t' and t is violated. Thus, following (7), check

$$\text{if } D^*_{tp} \leqslant D_{t'p} \leqslant D^*_{tp} + H(O_t,p,O_{t'},p',s,s+1)^{dd}. \tag{15}$$

If this is true, there is a headway conflict between trains t' and t, and as in (a) above, we can resolve this conflict by moving the departure time D^*_{tp} of train t back in time, until after the departure time of train t', which again yields exactly Eqs. (9) and (10). Alternatively, we can resolve the conflict by moving the departure time $D_{t'p}$ of train t' back in time, until after the departure time of train t, which again yields exactly Eqs. (11) and (12). Eq. (14) remains unchanged.

3.1.1. Rescheduling trains whose times have been changed when platforming the current train

Suppose that in the course of finding feasible times for train t at platform p (the platform eventually chosen for train t on the basis of least delays or costs), we delayed (knocked-on) the times of some other already scheduled trains t' at this platform p. Having scheduled train t to platform p, put these other trains t' back into the pool of trains that are still to be scheduled. The reason for doing this is that changing their times may have put them in conflict with yet other trains, hence their times have to be completely rechecked for time conflicts. When we put these trains t' back into the pool of trains still to be scheduled, we can use either of two heuristic rules, namely:

(i) Reset the draft arrival, dwell and departure times of the trains t' to the times that existed before checking for conflicts against the current train t, or,
(ii) leave their draft arrival, dwell and departure times at the adjusted (knocked-on) times computed in (a) and (b) above.

Intuitively, rule (i) seems preferable, and in computational examples it produced much better solutions than rule (ii), though it took slightly longer to run. A reason why (i) performs better than (ii) seems to be that (ii) "locks in" a time delay that may not be relevant when we reschedule train t'. For example, train t' may be rescheduled to a different platform where it will not suffer any delay. The above discussion can be summarised in the following algorithm.

Algorithm M1. To implement the above conflict checks and conflict resolutions, to schedule train t at station s, amend Algorithm A1 of Carey and Carville (2003) as follows. For train departures, replace the headway checks and resolutions in A1 with (a) and (b) above. This may cause the times of some already scheduled trains t' to be changed (delayed) when scheduling train t. If that occurs, then put these trains back into the list of non-scheduled trains, as in (i) or (ii) above, so that they can be re-checked for conflicts and re-scheduled if conflicts are found.

3.2. Conflicting dwell times at the same platform

As noted above, when finding the 'best' platform and time slot for each train t at a station, we search for all possible conflicts between train t and all other trains t'. To resolve such conflicts, in Algorithm M0, only the times of the current train t are adjusted, not the times of the conflicting trains t'. In Algorithm M1 the times of the current train t or the conflicting trains t' may be adjusted, depending on which of these results in the smaller immediate net time-adjustments (cost). However, even in Algorithm M1, this choice (of t or t') is applied only to resolving headway conflicts between trains using the same line, and headway conflicts between trains while departing from the same station. But there is also another type of time conflict check in Algorithms M0 and M1, namely between trains wishing to dwell at the same platform within a station. This dwell time conflict can occur even if there are no other headway conflicts. In Algorithms M0 and M1 above, such dwell time conflicts are resolved by adjusting only the times of the current train t.

We now (for Algorithm M1* below) consider resolving dwell time conflicts by adjusting the times of the current train t or the conflicting train t', depending on which of these results in the smaller immediate net time-adjustments (cost). To detect a dwell time conflict we need to check if the dwell time of train t overlaps with that of train t' at a platform, which requires checking three cases, namely: does train t arrive at the platform

(a) before t' arrives and depart after t' arrives; or
(b) before t' departs and depart after t' departs; or
(c) after t' arrives and depart before t' departs.

The further alternative (does t arrive before t' arrives and depart after t' departs) is already included in the first two cases above. Also, in each conflict check we include an allowance for the appropriate minimum headway for that particular pair of trains at that particular station.

Now suppose we are considering assigning train t to platform p. Check, as above, whether any other train t' has already been assigned to the same platform for a time duration that would overlap with the proposed time slot for train t. If there is an overlap, then to resolve this conflict we can adjust the times of trains t or t'. In Algorithms M0 and M1 we assume that this dwell time conflict is resolved by a minimal moving back (delaying) of the time(s) of train t. Here we propose to resolve it by a minimal moving back of the times of train t or t', depending on which causes the smaller net immediate delay. In computing the delay involved in changing the times of either train we of course include the appropriate minimum headway between the trains. We can set out an amended version of Algorithm M1 as follows.

