A multicommodity flow model for rerouting and retiming trains in real-time to reduce reactionary delay in complex station areas

Edwin Reynolds*[1], Matthias Ehrgott[2], Stephen J. Maher[3], Anthony Patman[4], and Judith Y.T. Wang[5]

[1]*STOR-i, Lancaster University, Bailrigg, Lancaster, LA1 4YX, UK*
e.s.reynolds@lancaster.ac.uk
[2]*Department of Management Science, Lancaster University, Bailrigg, Lancaster, LA1 4YX, UK*
m.ehrgott@lancaster.ac.uk
[3]*Department of Mathematics, University of Exeter, Exeter, EX4 4QF, UK*
s.j.maher@exeter.ac.uk
[4]*Eastern Region, Network Rail, George Stephenson House, Toft Green, York, YO1 6JT, UK*
anthony.patman@networkrail.co.uk
[5]*School of Civil Engineering & Institute for Transport Studies, University of Leeds, Leeds, LS2 9JT, UK*
j.y.t.wang@leeds.ac.uk

Abstract

By rerouting and retiming trains in real-time, the propagation of reactionary delay in complex station areas can be reduced. In this study, we propose a new optimisation model and solution algorithm that can be used to determine the best combination of route and schedule changes to make. Whilst several models have been proposed to tackle this problem, existing models either lack sufficient detail, or cannot be solved to optimality within the stringent time limits associated with a real-time environment. We formulate the problem as a multicommodity flow problem on a time-space graph with a novel representation of the track capacity constraints and a new objective function based on utility maximisation. We present a tailored branch-and-price solution algorithm and test it on a set of new instances based on real data. Our experiments show that most of these instances can be solved to optimality within 20 seconds, and provably near-optimal solutions can be found for the remainder.

Keywords— railway optimization, timetable rescheduling, multicommodity flow, branch-and-price

1 Introduction

Delays to passenger trains are a significant problem in Great Britain. In 2018–2019, only 65.3% of recorded passenger train stops occurred on time [Network Rail, 2020], and the

*Corresponding author.

time that passengers lose each year as a result of arriving late at their destinations has been estimated to be worth at least £1 billion [National Audit Office, 2008, p. 46]. In 2019, 64.35%[1] of total train delay minutes were due to *reactionary delays*, which are delays that are caused by the knock-on effect of prior delays. The prevalence of reactionary delays is therefore a very significant part of the overall problem.

In this paper, we present a model and algorithm that can be applied to reduce the amount of reactionary delay after an initial delay incident. This is achieved by solving the Train Timetable Rescheduling Problem (TTRP) [Cacchiani et al., 2014]. The TTRP consists of calculating new schedules (called *rescheduling*) for trains in real-time that are different from those originally planned. These new schedules should maximise a measure of utility so as to represent the best possible response to the delay incident. Rescheduling consists of both *rerouting* and *retiming*, finding new routes and timings for a train, respectively. Rerouting implicitly includes *replatforming*, finding new platforms at the same station for a train to stop at. The TTRP is solved within a defined area of a railway network over a fixed time horizon. The new schedules should be practically achievable under the constraints of the signalling system.

Although the TTRP has already benefited from a significant amount of research attention, which has been well documented by Corman and Meng [2015], Cacchiani et al. [2014] and Fang et al. [2015], the successful deployment of real-time optimization algorithms as a decision support tool for timetable rescheduling has so far been limited. One reason for this is that researchers have been unable to solve realistic instances to optimality in the strict solution time limits required in a real-time environment. Instead, some have relied on heuristic methods that offer no guarantees on the quality of solutions found. Others have proposed simplified models that either carry out rerouting and retiming separately, approximate the signalling constraints or use simplistic objective functions.

In this study, we propose a multicommodity flow model for the TTRP that performs rerouting and retiming simultaneously. The model employs a novel representation of the signalling constraints that is accurate and detailed, yet leads to a tractable formulation of the problem. Our objective function uses the concept of utility to flexibly represent complex rescheduling preferences. We present a new set of instances based on real data from an area of the British railway network centred around Doncaster station. Finally, we show that these instances can be solved to either optimality or provably near to optimality in sufficiently short times using a tailored branch-and-price algorithm.

This research has been carried out in collaboration with Network Rail, the railway infrastructure manager in Great Britain. In 2018–2019 on their network, 1.8 billion passenger journeys were made, totalling 68 billion passenger kilometres. Network Rail is seeking to use new technology to improve their ability to manage disruption on the railway, and therefore to improve the reliability of passenger services. This represents a good opportunity for optimization-based rescheduling to have an impact.

The paper is organised as follows. In Section 2, we define some prerequisite railway signalling concepts and terminology before reviewing the current literature in Section 3. Section 4 introduces the model. In Section 5, we show how to formulate the problem as an integer program, which can be solved using the branch-and-price solution algorithm presented in Section 6. Section 7 describes the creation of the test instances, Section

[1]Statistic provided by Network Rail.

8 presents our computational results, and finally Section 9 offers our conclusions and suggested directions for future research.

2 Railway Signalling

In order to ensure safe operation of the railway, the movement of trains is regulated by a signalling system. Signalling is a complex topic that is covered in detail by Pachl [2014]. For conciseness, in this paper we will focus only on the details relevant to our model for the TTRP. Although we describe the workings of the signalling system in Great Britain, it is similar elsewhere. The signalling system controls train movements at stations, junctions and on the lengths of uninterrupted track that run between then, called *open lines*. Signalling for stations and junctions is different to the signalling on open lines.

The vast majority of open lines in Great Britain are signalled with single-direction *Automatic Block Signalling*. In this system, the track is divided into *block sections*. Each block section is protected by a signal that is visible to drivers in advance of arrival at the block section. Signals display a green aspect to indicate that an approaching train may pass the signal and enter the block section. In contrast, a red aspect indicates that an approaching train must stop before entering the block section. Signals protect block sections by ensuring that there is at most one train in each block section at any one time and therefore that trains cannot collide. Though we do not include them in our model, yellow and double-yellow aspects are often used to indicate caution, warning trains to slow down when the next or next-but-one signal, respectively, is red.

In contrast, stations and junctions are usually signalled with an interlocking system. This is a more complicated signalling arrangement designed to cope with the increased complexity of converging, diverging and crossing tracks within these areas. The track within an interlocked area is divided into *track circuits*. In a station, some of these track circuits contain platforms. Signals are placed both at the limits of an interlocked area and at the end of platforms. The red and green aspects of these signals have the same meanings as the red and green aspects of signals on the open line. Trains travelling through interlocked areas begin and end this traversal at signals at the limit of the interlocked area. During this traversal, they may also visit signals positioned at platforms.

In an interlocked area, the route from one signal to an adjacent signal consists of a sequence of track circuits. The signal at the start of a route will allow a train to traverse the route only if it has first been *locked*. The interlocking system prevents a route from being locked unless all track circuits that make up the route are free, i.e. are not currently locked for another train. Therefore, when a route is locked for a train, no other route with track circuits in common can be locked. Since they cannot be locked, they cannot be traversed. In *route-release* interlocking systems, the track circuits in a route are *released* simultaneously when the entire route is cleared. In *sectional-release* interlocking systems, each track circuit is individually released when it is cleared. We will show how to model both of these release systems.

In our model, the open line is treated as an interlocked area. This is because the Au-

tomatic Block Signalling system used on the open line can be seen as a special case of interlocking, where each block section consists of a single track circuit. This observation is valid regardless of the interlocking release type.

In Great Britain, and in this paper, the word *berth* is used to refer to signals. We use the word *route* to mean the track that must be traversed to get from one berth to another. We use the terms berth and route regardless of whether the signals are on the open line or in an interlocking area. We define the *traversal time* to be the minimum amount of time required to traverse a route. We define the *headway* to be any additional time required between a train vacating a route or track circuit, and it being released and ready for use by another train. The sum of these two times is called the *blocking time*.

3 Literature Review

Since the literature on railway optimisation is very broad, we will focus specifically on the TTRP and its offline equivalent, the Train Routing Problem (TRP) where relevant.

Models for the TTRP divide railway track into discrete *track components*. This facilitates the representation of both the locations of trains over time, and the signalling constraints. Track components of different types can be used, such as track circuits, block sections, platforms or even whole stations or lines. The type of the track components used affects the granularity of the model. *Macroscopic* models usually represent stations using nodes and the tracks in between stations using edges. This approach allows for large areas of a railway network to be modelled, but fails to capture signalling constraints accurately. As a result, macroscopic models are unsuitable for rescheduling in complex station areas. *Microscopic* models usually model individual track circuits or block sections. While models that use block sections are very common in the rescheduling literature, they cannot model sectional-release interlocking systems accurately. As a result, such models often underestimate the capacity of large station areas [Corman et al., 2009; Pellegrini et al., 2014]. The capacity of station areas with sectional release interlocking systems can only be accurately represented by models that use track circuits, such as those proposed by Corman et al. [2009], Pellegrini et al. [2014], Caimi et al. [2011] and Lusby et al. [2013]. The main disadvantage of these models is that they are often intractable for large stations due to the large number of decision variables required to model the considered track circuits.

Models for the TTRP can also be classified according to the way in which time is modelled. *Disjunctive* models use continuous variables to represent the times at which trains begin using individual track components. *Time-indexed* models, on the other hand, divide time into discrete time intervals and use binary variables to indicate whether or not each train uses a given track component during a given time interval. Disjunctive and time-indexed models represent the constraints imposed by the signalling system differently. In both cases, signalling constraints are represented by capacity constraints on the track components. In disjunctive models, for a given track component, one capacity constraint is defined for each pair of trains that use the component. These constraints are in the form of a disjunction between the two possible orders in which the pair of trains can use the track component. In contrast, a capacity constraint on a track component in a time-indexed model can be represented by an upper bound on the number of trains that

use each track component in each time interval. A time-index model has been developed for the TTRP in this paper. The relative merits of disjunctive and time-indexed models are discussed below.

One of the most popular models for the TTRP is the Alternative Graph model of D'Ariano et al. [2007]. This is a disjunctive model that was originally developed by Mascis and Pacciarelli [2002] for machine scheduling. It is suitable for railway retiming because when the routes of the trains are fixed, the TTRP becomes a variant of the job-shop scheduling problem. D'Ariano et al. [2007] present an exact branch-and-bound procedure for minimising the maximum reactionary delay, which is incorporated in a full dispatching system described by D'Ariano et al. [2008]. This branch-and bound-algorithm is further developed by Mannino and Mascis [2009], who propose stronger lower bounds to speed up the enumeration process.

The Alternative Graph model has two important shortcomings. The first is that it cannot model rerouting. The ability to perform rerouting is crucial to finding good rescheduling solutions in stations with many routing alternatives. Though the model has been extended by Corman et al. [2010] to incorporate rerouting, the authors were only able to find heuristic solutions to the resulting problems. The second shortcoming of the Alternative Graph model is that the exact branch-and-bound algorithm of D'Ariano et al. [2007] can only be used when the objective function is to minimise the maximum of some measure over all station stops. The advantages of instead optimising the total of some measure over all station stops is discussed in Section 4.8. Although Samà et al. [2015] have shown that the model can be extended to accommodate many different objective functions, the resulting problems can only be solved with generic Mixed Integer Programming (MIP) solvers, the drawbacks of which are discussed next.

In order to overcome the shortcomings of the Alternative Graph model, Törnquist and Persson [2007], Pellegrini et al. [2015], Meng and Zhou [2014], Min et al. [2011] and Acuna-Agost et al. [2011] have proposed disjunctive MIP formulations for the problem. All of these models allow trains to be rerouted, and they all include either total delay or total weighted delay as part of the objective function. However, the representation of disjunctive constraints in a MIP requires the use of big-M constraints. As a result, these models have weak Linear Programming (LP) relaxations. This makes them difficult to solve to optimality using LP-based branch-and-bound in the short run times required for real-time rescheduling. Although both Samà et al. [2016] and Törnquist Krasemann [2012] have proposed heuristics for these models with acceptable solution times, it is not possible to know the quality of any solutions produced by these algorithms.

Time-indexed models are an alternative to disjunctive models which avoid the computational difficulties associated with disjunctions. In time-indexed models, the time horizon is divided into discrete time intervals. This allows *time-space resources* to be defined, each of which is a pair consisting of one of track component and one time interval. Typically, a binary variable is used to indicate whether a train consumes a given time-space resource. Rescheduling can be seen as problem of assigning time-space resources to trains. The track capacity constraints can be represented by introducing one set packing constraint for each time-space resource. Constraints are also required to ensure that the assignment of time-space resources for each train represents a feasible itinerary. This can be achieved using a time-space graph, where each node represents a time-space resource and each directed arc is a feasible transition between them. An allocation of time-space resources

to a train can be represented as a path in the time-space graph.

Time-indexed models have been predominantly used in offline railway optimisation problems such as the Train Timetabling Problem (TTP) [Brännlund et al., 1998; Caprara et al., 2002; Borndörfer and Schlechte, 2007; Cacchiani et al., 2008], train routing [Zwaneveld et al., 2001; Lusby et al., 2011; Harrod, 2011; Caimi et al., 2011] and train platforming [Caprara et al., 2011]. In contrast, time-indexed models have not been widely used for the TTRP, due to a perception that they contain too many variables for the solution of real instances in the stringent time limits imposed by real-time environments. Recently, however, some authors have successfully used time-indexed models for real-time applications. Lusby et al. [2013] use a branch-and-price algorithm for the TTRP, whilst Meng and Zhou [2014] use a time-indexed formulation for rescheduling on a railway network with multiple parallel tracks, and Bettinelli et al. [2017] present an effective iterative heuristic for real-time rescheduling.

Since time-indexed models can produce very large formulations, most authors decompose the problem so that each binary variable represents a full train path. Though this has been achieved in some settings by the enumeration of train paths [Zwaneveld et al., 2001; Caprara et al., 2011], it is most common to dynamically generate each path variable. This has been achieved within both Lagrangian relaxation [Brännlund et al., 1998] and column generation [Borndörfer and Schlechte, 2007; Cacchiani et al., 2008; Lusby et al., 2013] frameworks. Both Caimi et al. [2011] and Lusby et al. [2013] generate new path variables using tree structures that capture the sequence of decisions taken for each train over time. Though this approach allows detailed train speed profiles to be considered, it is not scalable, since the size of the trees grows exponentially with the number of decisions in this sequence. Our model avoids this problem, because new path variables are generated by solving shortest path problems on a time-space graph. Since the time-space graph is directed and acyclic, the complexity of these problems is linear in the number of edges in the time-space graph.

In this paper, we make the following contributions:

1. We present a new model for the Train Timetable Rescheduling Problem that is capable of performing rerouting and retiming simultaneously.
2. We demonstrate that it is possible to model a dense, sectional-release interlocking area accurately by using routes as the track components. This is made possible by introducing a distinction between two types of capacity consumption: *occupying* and *banning*. This distinction allows track circuits to be modelled implicitly. Our model is an improvement on models that explicitly use track circuits as the track components. Such models can be intractable due to the large number of variables.
3. We present a new objective function, which was developed in collaboration with Network Rail. This objective function uses the concept of utility to represent Network Rail's preferences more accurately than existing objective functions in the literature.
4. We present a new set of instances based on real data for an area around Doncaster station in the UK. We provide important details of how we generated the instances, the provision of which is scant in much of the other published literature.
5. We show that these instances can be solved either to optimality or provably near to

optimality in times suitably short for real-time operations. We present a tailored branch-and-price algorithm for this purpose in which the subproblem for each train is a shortest path problem, and therefore efficiently solvable. We demonstrate the success of several acceleration strategies from the column generation literature.

6. We show that it is now realistic to solve time-indexed models for the TTRP to optimality in real-time environments.
7. We demonstrate, both theoretically and empirically, the relationship between the number of conflicts in an instance and its difficulty.

4 Modelling the Train Timetable Rescheduling Problem

4.1 Description of the Problem

Following a disturbance to the timetable, the problem is to find a new route and set of timings (i.e. a new schedule) for each controlled train such that there are no conflicts between trains and the passenger utility is maximised. Controlled trains are those that are forecast to be inside a defined area of track during a time horizon. The new route can involve stopping at different platforms from those originally planned, taking different approaches to planned platform stops, or cancelling the stop altogether. There are no conflicts between trains if and only if the new schedule can be implemented in practice under the signalling system. The passenger utility is the total weighted utility over all train stops which are carried out, where at each stop the utility is a function of lateness.

4.2 Modelling Approach

Our approach to modelling the problem is to build a directed, acyclic time-space graph $G = (N, A)$. Each arc $a \in A$ has an arc weight c_a^k for each train $k \in \mathcal{K}$. In addition, we specify two different sets of arcs, $A_n \subseteq A$ and $\bar{A}_n \subseteq A$, for each node $n \in N$. The graph G is configured such that every finite-weight *source-sink* path describes a feasible timed train route through the modelled area. If a train k is assigned such a path, its total weight is the negative of the utility of the train carrying it out. The arc sets A_n and $\bar{A}_n$ are used in our novel representation of the track capacity constraints, which is described in Section 4.6.

The TTRP is formulated as follows:

TTRP. *Find a source-sink path in G for each train $k \in \mathcal{K}$ such that the total weight of all the paths is minimised, subject to the constraints that for each node $n \in N$, and across all train paths (i) at most one arc in A_n is used and (ii) if an arc in A_n is used then no arcs in $\bar{A}_n$ are used.*

The rest of this section explains how the graph G, the arc weights c_a^k and the arc capacity sets A_n and $\bar{A}_n$ are built. In order to describe how G is built in Section 4.5, we first

explain how time is modelled in Section 4.3 and how the track is modelled in Section 4.4. The track capacity constraints are explained in Section 4.6. Sections 4.7 and 4.8 explain the objective function, and details about the arc weights c_a^k are given in Section 4.9.

4.3 The Time Horizon

Let the time horizon over which trains should be rescheduled be $[0, T]$. The time horizon should be long enough to allow a wide variety of rescheduling options to be considered, but not so long that new primary delays during the time horizon are likely to occur, making further rescheduling necessary. We use a time horizon of length 60 minutes in our experiments.

The time horizon is divided into a discrete set of consecutive time intervals $\mathcal{T} = \{0, \ldots, T\}$ of equal length. The number of time intervals should be large enough to offer a reasonable approximation of continuous time, but not so large that the problem becomes intractable. We use time intervals of length 15 seconds in our experiments, resulting in 240 time intervals overall.

4.4 The Route Graph

Let N_b be the set of berths within the area of track being modelled, and let $A_b \subset N_b \times N_b$ be the set of permissible transitions between these berths (i.e. routes). Together, these form the directed berth graph $G_b = (N_b, A_b)$. The graph G_b is typically connected, although this is not necessary. A small example for illustrative purposes is provided in Figure 1(a).

From the berth graph, we can derive the route graph $G_r = (N_r, A_r)$, where $N_r = A_b$ and A_r consists of directed arcs between routes that together form a path of length 2 in G_b. This is sometimes known as the line graph of G_b, and is shown in Figure 1(b). A route consists of the track between two berths, or signals. A traversal of $r = (b_1, b_2)$ is defined as starting when a train passes signal b_1 and finishes when it passes signal b_2.

Each route $r \in N_r$ has data associated with it. Let L_r be the minimum number of time intervals required to traverse the route. Let h_r be the number of time intervals that should be left as headway between any two traversals, so that $L_r + h_r$ is the blocking time for route r. Let $TC_r = \{tc_r^i\}_{i=0}^{n_r-1}$ be the set of track circuits for route r, in order of traversal from tc_r^0 to $tc_r^{n_r-1}$ Our methodology for collecting and in some cases inferring this data is described in Section 7.

Finally, we expand G_r to take account of platforms. Let PL be the set of platforms. Each platform $pl \in PL$ occurs immediately before a berth $b^{pl} \in N_b$. This means that each platform pl lies at the end of any route that it occurs within. We truncate each of these routes, so that they finish immediately before the platform pl. We insert two new artificial routes pl_{stop} and pl_{pass}, which both represent the track alongside the platform. Traversal of pl_{stop} corresponds to stopping at platform pl to allow passengers to alight and embark. Traversal of pl_{pass} corresponds to passing the platform without stopping. This separation allows pl_{stop} and pl_{pass} to have different traversal times. Separating platforms

from the routes they lie on also allows us to measure the exact times at which trains arrive at platforms, which is important for the objective function.

In order to describe the procedure for separating these new platform routes, we first define

$$\sigma^-(b^{pl}) = \{(i,j) \in A_r : j = b^{pl}\} \text{ and } \sigma^+(b^{pl}) = \{(i,j) \in A_r : i = b^{pl}\}$$

to be the sets of routes ending and starting at berth b^{pl}, respectively. The steps for separating the new platform routes are as follows:

1. Remove all arcs in A_r that begin at a route in $\sigma^-(b^{pl})$ and end at a route in $\sigma^+(b^{pl})$.
2. Insert the new nodes pl_{stop} and pl_{pass} into N_r.
3. Insert into A_r all arcs from the sets $\{(r,r') : r \in \sigma^-(b^{pl}), r' = pl_{stop}, pl_{pass}\}$ and $\{(r,r') : r = pl_{stop}, pl_{pass}, r' \in \sigma^+(b^{pl})\}$.
4. Set the data for pl_{stop} and pl_{pass}. The traversal and headway times $L_{pl_{pass}}$ and $h_{pl_{pass}}$ for pl_{pass} are set to zero. The traversal and headway times $L_{pl_{stop}}$ and $h_{pl_{stop}}$ for pl_{stop} are set to the minimum dwell time D_{pl} and minimum headway H_{pl} for the platform pl, respectively. D_{pl} is the minimum number of time intervals required to allow passengers to alight and embark at platform pl, and H_{pl} is the minimum number of time intervals required once a train has left the platform pl before another train can arrive. Both pl_{stop} and pl_{stop} consist of the same single track circuit.

Figure 1(c) shows the graph that results from following this procedure in our example.

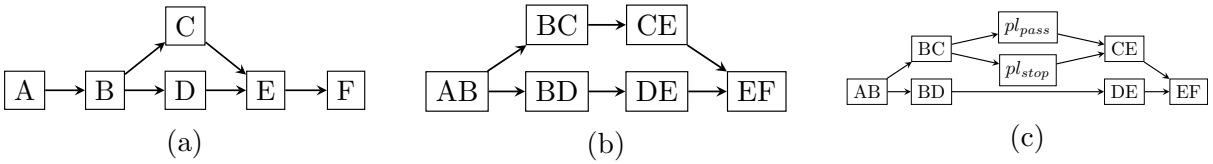


Figure 1: A small example to illustrate the construction of the route graph. (a) shows the berth graph G_b; (b) shows the corresponding route graph G_r before expanding the platforms. (c) shows the route graph expanded for a platform pl that exists at the end of route BC.

4.5 The Time-space Graph

Let the node set $N = (N_r \times \mathcal{T}) \cup \{source, sink\}$ of G consist of all of the route, time interval pairs (r,t), in addition to an artificial source and sink. For convenience, let $N_0 = N \setminus \{source, sink\}$.

The directed arc set $A = \bigcup_{i=1}^{6} A_i$ of G consists of the following groups of arcs:

(i) $A_1 = \{(source, (r_0^k, a_0^k)) : k \in \mathcal{K}\}$ corresponding to entering from the source node to the known location of the train either at the beginning of the time horizon or when the train first enters the area during the time horizon.

(ii) $A_2 = \{(source, sink)\}$ consists of a single arc, corresponding to a train cancellation. This can be omitted if desired.

(iii) $A_3 = \{((r,t),(r,t+1)) : (r,t) \in N \text{ and } (r,t+1) \in N\}$ corresponding to waiting in route r for one time interval.

(iv) $A_4 = \{((r,t),(r',t+L_r)) : (r,r') \in A_r, (r,t) \in N \text{ and } (r',t+L_r) \in N\}$ corresponding to traversing r and arriving in r' after the minimum traversal time L_r.

(v) $A_5 = \{((r,T), sink) : r \in N_r\}$ corresponding to exiting to the sink node at the end of the time horizon.

(vi) $A_6 = \{((r,t), sink) : \sigma^+(r) = \emptyset \text{ and } (r,t) \in N\}$ corresponding to exiting to the sink node from a node at the boundary of the area of track modelled.

An example of G is given in Figure 2. It uses the same track layout as the example in Figure 1.

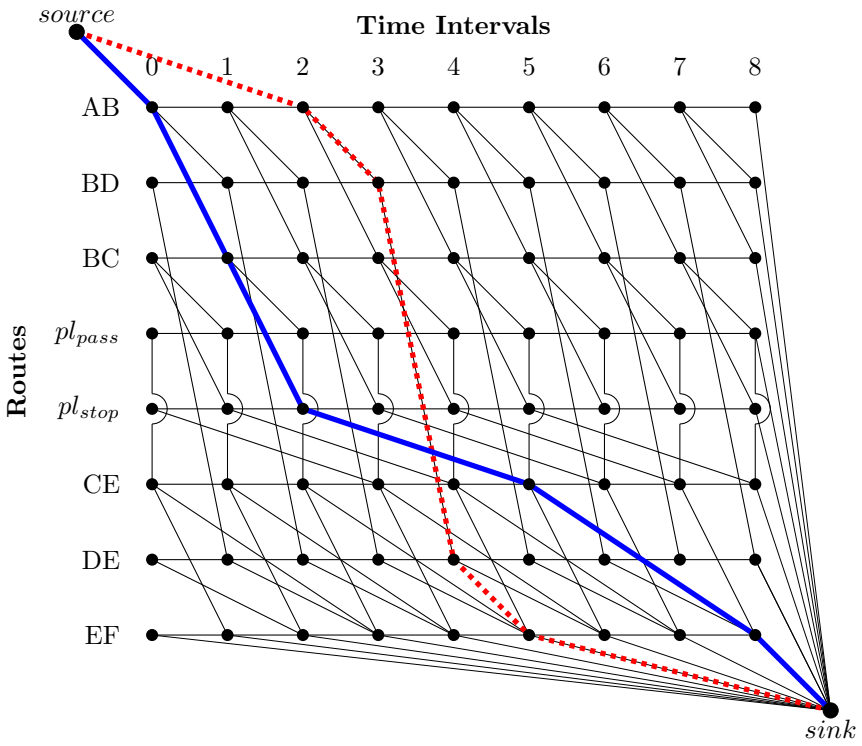


Figure 2: Example time-space graph G corresponding to the route graph example in Figure 1. Two *source-sink* paths are shown.