Algorithm M1*. Same as Algorithm M1 except that, when resolving conflicts between trains being considered for the same platform p we delay, by the minimum necessary amounts, the times at platform p of the current train t or the conflicting train t' depending on which of these adjustments is the smaller or has the smaller immediate costs.

4. Computational testing and examples

To test and illustrate the above model and algorithms, we programmed them in the 'C' language and applied them to numerical examples based on test networks. The basic test network and data are discussed in Section 4.1 below and set out in additional detail in Appendix A. Other test networks were constructed as variants on this. The results obtained from scheduling trains for this network are set out in Section 4.2 below.

4.1. Generating a test network and an initial draft schedule containing large numbers of conflicts

We constructed a test network consisting of 25-stations ($i = 1, \ldots, N$) on a West to East corridor, linked by one-way lines in each direction. Each station has 11 main platforms, sub-divided into a total of 33 sub-platforms. Some of these are 'through'-platforms and some are dead-end or terminal platforms. 464 trains per day use each station. Of these 464, there are 112 'through' trains in each direction: 112 originate at station 1 terminate at station N after travelling through all stations in between, and 112 do the reverse, starting at station N and terminating at station 1. The remaining 240 trains at each station i use only that station. Of these, 190 are 'turnaround' trains, that depart from the station in the same direction that they arrived, 25 originate at the station, and 25 terminate at the station.

The train scheduling algorithms in this paper all basically operate by starting from a given ''draft'' timetable (that is, draft arrival and departure times that may contain large numbers of conflicts, and no draft platform allocation). The algorithms adjust these draft times, and allocate platforms at stations, in order to produce a ''good'' feasible schedule. We deliberately constructed the draft timetable so that it contains a large number of train time conflicts, to provide a more challenging test of our algorithms for resolving these conflicts. We based the draft timetable for the initial station on an old British Rail draft timetable for this station, and from this constructed draft timetables for the other stations as follows. We assume the draft times of the non-through trains are the same at each station, and are the same as in the British Rail draft. For 'through' (intercity) trains going West–East let T_t be the minimum trip time between stations for train t, so that the draft arrival time at station $s + 1$ is obtained by adding T_t to the draft departure time from station s. Add a minimum dwell time to this to obtain a draft departure time from station $s + 1$. For East–West intercity trains, repeat the above process in reverse.

The above draft timetable has substantial numbers of conflicts for several reasons. First, about one quarter of the trains are West–East trains and one quarter are East–West trains. Even if the draft times of the trains at the initial station are reasonably free of conflicts, shifting about a quarter of the trains forward in time by an arbitrary amount (T_t plus dwell times, as described in the preceding paragraph), and shifting about quarter

backwards in time by a similar amount, scrambles the draft timetable at each station, causing large numbers of conflicts. Second, the data for the initial station is based on data from a real train station and for that station most of the through trains diverge onto separate lines relatively soon after the station. In contrast, we here force these trains to all stay on the same lines through a series of stations. Third, when scheduling trains for a corridor, the set of train types and speeds and the departure times from the initial station are usually selected or adjusted so as to reduce the chances of conflicts at later stations. We did not do that here. Fourth, in the British Rail draft timetable on which we based the draft timetable for the initial station, the dwell times for many of the trains are not the same as the ''minimum desired dwell times'' for each train type. In our draft timetable we changed the draft dwell times to the latter (i.e., to the minimum desired dwell times), which caused train time conflicts. Finally, to ensure a large number of conflicts for the algorithms to resolve, we assume that there is only one line in each direction between stations, so that all trains in the same direction between stations have to use the same line. Also, to reflect what is the normal situation in Britain and Europe, we assume there is nowhere along this line for trains (travelling in the same direction) to overtake and pass each other, hence that if a fast train is just behind a slow train it will have to stay behind it until the next station. We refer to such conflicts as ''line speed'' conflicts, or line trip-time conflicts.

In the examples below we also assume that:

(i) The draft trip times are the minimum trip times that will be allowed in the final schedule. This follows the standard procedure in Britain. There the minimum trip times between stations are based on the maximum safe speeds and accelerations, which are calculated taking account of the type of train and line, and the line gradients, curvatures, etc. all along the line. Hence, in the train timetable, trains cannot be speeded up to avoid conflicts.

(ii) The draft trip times between neighbouring stations is the same for all trains within any given class of trains. In initial experiments we assumed the draft minimum trip time was the same for all trains, but in most runs we assumed different minimum trip times for each class of train.