Note that a *source-sink* path in G specifies the routes that a train is momentarily traversing during every time interval from the current time up until the end of the time horizon. The routes to which a train is assigned also determine the platforms that the train uses. The definition of A_4 ensures that minimum traversal and dwell times are respected. The arcs in A_3 make it possible for a train to wait for additional time at any point in its journey. Waiting should be implemented by making alterations to the speed profile of the train. These alterations should be calculated separately, and are not addressed in this paper.

The directed graph G is acyclic. It has $|N_r||\mathcal{T}|+2$ nodes and typically at least twice that many arcs depending on the track topology. In our experiments, G had 64,082 nodes and 156,594 arcs.

4.6 Track Capacity

For each train $k \in \mathcal{K}$, a finite-weight *source-sink* path in G corresponds to an *individually feasible* schedule over the time horizon. This means that such a path would be feasible if there were no other trains using the track. However, a set containing more than one train path is not *jointly feasible* unless it is possible for all of the train paths to be carried out together under the constraints imposed by the signalling system. The signalling system limits the capacity of the track in the interests of safety, as described in Section 2.

Each node $(r, t) \in N_0$ can be viewed as a time-space resource. We define two different ways in which a train can consume the capacity of a resource (r, t), called *occupying* and *banning*, in Sections 4.6.1 and 4.6.2. These definitions allow the track capacity constraints to be expressed using two constraints on the capacity of each time-space resource. These constraints are described in Sections 4.6.1 and 4.6.2. This novel representation of the track capacity constraints accurately models sectional-release interlocking, despite the fact that routes rather than track circuits are used in the time-space graph. Two sets, $A_{r,t}$ and $\bar{A}_{r,t}$, are defined for each resource (r, t) in Sections 4.6.3 and 4.6.4. These are constructed such that a train path contains an arc in $A_{r,t}$ if and only if it occupies (r, t), and such that a train path contains an arc in $\bar{A}_{r,t}$ if and only if it bans (r, t).

4.6.1 Occupying

Before a train k can traverse a route r, all of the track circuits making up r must first be locked. This can only be done if all of the track circuits are free, i.e. they are not currently locked for another train. Once train k has traversed the route, the track circuits are released according to the release-type (either route-release or sectional-release). If any of the track circuits in route r are locked for train k during time interval t, then we say that (r, t) is *occupied* by k. Occupation is one of the ways in which a train can consume the capacity of a time-space resource.

If (r, t) is occupied by train k, then it is not possible for any other train to occupy (r, t). This is because there are track circuits in r that have not yet been released by k and therefore are not free during time interval t. This gives rise to the first capacity constraint:

Capacity Constraint 1. *No $(r, t) \in N_0$ can be occupied more than once.*

4.6.2 Banning

Track circuits can and do belong to more than one route. For example, Figure 3 shows two routes, r_1 and r_2, that share track circuits 2 and 3. Suppose that train k locks the track circuits of route r_1 in order to traverse r_1. Since r_2 contains track circuits 2 and 3, which have now been locked for train k, r_2 is now unavailable to be locked by another train. This will remain true until k releases track circuits 2 and 3. If r_2 is unavailable for this reason during time interval t, we say that (r_2, t) is *banned* by train k.

More generally, a train *bans* a time-space resource (r', t) if it makes route r' unavailable during time interval t as a result of occupying (r, t), where r is a route with track circuits in common with r'.

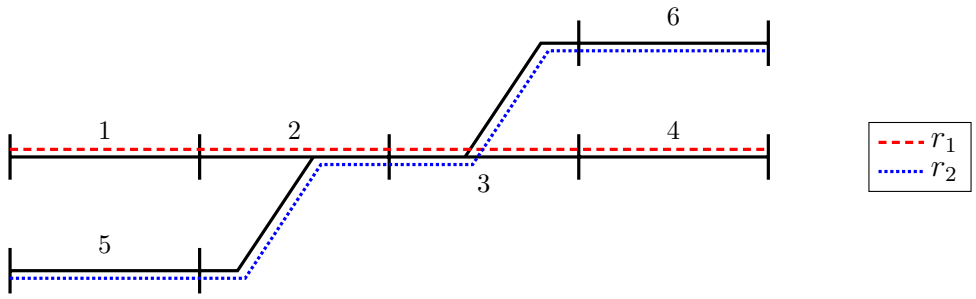


Figure 3: Routes r_1 and r_2 are shown by the dashed red and dotted blue lines, respectively, so that $TC_{r_1} = \{1, 2, 3, 4\}$, $TC_{r_2} = \{5, 2, 3, 6\}$. The two routes share the track circuits $TC_{r_1} \cap TC_{r_2} = \{2, 3\}$.

Notice that if (r', t) is banned by train k, then (r', t) cannot be occupied by k or any other train. This is the second capacity constraint:

Capacity Constraint 2. *No $(r, t) \in N_0$ can be both banned and occupied.*

There is no constraint on the number of times that a time-space resource can be banned. To understand why, consider routes r_1,r_2 and r_3 from the example in Figure 4. Since r_1

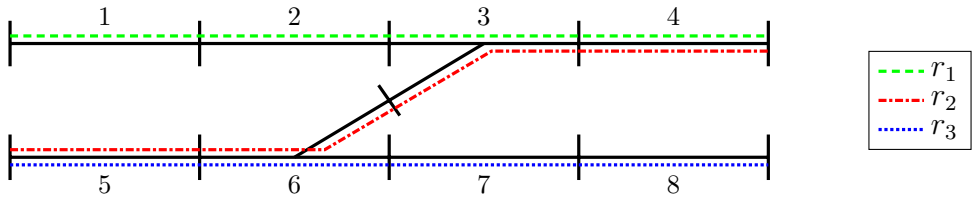


Figure 4: Routes r_1,r_2 and r_3 are shown by the dashed green, dot-dashed red and dotted blue lines, respectively, so that $TC_{r_1} = \{1, 2, 3, 4\}$, $TC_{r_2} = \{5, 6, 3, 4\}$ and $TC_{r_3} = \{5, 6, 7, 8\}$.

and r_3 share no track circuits, it is feasible for (r_1, t) to be occupied by one train, and for (r_3, t) to be occupied by a different train. Since r_2 shares track circuits with both r_1 and r_3, this may involve both of these trains banning (r_2, t). However, no track circuit is locked more than once and therefore there is no infeasibility.

The difference between occupying and banning is subtle. Both occupying a resource and banning a resource preclude the resource from being otherwise occupied. The crucial difference is that whilst a resource cannot be occupied more than once, it may be banned more than once.

4.6.3 Construction of $A_{r,t}$

To enforce the capacity constraints, it is necessary to be able to determine whether a given *source-sink* path in G occupies a given time-space resource (r, t). This is achieved by constructing a set $A_{r,t}$ of arcs such that a path occupies (r, t) if and only if it contains an arc in $A_{r,t}$.

If a train occupies (r, t), then by definition there are track circuits in route r that are locked for the train during time interval t. In other words, the train is traversing r during

time interval t. This traversal lasts for $L_r + h_r$ time intervals, which is the blocking time of route r. Therefore, the traversal began within the preceding $L_r + h_r$ time intervals, and the corresponding path in G must include a node in

$$W_{r,t} = \{(r, t') : t' \in \{t - (L_r + h_r), \ldots, t\}\}.$$

Since it must have entered this node via an arc, the path must contain an arc in

$$A_{r,t} = \bigcup_{(r',t') \in W_{r,t}} \sigma^-(r', t') \setminus \bigcup_{(r',t') \in W_{r,t}} \sigma^+(r', t').$$

The reverse is also true: if the path contains an arc in $A_{r,t}$, then it must contain a node in $W_{r,t}$ and therefore it must occupy the resource (r, t). Therefore, a train path contains an arc in $A_{r,t}$ if and only if it occupies (r, t). An example of the set of nodes $W_{r,t}$ and the set of arcs $A_{r,t}$ corresponding to a time-space resource (r, t) are shown in Figure 5.

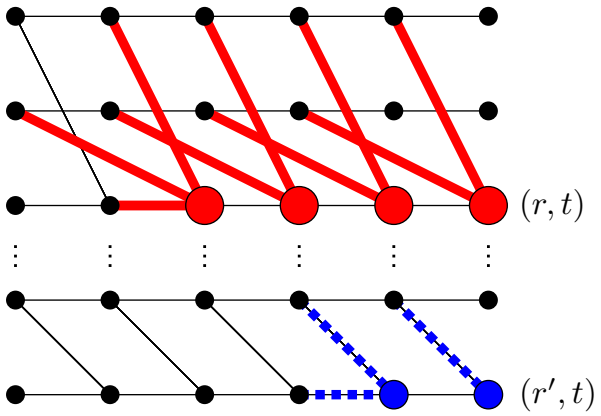


Figure 5: The nodes in $W_{r,t}$ and $\bar{W}_{r,t}$ are shown as large red and medium blue, respectively. The arcs in $A_{r,t}$ and $\bar{A}_{r,t}$ are shown as thick solid red and thick dotted blue, respectively. In the example, $L_r + h_r = 3$, $S_r = \{r'\}$ and $t(r', r) = 1$.

4.6.4 Construction of $\bar{A}_{r,t}$

We also show how to construct a set $\bar{A}_{r,t}$ of arcs such that a path bans (r, t) if and only if it contains an arc in $\bar{A}_{r,t}$. Let $S_r = \{r' \in N_r \setminus r : TC_r \cap TC_{r'} \neq \emptyset\}$ be the set of all routes with track circuits in common with r. In a route-release interlocking system, a train bans (r, t) if and only if it occupies (r', t), where $r' \in S_r$. This is because the track circuits common to r and r' are locked when r' is locked, and not released until route r' is released. Hence, r is prevented from being locked for the same time intervals as r' is locked.

Using the same reasoning as in the last section, we can say that a train bans (r, t) if and only if its path contains a node in

$$\bar{W}_{r,t} = \{(r', t') : r' \in S_r, t' \in \{t, \ldots, t - t(r', r), \}$$

where $t(r', r) = L_{r'} + h_{r'}$. Therefore, a train bans (r, t) if and only if its path contains an arc in

$$\bar{A}_{r,t} = \bigcup_{(r',t') \in \bar{W}_{r,t}} \sigma^-(r', t') \setminus \bigcup_{(r',t') \in \bar{W}_{r,t}} \sigma^+(r', t').$$

The situation is different in a sectional-release system. Because track circuits are released one by one as r' is traversed, the track circuits $TC_r \cap TC_{r'}$ common to r and $r' \in S_r$ can be released before all of the track circuits in r' are released. This means that there may be time intervals t during which a train occupies (r', t), but does not ban (r, t).

To reflect this difference, we must amend the definition of $t(r', r)$. When route $r' \in S_r$ is first locked, route r is prevented from being locked. Route r becomes available again when all of the track circuits in $TC_r \cap TC_{r'}$ are released. The last track circuit of these to be released as the train traverses r' is $tc_{r'}^{i(r,r')-1}$ where

$$i(r, r') = \min \left\{ i \in \mathbb{N} : \{tc_{r'}^{j}\}_{j=i}^{n_{r'}-1} \cap TC_r = \emptyset \right\}.$$

We define $\theta(r, r') \in [0, 1]$ to be the proportion of the route traversal time $L_{r'}$ after which this occurs. The time to release can therefore be written

$$t(r', r) = \lceil \theta(r, r') L_r + h_r \rceil.$$

Here, $\lceil \cdot \rceil$ indicates rounding up to the nearest integer number of time intervals. Rounding up ensures that the discretisation of time does not lead to an underestimation of the time for which track circuits are locked. Figure 5 shows an example of both $\bar{W}_{r,t}$ and $\bar{A}_{r,t}$ in a sectional-release system.

4.6.5 Antichain Condition

The method of representing the track capacity constraints that is described in the preceding sections relies on an important assumption. This assumption is described below and sufficient conditions for its validity are given.

Suppose that there is a *source-sink* path in G that includes arcs from both $A_{r,t}$ and $\bar{A}_{r,t}$, for some time-space resource (r, t). Any train assigned this path would, by definition of $A_{r,t}$ and $\bar{A}_{r,t}$, both occupy and ban (r, t). This would violate Capacity Constraint 2, which ensures that (r, t) cannot be both occupied and banned. Capacity Constraint 2 applies regardless of whether the resource is occupied and banned by the same train, or different trains. It is not possible within the real signalling system for a train to violate the track capacity single-handedly. Therefore, for the model to be correct, there should be no *source-sink* paths in G that include arcs from both $A_{r,t}$ and $\bar{A}_{r,t}$. This is stated mathematically in Assumption 1, which relies on Definition 1.

Definition 1. *An antichain in a directed graph $G = (N, A)$ is a set $Z \subseteq A$ of arcs such that each path in G contains at most one arc in Z.*

Assumption 1. *For each $(r, t) \in N_0$, $A_{r,t} \cap \bar{A}_{r,t} = \emptyset$ and $A_{r,t} \cup \bar{A}_{r,t}$ is an antichain in G.*

Proposition 1, which uses uses Definition 2, provides sufficient conditions for Assumption 1 to hold. Checking these sufficient conditions is easier than directly checking that Assumption 1 holds.

Definition 2. *For two routes $r_1, r_2 \in N_r$, let*

$$L(r_1, r_2) = \min \left\{ \sum_{r' \in p \setminus r_2} L_{r'} : p \text{ is an } r_1\text{-}r_2 \text{ path in } G_r \right\}$$

be the minimum total traversal time from r_1 up to but not including r_2. For example, if $(r_1, r_2) \in A_r$ then $L(r_1, r_2) = L_{r_1}$.

Proposition 1. *Assumption 1 is true if for each $r \in N_r$, either (i) $S_r \cup \{r\}$ is an antichain in G_r or (ii) $L(r_1, r_2) > L_{r_1} + h_{r_1}$ for each pair $r_1, r_2 \in S_r \cup \{r\}$.*

Proof. Let $(r, t) \in N_0$. Since $r \notin S_r$ by definition, $W_{r,t} \cap \bar{W}_{r,t} = \emptyset$ and hence $A_{r,t} \cap \bar{A}_{r,t} = \emptyset$.

Suppose for contradiction that at least one of (i) and (ii) holds, and that there is a *source-sink* path p in G that contains two distinct arcs $((r_0, t_0), (r_1, t_1))$ and $((r_2, t_2), (r_3, t_3))$ in $A_{r,t} \cup \bar{A}_{r,t}$, where without loss of generality $t_1 \leq t_3$. By the definitions of $A_{r,t}$ and $\bar{A}_{r,t}$, the routes r_1 and r_3 are in $S_r \cup \{r\}$. Since $t_1 \leq t_3$, there must be a subpath q of p from (r_1, t_1) to (r_3, t_3). If $S_r \cup \{r\}$ is an antichain in G_r, then this is a contradiction. Therefore, (ii) must hold. Because, by definition, $A_{r,t}$ and $\bar{A}_{r,t}$ do not contain arcs starting in nodes of $W_{r,t}$ and $\bar{W}_{r,t}$, respectively, q must visit a node not in $W_{r,t}$ or $\bar{W}_{r,t}$ before visiting (r_3, t_3). Therefore, q must span at least $L(r_1, r_3)$ time intervals. Subpath q starts at time t_1 and ends at time t_3. By definition of $A_{r,t}$ and $\bar{A}_{r,t}$, $t_1 \geq t - L_{r_1} - h_{r_1}$ and $t_3 \leq t$. Putting this together, $t - L_{r_1} - h_{r_1} + L(r_1, r_3) \leq t$, which rearranges to $L(r_1, r_3) \leq L_{r_1} + h_{r_1}$. This violates (ii), which proves the result by contradiction. □

It is easier to check the conditions of Proposition 1 than Assumption 1 directly. This is because they depend only on the route graph G_r, rather than the full time-space graph G. In our instances, condition (i) is satisfied for all routes, and this will be the case for most railways. Figure 6 shows an example of how Assumption 1 could fail to hold if the conditions of Proposition 1 are not satisfied.

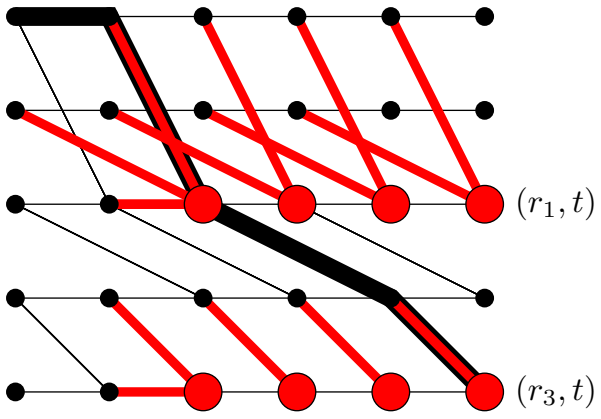


Figure 6: An example in which Assumption 1 does not hold. The arcs in $A_{r,t} \cup \bar{A}_{r,t}$ are shown as thick solid red lines, and $r_1, r_3 \in S_r \cup \{r\}$. The very thick black line is a path that contains two arcs in $A_{r,t} \cup \bar{A}_{r,t}$. Condition (i) doesn't hold, since $(r_1, r_3) \in A_r$. Condition (ii) doesn't hold because $L(r_1, r_2) = 2 \not> 3 = L_{r_1} + h_{r_1}$.

4.6.6 Innovations of Proposed Approach

In existing approaches, occupying is the only type of capacity that is considered. Typically, the track is modelled as consisting of components that can be used by at most one train at any one time. When these components are as large as block sections or routes,

the track capacity is underestimated by the model. This is convincingly explained and empirically demonstrated by both Pellegrini et al. [2014] and Corman et al. [2009].

In order to address this underestimation of track capacity, Pellegrini et al. [2014] and Corman et al. [2009] have explicitly modelled track circuits. These models have decision variables that describe the entry time of trains into each traversed track circuit. This approach is an accurate way of modelling the track capacity of a sectional-release interlocking area. However, there are two drawbacks to modelling all of the track circuits explicitly in this way. The first is that the large number of variables required to model interlocking areas with many track circuits can render models of this type intractable. The second drawback is that the solutions are difficult to implement in practice. This problem arises from the fact that track circuits are not individually signalled, and so the arrival of trains into individual track circuits cannot be controlled via signalling. If a solution specifies that a train must wait in a track circuit that does not immediately precede a signal, then a signaller cannot easily implement this.

Our model is able to accurately represent the track capacity of a sectional-release interlocking area without the drawbacks of explicitly modelling each track circuit. This is enabled by the consideration of banning as a second type of capacity in addition to occupying. Banning allows the release times of individual track circuits to be implicitly taken into account in the sense that although they could be calculated from a given solution, they are not associated with any decision variables. Using routes as the track components ensures that the model is tractable. This is because the size of the time-space graph depends on the number of routes in the area of track modelled rather than the number of track circuits, which is typically much greater. The use of routes rather than track circuits also ensures that solutions are clear and simple from an implementation perspective. The values of the decision variables in a solution indicate the time intervals in which trains should pass signals. This is something that signallers have full control over.

4.7 The Schedule

When railway traffic is perturbed due to primary delays, the schedule that was planned at the tactical phase may no longer be feasible. However, it is still the schedule that would ideally be carried out if possible. As a result, it is very important in determining the objective function.

Let $\mathcal{K}$ be the set of trains scheduled to pass though the controlled area during the time horizon. The schedule for each train $k \in \mathcal{K}$ can be represented by a sequence of triples $(r_j^k, a_j^k, d_j^k)_{j=0}^{J^k}$. In this notation, there are J^k *scheduled events* for train k within the area modelled during the time horizon. For event j, train k is scheduled to begin traversal of route $r_j^k \in N_r$ at time interval $a_j^k \in \mathcal{T}$, and depart route r_j^k at time interval $d_j^k \in \mathcal{T}$. Any values of a_j^k or d_j^k that lie outside the time horizon are ignored. Scheduled events can be *stopping events* or *passing events* depending on whether or not the train is scheduled to stop to allow passengers to embark or alight. Passing events have a scheduled arrival time but do not have a scheduled departure time.

The initial route and time interval of train k are given by r_0^k and a_0^k. If the train is within the area at the beginning of the time horizon, then $a_0^k = 0$, and r_0^k is the route

it is traversing at this time. Otherwise, r_0^k is the route on which train k first enters the area during the time horizon, and a_0^k is the time interval during which this occurs. In the latter case, a_0^k should be a forecast time. Using a forecast allows the model to take into account any anticipated delay in the arrival of the train into the controlled area. As a result, the model can suggest rescheduling solutions that pre-empt the effect of this anticipated delay, making rescheduling more effective.

Platform alternatives for the purpose of replatforming are inserted into the schedule. Suppose that (r_j^k, a_j^k, d_j^k) is a stop in the schedule, and that $\tilde{r}_j^k$ is an alternative platform to r_j^k of sufficient length to accommodate train k. Then $(\tilde{r}_j^k, a_j^k, d_j^k)$ is inserted into the schedule, in addition to (r_j^k, a_j^k, d_j^k). The set of all platform alternatives used for r_j^k must be an antichain in G_r. There is no constraint in the model on how many scheduled events each train carries out, but if the platform alternatives form an antichain in G_r then at most one of the platform alternative events can possibly occur.

4.8 The Objective Function

Most rescheduling models in the literature optimise a measure of performance that is inherent to the system. Though there are many examples (see [Samà et al., 2015]), the most prevalent objective function is to minimise a measure of train delays such as the maximum train delay. However, optimising a single inherent performance measure is inadequate because it fails to recognise that there may be many reasons for preferring one rescheduling solution over another. This is especially true for models that allow scheduled events to be skipped, such as the one in this paper. These models require an objective function that can express the priority of minimising delay relative to minimising skipped events. In recognition of the inadequacy of minimising a single inherent performance measure, Corman et al. [2012], Samà et al. [2015] and Binder et al. [2017] propose multicriteria methods for the TTRP. Whilst these methods take into account more than one performance measure, they are unsuitable for a real-time environment. One reason for this is that they do not achieve sufficiently fast computation times. In addition, it is not practical to require a railway signaller to choose between a set of solutions in real-time.

To overcome the challenges outlined above, the concept of utility has been used. Our objective is to maximise a utility function that models the rescheduling preferences of Network Rail. Accordingly, the objective function is the result of collaboration with Network Rail. The utility function is the total weighted utility over all trains k and scheduled events j, given by

$$U = \sum_{k \in \mathcal{K}} \alpha_k \sum_{j=1}^{J^k} \beta_k^j U_k^j(p^k).$$

In this formula, α_k is a weight reflecting the priority of train k, β_k^j is a weight reflecting the priority of event j for train k, and $U_k^j(p^k)$ is the utility accrued by train k at stop j when assigned *source-sink* path p^k in G.

Our use of total weighted utility over all train events is influenced by Network Rail's performance measure, *On-time at All Recorded Stations*. This measure is a summary of

the lateness of each train event. It was adopted in 2019 in preference to the *Public Performance Measure*, which only takes into account lateness at each train's final scheduled event. Considering delay only at the final event of each train is unsatisfactory because it results in objective functions that are indifferent to delays at intermediate stops, provided the time is subsequently recovered. It is also unsatisfactory to maximise the minimum utility over all train events, because this would correspond to indifference toward reductions in delay for all but the most delayed train. By modelling the utility as the total weighted utility over all trains and scheduled events, the objective function is never indifferent to a passenger delay reduction, and therefore reflects Network Rail's strategic priority of "putting passengers first" [Network Rail, 2019, p. 7].

The utility $U_k^j(p^k)$ accrued by train k at event j if it is assigned *source-sink* path p^k in G is defined as

$$U_k^j(p^k) = \begin{cases} 0 & \text{if path } p^k \text{ does not visit } r_j^k \\ \gamma(t - a_j^k) & \text{if path } p^k \text{ enters } r_j^k \text{ at time interval } t, \end{cases}$$

where $\gamma : \mathbb{Z} \to \mathbb{R}^+$, depending on the number $l = t - a_j^k$ of time intervals late the train arrives, is given by

$$\gamma(l) = \begin{cases} \phi^{-\omega|l|} & \text{if } |l| \leq \Gamma \\ 0 & \text{if } |l| > \Gamma. \end{cases}$$

Negative values of l correspond to a train arriving early. The function γ is sketched in Figure 7, with the parameter values used in our experiments ($\phi = 1.0000001$, $\omega = 150,000$ and $\Gamma = 240$, corresponding to 1 hour).

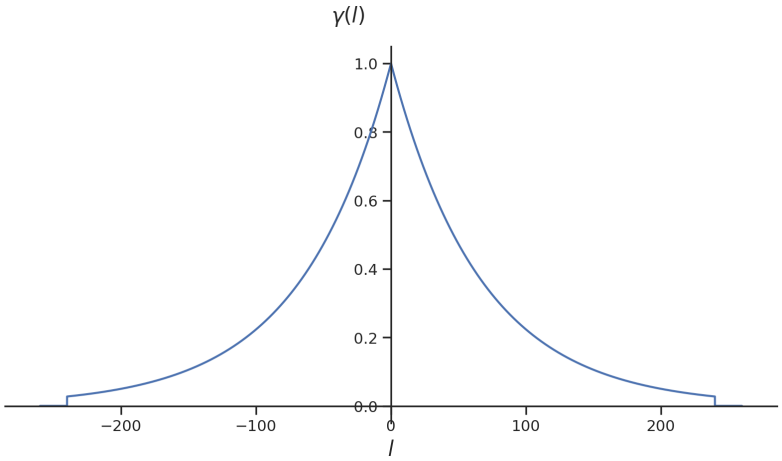


Figure 7: A sketch of γ.

The key features of this utility function are as follows:

- It depends on lateness, which is directly relevant to the passenger experience and therefore forms the basis of performance analysis within Network Rail. Using lateness is an improvement on the widely used measure of *consecutive delay*, defined as the additional delay created by rescheduling actions. Unlike lateness, consecutive delay is abstract to passengers.
- A preference is shown for events occurring as close to on-time as possible. This is achieved by making γ strictly increasing on $[-\Gamma, 0]$ and strictly decreasing on $[0, \Gamma]$.

This preference reflects the fact that late events are bad for passengers and early events can cause unanticipated difficulties at stations.