(iii) If a through train would depart from a station after 11.00 pm it instead spends the night there, or in a depot, and its draft arrival time at the next station is $(8 + x)$ hours after its arrival at the current station, where x is the trip time between stations.

Also, in the scheduling algorithm:

(iv) We first scheduled all trains at station 1, then all at station 2, and so on, which is a common procedure in existing ''manual'' methods for train scheduling for a rail corridor.

(v) When choosing the next train to schedule at a station, we initially chose the next train in chronological order of the draft arrival times. However, we also experimented with other orders, for example, sorting into order of decreasing business priority (e.g., through trains, turnaround trains, etc.) and into chronological order within this business order.

4.2. Comparing the three Algorithms M0, M1 and M1*

In the previous sections we set out three algorithms for scheduling and platforming trains through a series of busy complex stations. The first (M0) is the natural generalisation of Algorithm A1 from Carey and Carville (2003). In this, when there is a train time conflict that is not resolved by changing train platforms, it is always the time of the current train that is adjusted. In contrast, in Algorithm M1 time conflicts may be resolved by adjusting the time of the current train or of the other conflicting train, depending on which yields the smaller immediate (local) net delay. In Algorithm M1 this choice of which train times to adjust is confined to resolving conflicts between trains on lines between stations (line conflicts) and headway conflicts between trains departing from one station to the next (station exit conflicts). In Algorithm M1* we extend this idea (choice of which train to adjust) to resolving time conflicts between trains that use the same platform.

In this section we compare the three algorithms by applying them to the same scheduling problems. We apply them to scheduling and platforming trains over 25 busy complex stations using the data and network set out above and in Appendix A.

The results are illustrated in Figs. 1–7. These show that in general Algorithm M1 provides better solutions than Algorithms M0 and M1* provides better solutions than Algorithm M1. Figs. 1 and 2 show the arrival delays

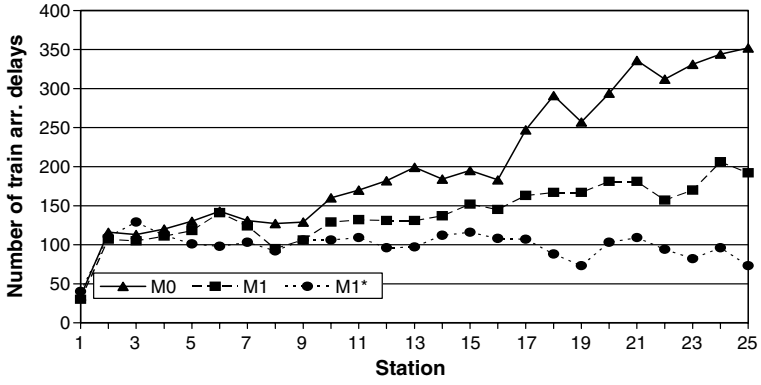

Fig. 1. Number of arrival delays with Algorithms M0, M1 and M1*.

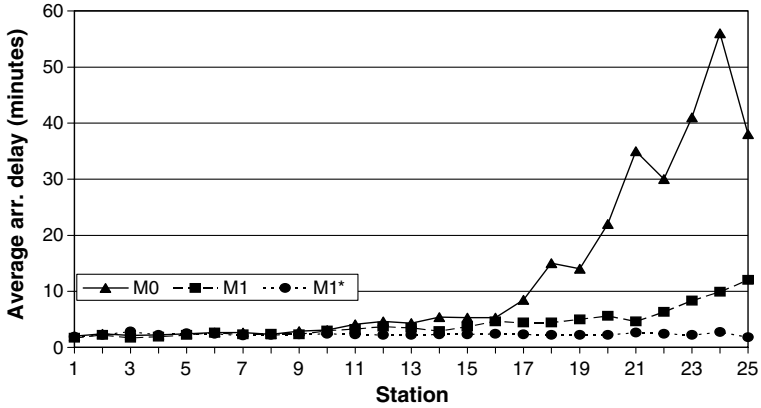

Fig. 2. Average size of arrival delays with Algorithms M0, M1 and M1*.

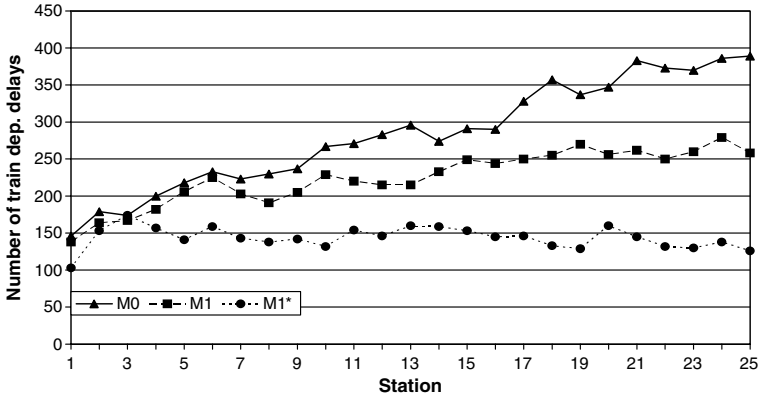

Fig. 3. Numbers of departure delays with Algorithms M0, M1 and M1*.

at each station, and Figs. 3 and 4 show the departure delays at each station. In Algorithm M0 and to a lesser extent in M1, the arrival delays increase with each successive station, as some delays at each station 'knock-on' to the next station. In Algorithm M1* this knock-on effect between successive stations is largely eliminated.

In Figs 1–4 we arbitrarily assumed that the minimum permitted trip times between stations are 10 min for through trains (intercity), 15 min for local (turnaround) trains, and 20 min for parcel and freight trains. To see

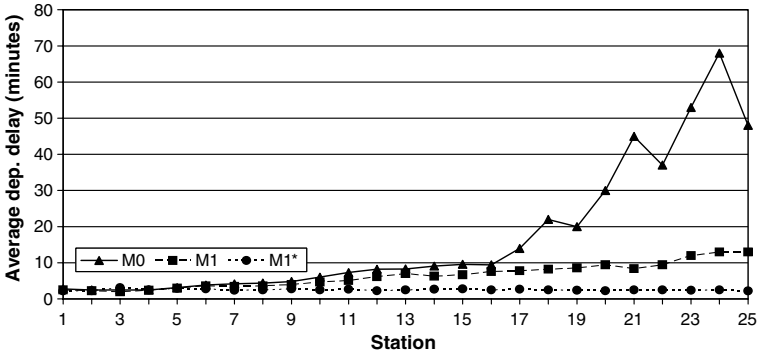

Fig. 4. Average size of departure delays with Algorithms M0, M1 and M1*.

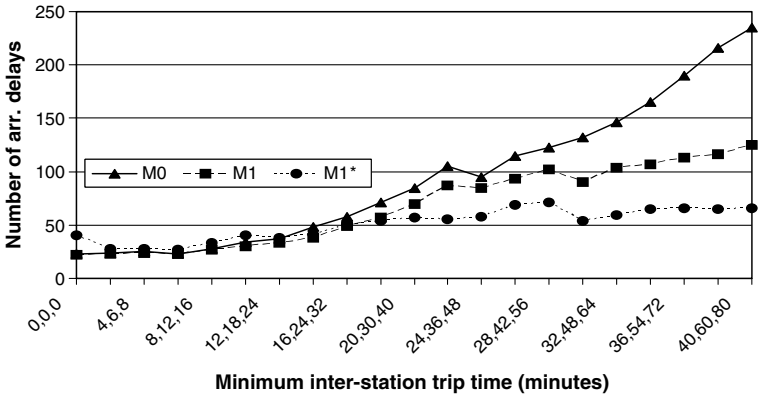

Fig. 5. Numbers of *arrival* delays, with Algorithms M0, M1 and M1* and increasing minimum trip-times between stations.

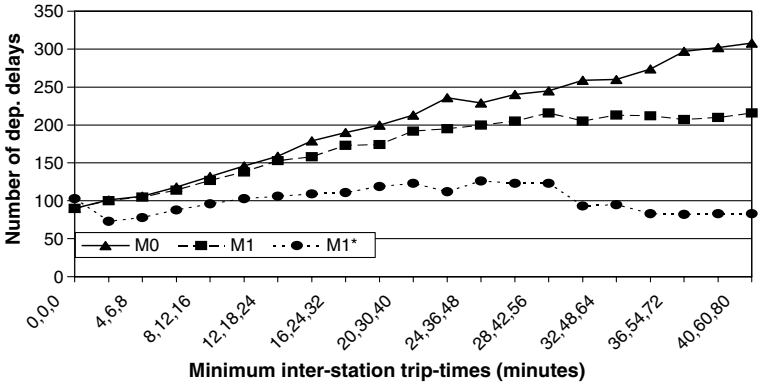

Fig. 6. Numbers of *departure* delays, with Algorithms M0, M1 and M1* and increasing minimum trip-times between stations.

how the results are affected by varying the minimum trip times between stations, we assumed minimum trip times of (2, 3, 4) for the three train types, then (4, 6, 8), (6, 9, 12), and so on. In each case we re-ran the three algorithms, and the results are illustrated in Figs. 5 and 6.