- A preference is shown for scheduled events occurring over not occuring, provided that they occur within Γ time intervals of the scheduled time. This is achieved by making $\gamma > 0$ on $[-\Gamma, \Gamma]$ and $U_k^j(p^k) = 0$ if the event does not occur. This reflects the fact that skipping scheduled stops is disruptive to passengers.
- The function γ is strictly convex on the intervals $[-\Gamma, 0]$ and $[0, \Gamma]$. This convexity reflects the idea that passengers are more sensitive to an additional time interval of lateness when a train is closer to on-time.
- Since the model is time-indexed, any set of values $\{\gamma(l) : l \in \mathbb{Z}\}$ could be used without compromising the tractability of the model. The objective function is therefore very flexible.

The value of α_k is set to 1 for express passenger (class 1) trains, and 0.4 for ordinary passenger (class 2) trains. The train weights α_k could also be used to take into account other factors affecting train priority such as the train's operator, the number of passengers or the number of remaining stops for the train once it has left the controlled area. However, these aspects are not considered in this paper.

The stop weights β_j^k are set as follows: A train's final arrival or passing event in the time horizon is given weight 0.7 to reflect the importance of trains leaving the controlled area punctually. All other arrivals excluding those representing platform alternatives receive an equal proportion of the remaining weight 0.3. Non-final passing events receive a weight of zero. When r_j^k is a platform alternative, β_j^k has the same weight as the planned platform arrival but discounted by a factor of 0.9. This reflects the disruption to passengers as a result of changing platforms. Other considerations could also be taken into account, such as the service patterns and passenger interchange importance of individual stations.

We will show in Section 4.9 how to set the arc weights in G to reflect the objective.

4.9 Arc Weights

The objective function is represented on the time-space graph G via the weights c_a^k for each train $k \in \mathcal{K}$ and each arc $a \in A$. These weights are set such that the total weight of a *source-sink* path for train k is equal to the negative of its total utility. Using the negative of the utility turns the problem into a minimisation problem. Note that we use the symbol ∞ to mean a real number sufficiently large that no good solution to the problem contains a path that uses an arc of this length — it is effectively a constraint.

The arcs in A_1 have weights $c^k_{(source,(r_0^j,a_0^j))} = \infty \mathbb{1}_{\{j \neq k\}}$ for each $k \in \mathcal{K}$, where $\mathbb{1}$ is the indicator function. This ensures that trains cannot begin in the initial position of another train. The cancellation arc in A_2, has $c^k_{(source,sink)} = 0$ for all trains k if cancellations are to be included in the model, and $c^k_{(source,sink)} = \infty$ otherwise. All of the arcs a in A_3, A_5 and A_6 have $c_a^k = 0$ for all k, because the sink is artificial and waiting does not contribute to the modelled utility function.

Arcs a in A_4 have $c_a^k = 0$ for all k except in two very important cases:

- **Departures**:

  For $k \in \mathcal{K}$, $j = 0, \ldots, J^k$ and $t < d_j^k$, if $a = ((r_j^k, t), (r', t')) \in A_4$ then $c_a^k = \infty$. This prevents trains departing from a stop early if they visit that stop.

- **Arrivals**:

  For $k \in \mathcal{K}$ and $j = 0, \ldots, J^k$, if $a = ((r', t'), (r_j^k, t)) \in A_4$ then $c_a^k = -\alpha_k \beta_k^j \gamma(t - a_j^k)$. Such an arc is included in a train path if and only if event j for train k occurs at time t. The negative of the associated utility is accrued.

Assumption 2 must hold, so that it is never possible for a train path to collect more than one positive reward for carrying out a scheduled event more than once.

Assumption 2. $\Gamma < L(r_j^k, r_j^k)$ *for all* $k \in \mathcal{K}, j = 0, \ldots, J^k$.

This is only a relevant consideration when short cycles containing platforms exist in G_r. There are no cycles containing platform routes in our instances, so we are free to pick any value of Γ.

5 Integer Programming Formulation

5.1 Arc Formulation

The TTRP, defined in Section 4.2, can be formulated as an Integer Linear Program similarly to the classical multicommodity flow problem. For each train $k \in \mathcal{K}$ and arc $a \in A$, we introduce a binary variable x_a^k which takes the value 1 if and only if the path in G corresponding to train k includes the arc a. The formulation is as follows:

$$\min \quad \sum_{k \in \mathcal{K}} \sum_{a \in A} c_a^k x_a^k \tag{1a}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{K}} \left(\sum_{a \in A_{r,t}} x_a^k + \delta \sum_{a \in \bar{A}_{r,t}} x_a^k \right) \leq 1 \quad \forall (r,t) \in N_0 \tag{1b}$$

$$\sum_{a \in \sigma^+(n)} x_a^k - \sum_{a \in \sigma^-(n)} x_a^k = b_n^k \quad \forall k \in \mathcal{K}, n \in N \tag{1c}$$

$$x_a^k \in \{0, 1\} \quad \forall k \in \mathcal{K}, a \in A. \tag{1d}$$

where $b_{source}^k = 1$, $b_{sink}^k = -1$ for all $k \in \mathcal{K}$, $b_n^k = 0$ for all $n \in N_0$ and $k \in \mathcal{K}$, and $\delta > 0$ is a small positive constant. The notation $\sigma^+(n)$ and $\sigma^-(n)$ denotes the set of arcs starting and ending at node n, respectively.

The objective (1a) minimises the total weight of the paths. The constraints have a block-diagonal structure: (1b) are linking constraints enforcing the track capacity, whilst the flow constraints (1c) form one block per train. For each train $k \in \mathcal{K}$, the corresponding block of flow constraints ensures that the solution for train k corresponds to a *source-sink* path in G.

The track capacity constraints are expressed by (1b). If a train k occupies a resource $(r,t) \in N_0$, then $x_a^k = 1$ for some $a \in A_{r,t}$ and the constraint ensures that (r,t) cannot otherwise be occupied or banned. If no train occupies $(r,t) \in N_0$, then up to $\lfloor \frac{1}{\delta} \rfloor$ trains may ban it. We make δ sufficiently small that this limit is never exceeded in practice. In our experiments, $\delta = 0.05$ was sufficient.

This formulation has $|\mathcal{K}||A|$ binary variables and $|N_0| + |\mathcal{K}||N|$ constraints. A typical instance in our computational experiments has around 3,500,000 binary variables and around 1,500,000 constraints. We could not even obtain feasible solutions using the generic Mixed Integer Programming solver Gurobi because the instances were too large.

5.2 Danzig-Wolfe Decomposition into a Path Formulation

Decomposition is a popular approach for integer programs that are prohibitively large to solve using general purpose solvers. Formulation (1a)–(1d) is particularly amenable to Danzig-Wolfe decomposition because the constraint matrix has a block-diagonal structure where each block is the constraint matrix of a shortest path problem on a directed acyclic graph.

We perform Danzig-Wolfe decomposition using the discretisation approach presented by Vanderbeck and Wolsey [2010]. For each train $k \in \mathcal{K}$, define

$$X^k = \left\{ \mathbf{x}^k \in \{0,1\}^{|A|} : \sum_{a \in \sigma^+(n)} x_a^k - \sum_{a \in \sigma^-(n)} x_a^k = b_n^k \quad \forall n \in N \right\}$$

to be the set of feasible paths, and P^k to be its finite index set so that $X^k = \{\mathbf{x}^{k,p} : p \in P^k\}$. We can rewrite variables $\mathbf{x}^k$ in terms of the constants $\mathbf{x}^{k,p}$ and new binary variables $\lambda^{k,p}$:

$$\mathbf{x}^k = \sum_{p \in P^k} \lambda^{k,p} \mathbf{x}^{k,p}, \quad \sum_{p \in P^k} \lambda^{k,p} = 1, \quad \lambda^{k,p} \in \{0,1\} \quad \forall p \in P^k.$$

This variable transformation makes the constraint $\mathbf{x} \in X^k$ implicit, so when we substitute the new variables into formulation (1a)–(1d), constraint (1c) can be dropped. This results in the path formulation:

$$\min \quad \sum_{k \in \mathcal{K}} \sum_{p \in P^k} \left(\sum_{a \in A} c_a^k x_a^{k,p} \right) \lambda^{k,p} \tag{2a}$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{K}} \sum_{p \in P^k} \left(\sum_{a \in A_{r,t}} x_a^{k,p} + \delta \sum_{a \in \bar{A}_{r,t}} x_a^{k,p} \right) \lambda^{k,p} \leq 1 \quad \forall (r,t) \in N_0 \tag{2b}$$

$$\sum_{p \in P^k} \lambda^{k,p} = 1 \quad \forall k \in \mathcal{K} \tag{2c}$$

$$\lambda^{k,p} \in \{0,1\} \quad \forall k \in \mathcal{K}, p \in P^k. \tag{2d}$$

Formulations (1a)–(1d) and (2a)–(2d) are equivalent. Applying the classical results of Geoffrion [1974], the linear relaxations of these formulations provide the same lower bound on their optimal values, which is the same bound provided by Lagrangian relaxation of (1a)–(1d). This is because all of the extreme points of the linear relaxation of each X^k are integral.

Formulation (2a)–(2d) has $|\mathcal{K}| + |N_0|$ constraints, which can be significantly fewer than (1a)–(1d). Although formulation (2a)–(2d) typically has many more variables, we know that each integer feasible solution will have exactly $|\mathcal{K}|$ non-zero variables. As a result, (2a)–(2d) can be effectively solved using a branch-and-price algorithm.

6 Branch-and-Price Algorithm

Formulation (2a)–(2d) can be solved using a branch-and-price algorithm. This is a branch-and-bound algorithm in which the LP relaxation at each node is solved by column generation. In Sections 6.1 and 6.2 we describe how column generation can be used to solve the LP relaxation of (2a)–(2d). Two acceleration strategies are described in Sections 6.3 and 6.4. Finally, in Section 6.5, we describe our branching scheme.

6.1 Column Generation

Here we outline the basic column generation algorithm (see Desrosiers and Lübbecke [2005] for a more general explanation) for solving the LP relaxation of (2a)–(2d), which is referred to as the Master Problem (MP).

In iteration n, we restrict P^k to a much smaller subset P^k_n, resulting in a smaller LP called the Restricted Master Problem (RMP). Solving the RMP yields an optimal primal solution $\boldsymbol{\lambda}^* = (\lambda^{*k,p})_{k\in\mathcal{K},p\in P^k_n}$ and optimal dual values $\boldsymbol{\rho}^* = (\rho^*_k)_{k\in\mathcal{K}}$ corresponding to constraints (2c), and $\boldsymbol{\pi}^* = (\pi^*_{r,t})_{(r,t)\in N_0}$ corresponding to constraints (2b).

These dual values can be used to find a column $p \in P^k \setminus P^k_n$ of minimum reduced cost for each $k \in \mathcal{K}$. If none of these columns have negative reduced cost, then $\boldsymbol{\lambda}^*$ is optimal for the MP, which is now solved. Otherwise, at least one column with negative reduced cost must be added to the RMP (i.e. $P^k_n \subset P^k_{n+1}$) and the algorithm proceeds to iteration $n+1$.

The reduced cost of a column corresponding to path $p \in P^k$ for train k is

$$\begin{aligned}
\bar{c}^{k,p} &= \sum_{a\in A} c^k_a x^{k,p}_a - \sum_{(r,t)\in N_0} \left(\sum_{a\in A_{r,t}} x^{k,p}_a + \delta \sum_{a\in \bar{A}_{r,t}} x^{k,p}_a \right) \pi^*_{r,t} - \rho^*_k \\
&= \sum_{a\in A} c^k_a x^{k,p}_a - \sum_{(r,t)\in N_0} \sum_{a\in A} \left(\mathbb{1}_{\{a\in A_{r,t}\}} + \delta \mathbb{1}_{\{a\in \bar{A}_{r,t}\}} \right) x^{k,p}_a \pi^*_{r,t} - \rho^*_k \qquad (3) \\
&= \sum_{a\in A} (c^k_a - \pi^*_a) x^{k,p}_a - \rho^*_k
\end{aligned}$$

where

$$\pi^*_a = \sum_{(r,t)\in N_0} \left(\mathbb{1}_{\{a\in A_{r,t}\}} + \delta \mathbb{1}_{\{a\in \bar{A}_{r,t}\}} \right) \pi^*_{r,t} \qquad (4)$$

is the dual value for arc a, and $\mathbb{1}$ is the indicator function.

The task of finding the minimum reduced cost column for train k, known as the k^{th} subproblem (SP^{k}), therefore amounts to finding a shortest *source-sink* path in G where each weight c_a^k is replaced by $c_a^k - \pi_a^*$.

6.2 Subproblem Solution

Solving the subproblem for each train amounts to finding a *source-sink* path in G with modified weights. Because G is directed and acyclic, we can use the classical algorithm of Kahn [1962]. This is an exact labelling algorithm that first sorts the nodes $n \in N$ into a topological order $N_{top} = (n_0, \ldots, n_{|N|})$ so that $(n_i, n_j) \in A \iff i < j$. The node labels are initialised to ∞ and the nodes are scanned once in topological order starting at the node *source*. At each node $n_i \in N$, the directed arcs starting at n_i are relaxed in turn. An arc $(n_i, n_j) \in A$ is relaxed by checking whether the sum of the label at n_i and the weight of (n_i, n_j) is smaller than the label at node n_j. If so, the label of node n_j is updated to this smaller value, and n_i is recorded as the predecessor of n_j. When the node *sink* is reached in the scan of N_{top}, the algorithm backtracks through the succession of recorded predecessors of the node *sink* to yield the shortest *source-sink* path in G. This algorithm is of complexity $\mathcal{O}(|A|)$.

Note that although the subproblem is solved multiple times in the column generation algorithm, the topological ordering N_{top} needs to be calculated only once at the beginning of the entire solve process. This ordering remains valid for every subproblem.

6.3 Acceleration Strategy 1: Partial Pricing

To improve the convergence of the column generation algorithm, we use partial pricing (see Desaulniers et al. [2002]). This popular technique involves solving only a subset of the subproblems in each pricing round. If no negative reduced cost columns are found from this subset then a full pricing round is used to determine whether the MP is solved. This causes the dual values to converge more quickly and hence reduces the total number of subproblems that must be solved, at the expense of potentially solving more RMPs. Empirical evidence suggests that solving subproblems in three equally sized groups achieves the best performance.

6.4 Acceleration Strategy 2: Reduced Cost Variable Fixing

Reduced cost variable fixing (see Irnich et al. [2010]) has also been used to accelerate the column generation algorithm. Reduced cost variable fixing is based on the observation that if the reduced cost $\bar{c}_a^k$ of an original variable x_a^k exceeds the integrality gap $UB - LB$, then arc a cannot appear in an optimal path p^k for train k. LB and UB can be any lower or upper bounds on the optimal objective value, respectively. At each execution of the reduced cost fixing procedure, UB is set to the objective of the current incumbent primal feasible solution. At each execution, LB is set to the sum of the RMP objective value

and $|\mathcal{K}| \min\{rc^k : k \in \mathcal{K}\}$, where rc^k is the reduced cost for subproblem k (see [Desrosiers and Lübbecke, 2005, p. 11]).

Any arc a that cannot appear in an optimal path p^k for train k can be eliminated from subproblem k. In practice, this is achieved by setting its weight c_a^k to ∞, where ∞ is a number sufficiently large to prevent arc a ever forming part of an optimal path p^k for train k. Eliminating these arcs prevents the generation of columns that cannot form part of an optimal integer solution. This causes the column generation algorithm to discover good quality integer feasible solutions more quickly.

Directly translating the results from Irnich et al. [2010], the reduced cost $\bar{c}_a^k$ of variable x_a^k, where $a = (i, j)$, is calculated from an optimal dual solution $(\boldsymbol{\pi}^*, \boldsymbol{\rho}^*)$ as

$$\bar{c}_a^k = c_a^k - \pi_a^* + l_i - l_j$$

where l_i is the length of the shortest *source-i* path in G when the arc weights are modified to $(c_a^k - \pi_a^*)$. These shortest path distances are available from the graph labels after solving the subproblem and can be extracted with little computational effort. Reduced cost variable fixing in a given subproblem is therefore of complexity $\mathcal{O}(|A|)$.

Our computational experiments show that reduced cost variable fixing is only effective when the integrality gap is small, for example under 5%. Therefore, the procedure is not executed until the root node is solved, and after that it is executed each time a new best primal solution is found.

6.5 Branching

An optimal solution to the linear relaxation of (2a)–(2d) may contain fractional variables and therefore not be integer feasible for (2a)–(2d). The column generation algorithm is therefore embedded inside a branch-and-bound algorithm, resulting in a branch-and-price algorithm.

The following result characterises the possible fractionalities by showing that a fractional variable always arises as a result of a track capacity conflict between two different trains.

Definition 3. *A solution $\boldsymbol{\lambda}$ to an RMP has a fraction-inducing conflict if there exists an active constraint $(r, t) \in N_0$ in the RMP containing both a fractional variable $\lambda^{k_0,p_0} \in (0, 1)$ and a non-zero variable $\lambda^{k_1,p_1} \in (0, 1]$ with $k_0 \neq k_1$. We denote the fraction-inducing conflict $(r, t, \lambda^{k_0,p_0}, \lambda^{k_1,p_1})$.*

Proposition 2. *If $\boldsymbol{\lambda}$ is a fractional, basic solution of the RMP that is optimal for the MP, then $\boldsymbol{\lambda}$ has a fraction-inducing conflict.*

Moreover, if a fractional variable has minimal objective coefficient among all basic variables for its train, and these objective coefficients are not all equal, then it is part of a fraction-inducing conflict.

Proof. Suppose $\boldsymbol{\lambda}$ is fractional. Then $\exists k \in \mathcal{K}$ and $p \in P^k$ such that $\lambda^{k,p} \in (0, 1)$.

Letting $\{\lambda^{k,\hat{p}} : \hat{p} \in \hat{P}^k\}$ be the set of variables for train k taking fractional values, we know by constraints (2c) that $|\hat{P}^k| \geq 2$.

For each $\hat{p} \in \hat{P}^k$, let $\boldsymbol{\lambda}_{\hat{p}}$ be the solution obtained from $\boldsymbol{\lambda}$ by replacing

$$\lambda^{k,\hat{p}} \leftarrow \lambda^{k,\hat{p}} + \epsilon \text{ and } \lambda^{k,q} \leftarrow \lambda^{k,q} - \frac{\epsilon}{|\hat{P}^k| - 1}$$

for all other $q \in \hat{P}^k \setminus \{\hat{p}\}$, where $\epsilon > 0$ is a small constant.

By assumption, $\boldsymbol{\lambda}$ is a basic solution and hence an extreme point of the feasible set of the RMP. However, it can also be written as a sum

$$\boldsymbol{\lambda} = \frac{1}{|\hat{P}^k|} \sum_{\hat{p} \in \hat{P}^k} \boldsymbol{\lambda}_{\hat{p}}$$

of the distinct solutions $\boldsymbol{\lambda}_{\hat{p}}$. Therefore, $\exists p_0 \in \hat{P}^k$ for which $\boldsymbol{\lambda}_{p_0}$ is infeasible.

By construction, $\boldsymbol{\lambda}_{p_0}$ satisfies (2c) and so $\boldsymbol{\lambda}_{p_0}$ must violate (2b). In other words, there is a capacity constraint $(r,t) \in N_0$ that is active for $\boldsymbol{\lambda}$ and violated by $\boldsymbol{\lambda}_{p_0}$. Constraint (r,t) would not be violated by $\boldsymbol{\lambda}_{p_0}$ if it contained only variables in $\hat{P}^k$, so it must contain another variable $\lambda^{k_1,p_1} \in (0,1]$ with $k_1 \neq k_0$, which takes the same value in $\boldsymbol{\lambda}$.

Note that if $c^{k,p} = \min\{c^{k,\hat{p}} : \hat{p} \in \hat{P}^k\}$ and $\exists q \in \hat{P}^k$ such that $c^{k,p} < c^{k,q}$, then $\boldsymbol{\lambda}_p$ as constructed above is infeasible by the optimality of $\boldsymbol{\lambda}$, and so we can set $p_0 = p$.

□

One consequence of this result is that instances with no conflicts, for example instances in which there is no primary delay, produce integral LP relaxations. More generally, we would expect the number of fractional variables to be correlated with the number of conflicts in the instance.

To devise a branching rule, we must first understand the different types of fraction-inducing conflicts that can arise:

Definition 4. *Given a fraction-inducing conflict $(r,t,\lambda^{k_0,p_0},\lambda^{k_1,p_1})$ for $\boldsymbol{\lambda}$, let $y_{r,t}^{k_0,p_0}$ and $y_{r,t}^{k_1,p_1}$ be the coefficients of λ^{k_0,p_0} and λ^{k_1,p_1} in constraint (r,t). These are equal to 1 if the path corresponding to the variable occupies (r,t), or δ if it bans (r,t). Then the fraction-inducing conflict is, respectively, of type 'OO', 'OB', 'BO' or 'BB' if the pair $(y_{r,t}^{k_0,p_0}, y_{r,t}^{k_1,p_1})$ is equal to $(1,1),(1,\delta),(\delta,1),(\delta,\delta)$.*

Proposition 3. *Under the conditions of Proposition 2, $\boldsymbol{\lambda}$ has a fraction-inducing conflict of type 'OO' or 'OB'.*

Proof. By Proposition 2, $\boldsymbol{\lambda}$ has a fraction-inducing conflict $(r,t,\lambda^{k_0,p_0},\lambda^{k_1,p_1})$. It must be of one of the four types, 'OO', 'OB', 'BO' or 'BB'. If it is of type 'OO' or 'OB', then the result is immediately true.

Suppose that $(r,t,\lambda^{k_0,p_0},\lambda^{k_1,p_1})$ of type 'BO'. Since $\delta\lambda^{k_0,p_0} > 0$, we know from constraint (r,t) that $\lambda^{k_1,p_1} < 1$. Consequently, $(r,t,\lambda^{k_1,p_1},\lambda^{k_0,p_0})$ is a fractional-inducing conflict of type 'OB'.

Finally, suppose that $(r,t,\lambda^{k_0,p_0},\lambda^{k_1,p_1})$ is of type 'BB'. Since constraint (r,t) is active, and δ was chosen sufficiently small that in practice fewer than $\lceil \frac{1}{\delta} \rceil$ trains can ban it, constraint (r,t) must contain a variable $\lambda^{k_2,p_2} > 0$ with $y_{r,t}^{k_2,p_2} = 1$. Therefore, $(r,t,\lambda^{k_0,p_0},\lambda^{k_2,p_2})$ is a

fractional-inducing conflict of type 'BO'. Using the argument in the preceding paragraph, $(r, t, \lambda^{k_2,p_2}, \lambda^{k_0,p_0})$ is of type 'OB'.

□

Proposition 3 establishes that the following branching is always available:

Branching Rule 1.

$$\begin{array}{llllll} (L) & k_0 \textit{ cannot occupy } (r,t) & (R) & k_1 \textit{ cannot occupy } (r,t) & \textit{if type 'OO'} \\ (L) & k_0 \textit{ cannot occupy } (r,t) & (R) & k_0 \textit{ cannot ban } (r,t) & \textit{if type 'OB'} \end{array}$$

Branching Rule 1 breaks the conflict by preventing k_0 and k_1 from occupying or banning (r, t) in the left and right hand branches, respectively, according to the type of conflict. This leaves (r, t) free for the other train in each branch.

This branching is *compatible* with the subproblem, meaning that the additional constraint created by each branch can be enforced in the subproblem without compromising the efficiency of its solution algorithm. To prevent a train k from occupying a resource (r, t), we set the arc weights $c_a^{k_0} = \infty$ for each $a \in A_{r,t}$, and to prevent banning we similarly set $c_a^{k_0} = \infty$ for each $a \in \bar{A}_{r,t}$. The subproblems remain shortest path problems on a directed acyclic graph.

A practical algorithm for identifying a fraction-inducing conflict is given in Algorithm 1. We choose to sort the variables by $c^{k,p}$ in ascending order for two reasons. First, variables with low objective value are more likely to have the lowest objective value among all basic variables for their train and hence, by Proposition 2, yield a fraction-inducing conflict. Secondly, variables with low objective value are likely to be more important for establishing good primal and dual bounds, because they represent the most desirable paths.

Algorithm 1 Conflict Branching

```
for (k_0, p_0) ∈ {(k,p) : λ^{k,p} ∈ (0,1)} sorted by c^{k,p} ascending do
    for (r,t) ∈ N_0 such that λ^{k_0,p_0} in constraint (r,t) do
        for λ^{k_1,p_1} in constraint (r,t) do
            if k_1 ≠ k_0 and λ^{k_1,p_1} ∈ (0,1) then
                return (r, t, k_0, k_1)
```

7 Instance Data

We have developed a new set of 310 instances for the TTRP to test our approach. These instances are based on real data from an area of railway around Doncaster station in the UK. It is of critical importance that instances used to test rescheduling algorithms are realistic, because the difficulty can depend heavily on the instance. For our model, Proposition 2 shows that there is a direct link between the amount of track capacity conflict in a given instance and the strength of the formulation for this instance. Important aspects of the instance generation are described below.

7.1 Track Data

We have chosen to use an area around Doncaster station for our instances. Doncaster station lies on the East Coast Main Line, a busy railway corridor connecting London with Leeds, York, Newcastle and Edinburgh. It is an important interchange for inter-city services and terminus for local services, with 3,857,000 passenger entries and exits in 2017–2018 and over 30 trains per hour at peak times. Doncaster is a bottleneck which is responsible for reactionary delay to trains on the main line. This makes it ideal for testing our method.

Doncaster station, shown in Figure 8, has 9 platforms and 85 track circuits. It is bounded by 17 signals, and has 17 signals inside the station. In addition to the mainline that runs through the station, 4 double track lines start at Doncaster, going to Sheffield, Lincoln, Leeds and Hull, respectively. There are several heavily used routes within the station which cross the main lines and create conflicts. A good example is the route into platform 1, going from berth 0302 to berth 0278 (see Figure 8). The wider area covered is shown in Figure 9. It contains portions of each of the lines mentioned, but lies within a single area of signalling control. It consists of 225 berths with 313 valid berth transitions.