As expected, the delays recorded in Figs. 5 and 6 increase as the inter-station trip times increase. The reason is as follows. In Figs. 5 and 6 (as in Figs. 1–4) we have assumed a single line between stations. This means that trains with different trip times (e.g., 40, 60 and 80 min) are using the same line. After each slow (60 min) train, any fast (40 min) train departing in the next 20 min will catch up with the slower (40 min) train ahead. To

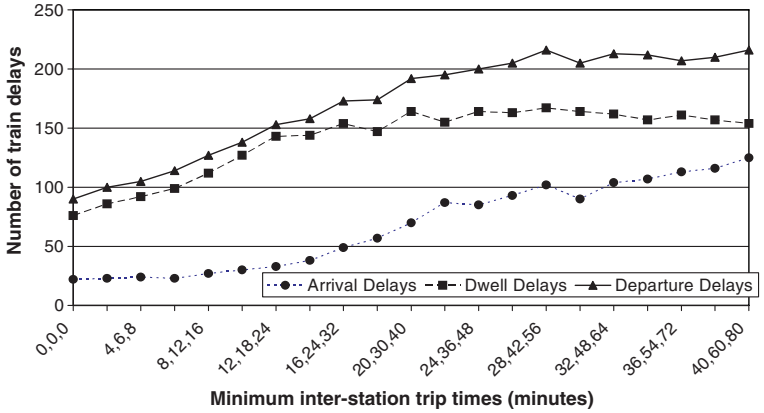

Fig. 7. Effects on delays, of increasing the minimum trip times between stations, for Algorithm M1.

avoid this, as far as possible we should avoid sending a fast train just after a slow train. However, if there is a mixture of train types (speeds) we cannot always let trains with the same speeds depart in succession. We eventually have to let a fast train go following a slow train and in that case there will be a conflict. However, since there are trains ready to depart almost every minute during rush hour, this means delaying several trains after a slow train, and the number of trains affected by this knock-on delay is larger if the differences in inter-station trip-times are larger. Along the horizontal axis in Figs. 5 and 6 the inter-station trip times are increasing hence the differences in the inter-station trip times are increasing, hence the numbers or durations of train delays tends to increase.

In practice, the delays may not increase as fast as in Figs. 5 and 6 with increases in the (differences between) inter-station trip times. The reason is, in practice different lines are often used for trains of different speeds, when this is possible and when they are available. But we are here assuming only a single line in each direction, since this is a more challenging test for our algorithms.

Perhaps the most surprising aspect of the above comparisons in Figs. 5 and 6 is that, for inter-station minimum trip times less than about (10, 15, 20 min), Algorithm M0 does as well as Algorithms M1 and M1*.

In Fig. 7 we compare the arrival, dwell and departure time delays obtained using Algorithm M1. We can obtain similar graphs for Algorithms M0 and M1* and, as expected from Figs. 4 and 5, the graph lines for M0 will slope up more steeply than for M1, and the graph lines for M1* will slope up less steeply than for M1. In each case the departure delays are larger than the arrival or dwell delays.

Note that if a train has an arrival delay and/or a dwell delay this is quite likely to cause a departure delay so that, in each graph of departure delays, a large part of the delay is due to arrival or dwell delays. The arrival, dwell and departure delays should therefore not be added up to give a "total" delay as that would be double counting. An arrival or dwell delay does not always cause a departure delay, since the draft departure time may allow for some delay in arriving or dwelling.

We can significantly reduce the departure delays in the above graphs by changing the dwell time rules assumed in the algorithms. We have assumed that the minimum dwell times have to be met. However, in practice, these minimum dwell times are often only desired dwell times, and train planners in practice allow somewhat smaller dwell times if this will reduce the train delays. We can adopt this practice here by using the desired dwell times when drawing up the draft or desired timetable, but allowing dwell times down to an absolute minimum when resolving conflicts.

For example, if a train's arrival time is delayed, so that its arrival time plus desired dwell time exceeds the desired departure time, than allow only the absolute minimum dwell time.