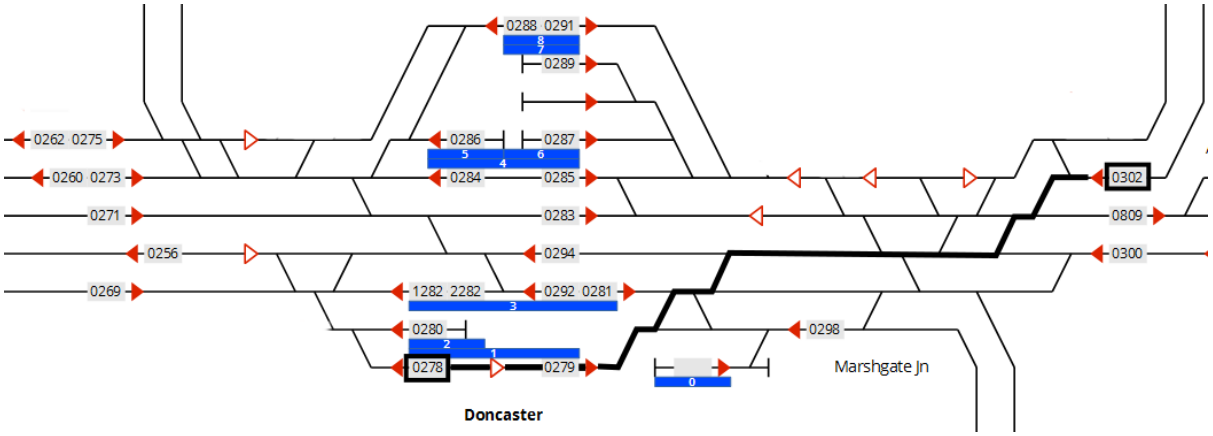


Figure 8: A berth diagram of Doncaster Station. The route from berth 0302 to berth 0278 is highlighted. Retrieved from `https://wiki.openraildata.com/index.php?title=DR`. Accessed 14/04/20.

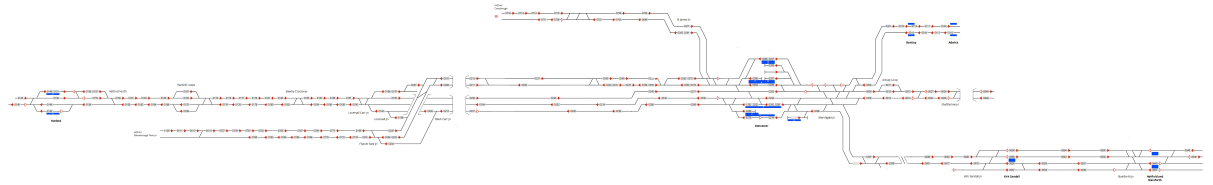


Figure 9: A berth diagram of the area of track modelled. Retrieved from `https://wiki.openraildata.com/index.php?title=DR`. Accessed 14/04/20.

In order to infer the track data, we used 18 months of historical data from a Train Describer (TD), one of Network Rail's real-time information systems. Train Describers record, directly from the track circuits, the time at which every berth transition occurs within a particular area of signalling control. From this data, we were able to deduce the set $\mathcal{S}$ of berths within the area, and set $\mathcal{A}$ of permitted berth transitions. A significant

amount of data cleaning was required to remove incorrectly recorded berth transitions, and berth transitions that are allowed only in exceptional circumstances. From $\mathcal{S}$ and $\mathcal{A}$, we were able to build the nodes and arcs of G_b and hence G_r.

Data about the track circuits TC_r included in each route $r \in N_r$ were not available digitally, so we recorded them manually from Network Rail drawings. The minimum dwell time D_{pl} was taken from timetable planning rules to be 120 seconds for all platforms. The headway times h_r and H_{pl} were estimated at 30 seconds for every route and platform, respectively.

7.2 Traversal Time Estimation

For each route $r \in N_r$, we estimated the traversal time $L_r \in \mathbb{Z}^+$ from the historical observations $\mathbf{y}^r$ in the TD data. These values play a crucial role in the track capacity constraints, and hence in determining how difficult the instances are to solve. By estimating them from the historical data, we were able to overcome any need for data about distances, speed limits and rolling stock characteristics.

We first estimate the time $l_r \in \mathbb{R}^+$ in seconds, and then calculate L_r by rounding l_r up to the nearest whole number of time intervals. Although this introduces approximation error depending on the size of the time interval, rounding up ensures that solutions to the model are not unachievable in practice.

For some routes, the observed times in $\mathbf{y}^r$ can be seen to arise from two different processes: the transition times of trains travelling close to the line speed, and the transition times of trains that were not close to the line speed. This second process consists of trains which were slowed or even stopped altogether by a signal, and trains which stopped at platforms during their traversal of the route. Since we do not know which observations are from which process, we fit a Gaussian mixture model with two components to cluster the observations into these two processes. We fit the Gaussian mixture model with the EM algorithm, and take l_r to be the minimum of the means of the components. See Bouveyron et al. [2019] for an introduction to Gaussian mixture models and the EM algorithm.

Many routes, for example those on the open line, appear to have only one mixture component. To decide whether to use one component or two, we fit both models and choose the model that optimises the Bayesian Information Criterion [Bouveyron et al., 2019, p .51]. When there is just one component, we take l_r to be the mean of this component.

7.3 Timetable Data

There are 310 instances in the test set. Each one covers a different hour long period starting on the hour between 8am - 6pm during January 2017 (10 instances per day over 31 days). They were created using the actual timetables and actual traffic perturbations during these times. We consider the use of real traffic perturbation scenarios to be crucial in assessing the algorithm fairly, despite the fact that it is not universal practice in the published literature.

The data was collected from TRUST (Train Running Under System TOPS), another information system used by Network Rail. This system is used to compare the timetable with actual performance and is widely used within the organisation for performance analysis. Location data in the timetable, and therefore TRUST, is recorded at the level of stations and junctions (called *timing points*), with platforms and directions specified where appropriate. Directions are recorded as 'up' or 'down', and correspond to the two directions in which the most important railway line passing through a given station can be traversed.

By creating a mapping from each combination of timing point, direction and platform to the corresponding berth, we were able to reconcile our track data with the TRUST data. This mapping was created automatically by linking trains in the Train Describer (see Section 7.1) and TRUST datasets. This allowed us to infer the data $(r_j^k, a_j^k, d_j^k)_{j=1}^{J^k}$ for each train k, as described in Section 4.7.

Because the instances are drawn from real examples, the level of perturbation varies considerably between them. However, the set of instances as a whole is an unbiased sample of the daily and weekly fluctuations in traffic, and the variations in primary delay over that month.

Figure 10 shows some relevant characteristics of the test set. Figure 10(a) shows the distribution of the number of trains over the test set. The majority of instances have between 20 and 32 trains, with the most common value being 29. The 4 instances with 3 trains or fewer are taken from data corresponding to public holidays. Figure 10(b) shows the distribution of the number of *conflicts* in each instance over the testset. This is calculated as the number of capacity constraints which are violated by the solution obtained by solving the problem without track capacity constraints. Finally, Figure 10(c) concerns the number of *train pair conflicts*. This is defined as the number of distinct pairs of trains between which there is at least one conflict. The number of conflicts is higher than the number of train pair conflicts, because one conflicting pair of trains usually causes several conflicts. It is common for these conflicts to occur on a single route over several consecutive time intervals.

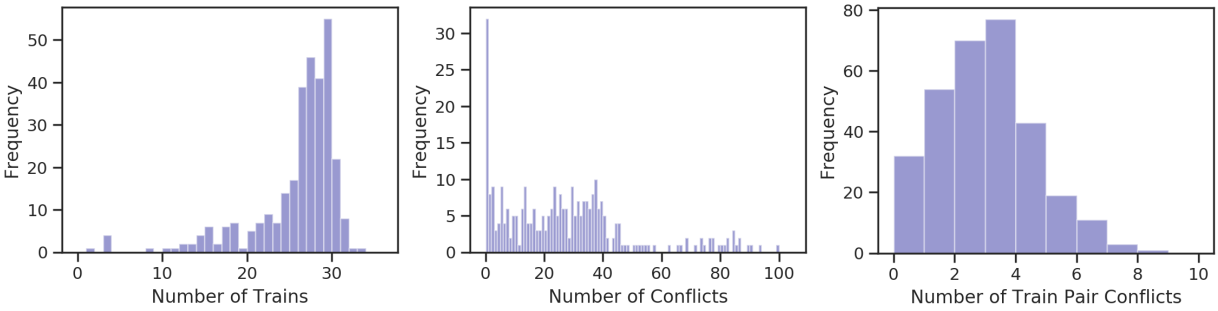


Figure 10: Histograms showing the test set distribution of the number of (a) trains, (b) conflicts and (c) train pair conflicts.

8 Computational Results

A computational study has been carried out to evaluate the performance of our branch-and-price algorithm on our new instances. In Section 8.1 we investigate the run time required to solve the instances to optimality. The effects of the acceleration strategies, the relationship between conflicts and algorithm performance, and the number of reroutings in the solutions are also explored. In Section 8.2, we present the results of imposing a 20 second time limit. These results evaluate the suitability of the developed algorithm for a real-time environment.

For the purpose of the analysis, we have omitted the results of the 32 instances in which there were no conflicts. By Proposition 2, the LP relaxation of an instance with no conflicts cannot have a fractional solution. As a result, all such instances were solved to optimality in less than 3 seconds, without the need for branching. Many of these instances do include small traffic perturbations, but they are not severe enough to cause conflicts.

The algorithm is implemented using SCIP 6.0.2 [Gleixner et al., 2018] as a branch-and-price framework with custom plugins written in the C language using Gurobi 9.0 [Gurobi Optimization LLC, 2020] as the linear programming solver. The experiments were carried out on a computing node equipped with an 18 core Intel Xeon E5-2699 v3 CPU with 2.30GHz and 500GB of RAM, running Ubuntu 16.04.

The full tables of results are available online (see [Reynolds, 2020]).

8.1 Solving to Optimality

First, we solved each instance with a time limit of 600 seconds to evaluate the time required to solve the instances to optimality. Figure 11 shows that across all 278 instances with at least one conflict, the median time to solve to optimality using no acceleration strategies was 12.55 seconds. It also shows that 90% of instances were solved within 45.45 seconds, and 4 instances were not solved to optimality in 600 seconds. Using acceleration strategies, we were able to reduce the median time to 10.84 seconds, the 90^{th} percentile to 31.55 seconds and solve all but 2 instances. These results demonstrate that whilst most instances can be solved in very reasonable times, it is not realistic to solve every instance to optimality in a real-time setting.

Considering the instances that were solved using acceleration methods, we find that the number of branch-and-bound nodes is highly correlated with the time to solve to optimality, with a Pearson correlation coefficient of 0.87, significant with p-value < 0.001. A mean of 88 variables per instance were generated throughout the column generation algorithm, demonstrating fast convergence. Although all of the instances had 64,082 constraints, solving LPs took only a small proportion of the total solve time: on average 74% of the total solving time was spent pricing variables.

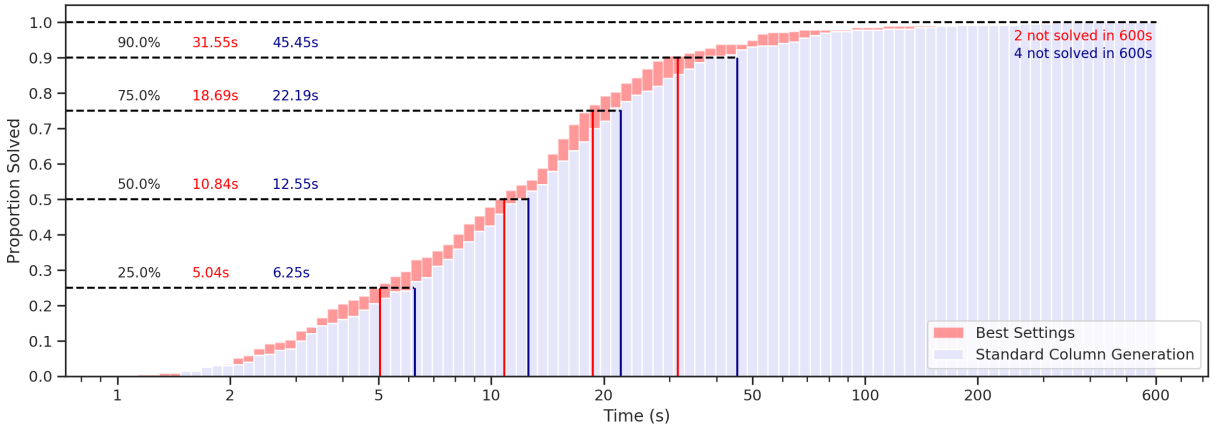
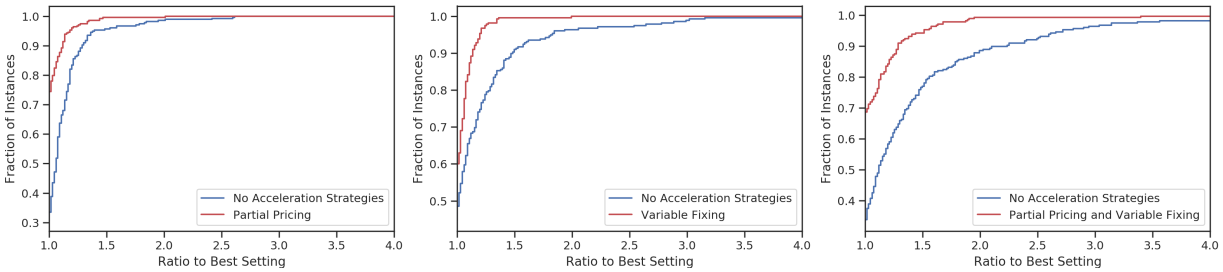


Figure 11: A cumulative histogram showing the proportion of instances solved for each number of seconds. Note that a log scale is used on the x-axis.