5. Concluding remarks

In this paper we took an algorithm for scheduling trains at a single-station (from Carey and Carville, 2003) and extended it to obtain algorithms for multiple stations with single or multiple one-way lines in each

direction between the stations. In the first Algorithm M0 developed here we assume that when the current train t conflicts with any other train t' the conflict is resolved by adjusting the times of the current train t. In the second Algorithm M1 we assume that conflicts are resolved by adjusting the times of either train, depending on which requires the smaller adjustments or smaller costs or penalties. Somewhat surprisingly, we found that allowing this choice, of which train to adjust or delay, did not significant improve the results (delays) compared to Algorithm M0, if we applied it only to resolving the ordinary headway conflicts between trains entering or exiting at a single station (but not to conflicts on lines between stations). Because this made little change compared to Algorithm M0 we do not report numerical examples for it here. However, in Algorithm M1 we apply the new procedure (adjusting the times of the current train t or the other trains t') to resolving conflicts between trains at station exits *and* conflicts between trains on lines between stations. In Algorithm M1* we extend this also to resolving conflicts between trains using the same platform.

We applied each algorithm to a test network consisting of 25 busy complex stations, with a draft timetable. We designed the network and initial draft timetables to ensure a large number of conflicts to be resolved, hence providing a better test of the algorithms. All three algorithms resolved all conflicts quite fast, but the second Algorithm M1 gave much better solutions than the first, and similarly the third (M1*) gave better solutions than the second. However, this improvement is not guaranteed, and it is possible to construct an example in which there is no improvement. In view of this, in ongoing and future research we are developing further extensions and combinations of these algorithms. In an appendix of Carey and Crawford (2005) we illustrate how the solution is affected by having several lines, rather than only a single line, in each direction, between stations, or by having trains with the same speed rather than different speeds.

In the computational examples presented in this paper we assumed that the times of any or all trains at all stations could be adjusted, to produce a better feasible schedule. However, in practice train planners are often concerned with finding feasible times and platforms for a subset of trains, or even for a single train, while holding the times and platforms of all other trains fixed. The algorithms in this paper can be used in this way. Each of the algorithms maintains a list of scheduled trains and a list of trains still to be scheduled or re-scheduled, and trains are moved between these lists as the algorithms proceed. We can easily flag trains to stay permanently in the former list.

The algorithms set out in this paper may be used in various ways. They may be used to generate feasible train timetables and platform allocations for a series of stations, or to explore the effects of proposed changes in schedules. They may also be used in exploring how a schedule would be affected by proposed changes in operating policies (e.g., changes in minimum headways or dwell times), proposed changes in train services (e.g., numbers or times of trains), proposed changes in the infrastructure (layout of lines, platforms, signals, etc.), or the effects of a line or platforms being temporarily out of use.

Acknowledgements

This research was supported by a UK Engineering and Physical Science Research Council (EPSRC) grant number GR/K/75798, which is gratefully acknowledged. The authors also wish to thank anonymous referees and the editor of this special issue for their comments and suggestions, and wish to thank various people within Network Rail, formerly Railtrack, formerly British Rail, for information and discussions over several years.

Appendix A. Network and data used in examples

The data for the network, trains and initial draft timetables used in the computational examples in Section 4 are as follows.

A.1. A basic test network

Since this is a test network, we wish to keep its structure relatively simple and regular in order to understand its behaviour. In scientific experiments, when examining the influence of one variable on another, it is standard practice to hold all other variables constant if that is possible. Hence we hold the station type and layout constant for all stations. That is, we assume that the stations $s = 1, \ldots, N$, are all identical. Further, in our basic

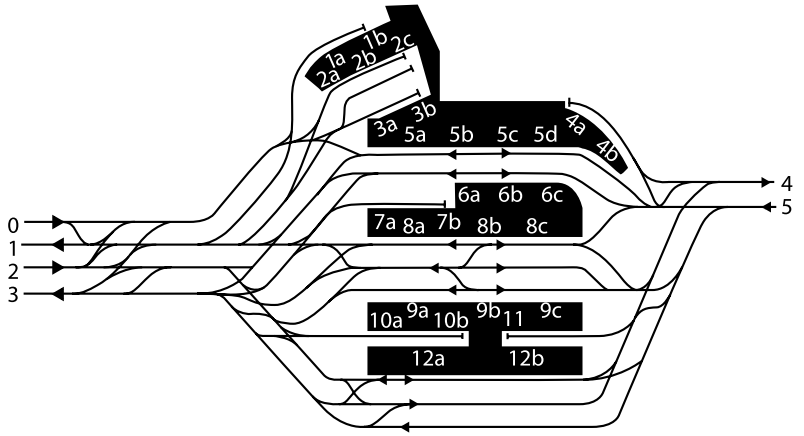

Fig. A.1. Lines and platforms at Leeds station (adapted from Quail, 1996).

experiments we assumed that they are all identical to Leeds station (Fig. A.1, adapted from Quail Map Company (1996)). We chose Leeds since it a busy complex station, which has a mixture of platform types (through platforms and terminal (or 'bay') platforms, sub-divided platforms, short and long platforms) and experiences all types of conflicts and delays. The relevant data for Leeds station is set out in an appendix to Carey and Carville (2003).

A.2. Lines between stations

As can be seen from Fig. A.1, Leeds station has 4 lines to the West and 2 to the East, hence so have all the stations $s = 1, \ldots, N$. To connect these stations (Fig. A.2), we connect lines 4 and 5 to the East of station s to lines 2 and 3, respectively, to the West of the next station $s + 1$. We do this since in practice these lines carry almost all of the "through" traffic at Leeds station. The other 2 lines (0 and 1) to the West of the station branch off, and after they leave the station they are not included in the present network (see Fig. A.2).

A.3. Trains, in-lines and out-lines

The number of trains and their in-lines and out-lines are taken from a draft BR timetable for 1994. There are 490 trains and the numbers on each in-line and out-line are given in Fig. A.3 and Table A.1. We divided the trains into Through trains and Local trains. Through trains are assumed to travel through all stations, from 1 to N or from N to 1. In contrast, it is assumed that Local trains do not travel through any other stations in the present network. Just outside each station they branch off onto other lines not included in the present network, and can of course cause headway conflicts when entering or exiting from the station. It is assumed that the set of Local trains, and their in-lines and out-lines, is replicated at each station. The trains using each station are divided into Through and Local trains as follows.

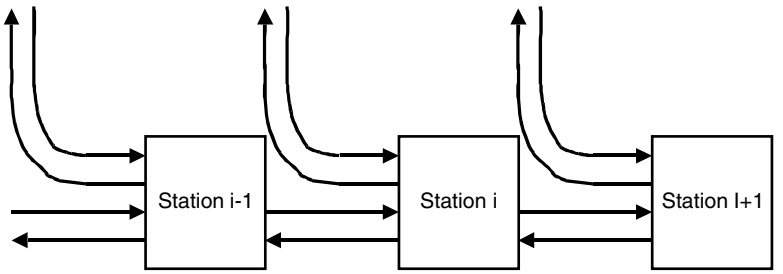

Fig. A.2. Three neighbouring stations from the corridor of 25 interconnected stations.

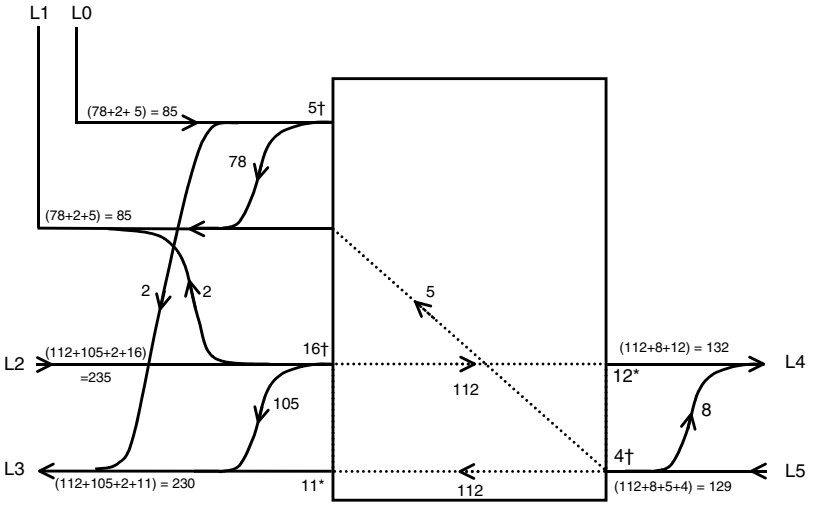

Fig. A.3. Schematic of train movements at each station.

Table A.1
Number of trains going from each in-line to each out-line

		Out lines				Total in
		1	3	4	+	
In lines	0	78	2	–	5	85
	2	2	105	112	16	235
	5	5	112	8	4	129
	*	–	11	12	–	23
Total out		85	230	132	25	472

+ Trains terminating at the station.
* Trains originating at the station.

A.3.1. Through trains

One hundred and twelve West–East trains entering on line 2 and exiting on line 4. One hundred and twelve East–West trains entering on line 5 and exiting on line 3.

A.3.2. Local trains

All other trains, that is: 195 turnaround trains (arrive and depart from same side of station), 23 originating trains, 25 terminating trains, 5 trains which pass through from line 5 to line 1.

A.3.3. Additional through trains

In some runs of the model we invent additional Through trains by converting some of the Local trains to Through trains. To do this we take the 12 trains which originate at the station and exit on line 4 and instead assume that these continue on through all stations, entering each station on line 2 and exiting on line 4. Similarly, we take the 4 terminating trains which arrive on line 5, and assume that these have travelled through all stations, entering each station on line 5 and exiting on line 3.

A.4. Platforms at each station

At Leeds station there are 12 platforms divided into a total of 33 sub-platforms, and the same is assumed at the other stations.

A.5. Platform feasibility

Which platforms are feasible for each train is assumed to be the same as for Leeds station. One aspect of feasibility, namely which lines are connected to each platform, was obtained from Fig. A.1.

A.6. Minimum headways

The minimum headways required between trains (i.e., $H^{(2)}$ in Eq. (1) etc.) are 1–4 min depending on the train types and on whether they are arriving or departing, and the minimum requirements are assumed to be the same at all stations.

A.7. Draft arrival, departure and dwell times at each station

The draft arrival, departure and dwell times of all trains at station 1 are assumed to be those for Leeds station, and are obtained from an internal British Rail draft timetable for the station. The draft arrival, departure and dwell times at all other stations (2 to N) are computed from those for station 1, as follows.

A.7.1. Through trains

For trains travelling from station 1 to N, the draft arrival time at station 2 is the draft departure time from station 1 plus the draft trip time from 1 to 2 (e.g., 20 min). The draft departure time from 2 is the draft arrival time at 2 plus the draft dwell time at 2. And so on at stations 3 to N. Similarly, for trains travelling N to 1, the draft times at stations 2, 3, etc., are obtained by working backwards from station 1, subtracting inter-station trip times and dwell times at each station 2, 3, etc.

A.7.2. Overnight stops

Almost all trains in our example are passenger trains, and we assume these do not run during the night. We therefore assumed that if a through train would depart from a station after 11.00 pm it instead spends the night there, or in a depot, and its draft arrival time at the next station is $(8 + x)$ hours after its arrival at the current station, where x is the trip time between the stations.

A.7.3. Trains originating, terminating or turning around

We assume that the draft arrival, departure and dwell times for these are the same at each station as at station 1.

References

Carey, M., 1994a. A model and strategy for train pathing with choice of lines, platforms and routes. Transportation Research B 28 (5), 333–353.

Carey, M., 1994b. Extending a train pathing model from one-way to two-way track. Transportation Research B 28 (5), 395–400.

Carey, M., Carville, S., 2000. Testing schedule performance and reliability for train stations. Journal of the Operational Research Society 51 (6), 666–682.

Carey, M., Carville, S., 2003. Scheduling and platforming trains at busy complex stations. Transportation Research A 37 (3), 195–224.

Carey, M., Crawford, I., 2005. Scheduling trains through a corridor of busy complex stations. Research Report, Queen's University, Belfast, N. Ireland (A longer version of the present paper containing an appendix omitted here for reasons of space).

Carey, M., Lockwood, D., 1995. A model, algorithms and strategy for train pathing. Journal of the Operational Research Society 46B (8), 988–1005.

Cordeau, J.-F., Toth, P., Vigo, D., 1998. A survey of optimization models for train routing and scheduling. Transportation Science 32, 380–404.

Jovanovic, D., 1989. Improving railroad on-time performance: models, algorithms and applications, PhD thesis, University of Pennsylvania, PA, USA.

Jovanovic, D., Harker, P.T., 1991. Tactical scheduling of rail operations: the SCAN I system. Transportation Science 25 (1), 46–64.

Kraay, D., Harker, P.T., Chen, B.T., 1991. Optimal pacing of trains in freight railroads: model formulation and solution. Operations Research 39 (1), 82–99.

Kroon, L.J., Romeijn, H.E., Zwaneveld, P.J., 1997. Routing trains through railway stations: complexity issues. European Journal of Operational Research 98, 458–498.

Odijk, M.A., 1996. A constraint generation algorithm for construction of periodic railway timetables. Transportation Research B 30 (6), 455–464.

Petersen, E.R., Taylor, A.J., Martland, C.D., 1986. An introduction to computer aided train dispatching. Journal of Advanced Transportation 20, 63–72.

Quail Map Company, 1996. British Rail Track Diagrams 2: Eastern and Anglia Regions. Quail Map Company, Exeter, England.

Watson, R., 2001. The effects of railway privatisation on train planning. Transport Reviews 21 (2), 181–193.

Zwaneveld, P., Kroon, J.L., Romeijn, H.E., Salomon, M., 1996. Routing trains through railway stations: model formulation and algorithms. Transportation Science 30 (3), 181–194.