8.1.1 Acceleration Methods

The effects of both acceleration methods were examined separately. The performance profile in Figure 12(a) shows that partial pricing was effective at reducing solution times, particularly by producing modest improvements across many of the instances. Partial pricing speeds up the convergence of column generation, so it has the potential to be effective on all instances. However, column generation convergence was less important than the number of branch-and-bound nodes in determining overall solution time, so its effect is limited.

Figure 12(b) shows that reduced cost variable fixing was also effective at reducing solution times. The performance profile reveals that it produced gains in a smaller number of instances. This might be due to the fact that it works by reducing the number of branch-and-bound nodes required to reach optimality, and therefore there was limited scope to be effective on instances with relatively few nodes. Figure 12(c) shows that partial pricing and variable fixing were most effective when used together. The remainder of the results analysis refers to results obtained by using both of the acceleration strategies.

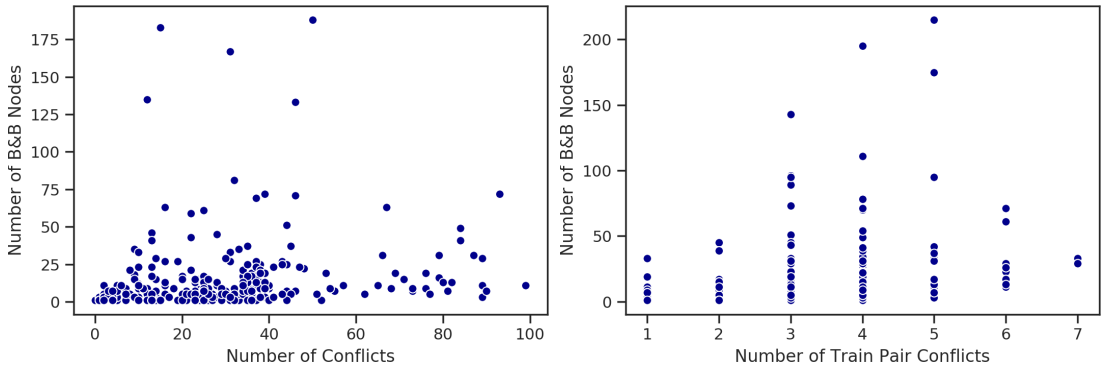


(a) (b) (c)

Figure 12: Performance profiles comparing standard column generation without any modifications with (a) partial pricing (b) reduced cost variable fixing and (c) both.

8.1.2 The Effect of Conflicts

In Section 6.5, we conjectured that there is a positive relationship between the number of branch-and-bound nodes and the level of conflict in an instance. The relationship between the number of conflicts and the number of branch-and-bound nodes is plotted in Figure 13(a). This plot is repeated with the number of train pair conflicts in Figure 13(b).



(a) (b)

Figure 13: Scatter plots showing the relationship between conflicts and the number of branch-and-bound nodes. Note that 3 instances with more than 250 branch-and-bound nodes have been omitted from both plots for ease of understanding.

The number of branch-and-bound nodes has a moderate positive relationship with both the number of conflicts, and the number of train pair conflicts. Because the quantities are discrete, we measure this relationship with the Spearman correlation coefficient which looks at the correlation between the ranks of the values. The Spearman correlation coefficients are 0.38 and 0.68 respectively, both with p-value < 0.001. One possible explanation for the higher correlation with the number of train pair conflicts is that branching often resolves multiple conflicts at the same time. This is especially likely to happen when conflicts occur for the same route in consecutive time intervals.

Despite this apparent relationship, a significant amount of the variation in the number of branch-and-bound nodes is unexplained by the number of conflicts. It is likely that this is accounted for by the variation in difficulty of resolving each conflict. Understanding the relationship between the difficulty of resolving conflicts and the instance difficulty would be an interesting topic for further research.

8.1.3 Rerouting

As part of the validation of the model, we investigated how many times a train was rerouted in each instance. Figure 14 shows that at least one rerouting was carried out in 48% of instances, with the most common number of reroutings being 1. Greater numbers of reroutings were less frequent, with a maximum of 5 carried out in a single instance.

The presence of rerouting decisions in the solutions justifies the inclusion of rerouting in the model. A model without the possibility of rerouting would not be able to produce

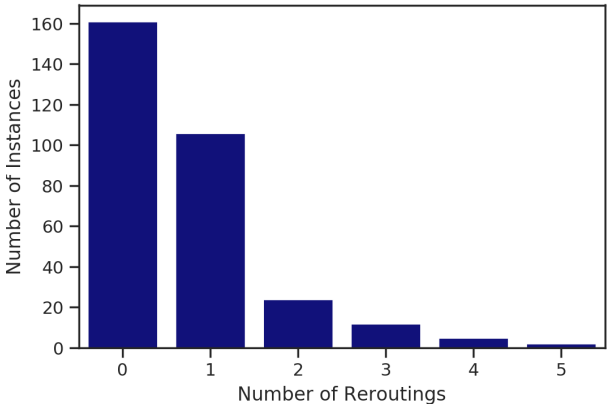


Figure 14: A bar plot of the number of reroutings in each solution.

these solutions, which are optimal under our objective function. In these instances, the schedule event weights β_j^k for platform alternatives were discounted by a factor of 0.9. The number of reroutings would be reduced if this parameter in the objective function were smaller.

8.2 Solving with a Strict Time Limit

Each instance was solved with a time limit of 20 seconds, in order to assess the performance of the algorithm when the available time is severely limited as it is in the real-time setting. We found that 81.6% of instances were solved to optimality within this time limit. Of those that were not solved to optimality, the median gap between the best integer solution and the best bound on the optimal integer solution was 0.17% and the mean was 0.64%. A histogram of the obtained integrality gaps is shown in Figure 15.

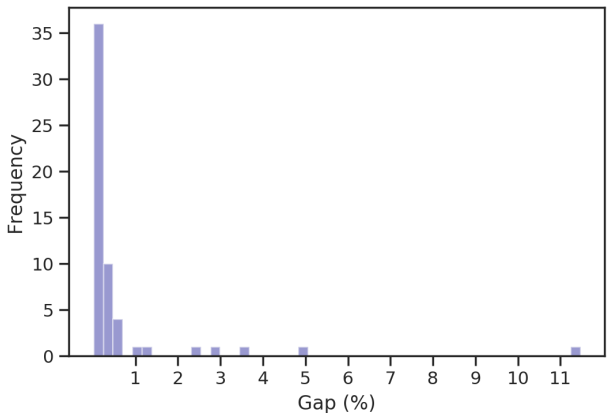


Figure 15: A histogram of the percentage gap between the best integer feasible solution found, and the best bound on the optimal solution.

These results demonstrate that the developed algorithm could be effectively used in a real-time setting. Even within the very stringent time limit of 20 seconds, most of the instances can be solved to optimality. Solutions within a few percentage points of optimality can be found for the remainder. The fact that solution quality can be guaranteed even when

the optimal solution is not found is a significant advantage of this method over heuristic methods that cannot do the same. This is because knowledge of the solution quality would be an important factor in any decision over whether or not to accept a solution and implement it in practice.

9 Conclusion

In this paper, we present a new model for the Train Timetable Rescheduling Problem that is capable of performing rerouting and retiming simultaneously. The model employs the concepts of occupying and banning to model sectional-release interlocking whilst using routes as the track components. This is a key contribution, since it shows how track capacity can be modelled in an accurate yet tractable fashion. We also present a new objective function that was developed in collaboration with Network Rail. This objective function uses the concept of utility to tractably and flexibly represent Network Rail's complex rescheduling preferences.

Our experiments, which were conducted on a new set of test instances based on real data, show that the majority of instances can be solved to optimality in 20 seconds. The remainder can be solved to provably near to optimality in this time limit, with very good guarantees on solution quality. We show that using partial pricing and reduced cost variable fixing are effective in accelerating our tailored branch-and-price algorithm. At a broader level, these results demonstrate that it is realistic to use exact time-indexed methods in a real-time environment. Finally, our results confirm the conjecture raised by our theoretical result, that instances with a greater level of conflict between train movements are more difficult to solve.

Future research will focus on simultaneously calculating train speed profiles. The assumption that all trains are capable of achieving the same traversal times may not be realistic in railways with mixed traffic. Another potential direction for further work is to refine the objective function. Economic methods could be used to estimate the utility function of Network Rail, for example by testing their preferences between solutions produced with different parameter values. The empirical relationship between these values and the resulting optimal rescheduling actions is not well understood. Finally, a big challenge for the future is to extend this model to whole regional networks of track, either by coordinating many smaller models or by improving the solution techniques so that very large instances can be solved.

Acknowledgements

Funding: ER was supported by the EPSRC-funded STOR-i Centre for Doctoral Training [grant number EP/L015692/1] and Network Rail. SJM was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/P003060/1.

References

Acuna-Agost, R., Michelon, P., Feillet, D., and Gueye, S. (2011). A MIP-based Local Search Method for the Railway Rescheduling Problem. *Networks*, 57(1):69–86.

Bettinelli, A., Santini, A., and Vigo, D. (2017). A real-time conflict solution algorithm for the train rescheduling problem. *Transportation Research Part B: Methodological*, 106:237–265.

Binder, S., Maknoon, Y., and Bierlaire, M. (2017). The multi-objective railway timetable rescheduling problem. *Transportation Research Part C: Emerging Technologies*, 78:78–94.

Borndörfer, R. and Schlechte, T. (2007). Models for railway track allocation. In *7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*.

Bouveyron, C., Celeux, G., Murphy, T. B., and Raftery, A. E. (2019). Model-Based Clustering and Classification for Data Science: With Applications in R. In *Model-based Clustering and Classification for Data Science*, pages 15–75. Cambridge University Press.

Brännlund, U., Lindberg, P. O., Nou, A., and Nilsson, J.-E. (1998). Railway Timetabling Using Lagrangian Relaxation. *Transportation Science*, 32(4):358–369.

Cacchiani, V., Caprara, A., and Toth, P. (2008). A column generation approach to train timetabling on a corridor. *4OR*, 6(2):125–142.

Cacchiani, V., Huisman, D., Kidd, M. P., Kroon, L. G., Toth, P., Veelenturf, L. P., and Wagenaar, J. C. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37.

Caimi, G., Chudak, F., Fuchsberger, M., Laumanns, M., and Zenklusen, R. (2011). A New Resource-Constrained Multicommodity Flow Model for Conflict-Free Train Routing and Scheduling. *Transportation Science*, 45(2):212–227.

Caprara, A., Fischetti, M., and Toth, P. (2002). Modeling and Solving the Train Timetabling Problem. *Operations Research*, 50(5):851–861.

Caprara, A., Galli, L., and Toth, P. (2011). Solution of the Train Platforming Problem. *Transportation Science*, 45(2):246–257.

Corman, F., D'Ariano, A., Pacciarelli, D., and Pranzo, M. (2010). A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B*, 44(1):175–192.

Corman, F., D'Ariano, A., Pacciarelli, D., and Pranzo, M. (2012). Bi-objective conflict detection and resolution in railway traffic management. *Transportation Research Part C*, 20(1):79–94.

Corman, F., Goverde, R. M. P., and D'Ariano, A. (2009). Rescheduling Dense Train Traffic over Complex Station Interlocking Areas. In *Robust and Online Large-Scale Optimization*, pages 369–386. Springer.

Corman, F. and Meng, L. (2015). A review of online dynamic models and algorithms for railway traffic control. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1274–1284.

D'Ariano, A., Corman, F., Pacciarelli, D., and Pranzo, M. (2008). Reordering and Local Rerouting Strategies to Manage Train Traffic in Real Time. *Transportation Science*, 42(4):405–419.

D'Ariano, A., Pacciarelli, D., and Pranzo, M. (2007). A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657.

Desaulniers, G., Desrosiers, J., and Solomon, M. M. (2002). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In *Essays and surveys in metaheuristics*, pages 309–324. Springer US.

Desrosiers, J. and Lübbecke, M. E. (2005). A primer in column generation. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, volume 3, pages 1–32. Springer.

Fang, W., Yang, S., and Yao, X. (2015). A Survey on Problem Models and Solution Approaches to Rescheduling in Railway Networks. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):2997–3016.

Geoffrion, A. (1974). Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114.

Gleixner, A., Maher, S. J., Fischer, T., Gally, T., Gamrath, G., Gottwald, R. L., Hendel, G., Koch, T., Lübbecke, M. E., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Weninger, D., Witt, J. T., and Witzig, J. (2018). The SCIP Optimization Suite 6.0. Technical report, ZIB.

Gurobi Optimization LLC (2020). Gurobi Optimizer Reference Manual, http://www.gurobi.com.

Harrod, S. (2011). Modeling Network Transition Constraints with Hypergraphs. *Transportation Science*, 45(1):81–97.

Irnich, S., Desaulniers, G., Desrosiers, J., and Hadjar, A. (2010). Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing*, 22(2):297–313.

Kahn, A. B. (1962). Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

Lusby, R. M., Larsen, J., Ehrgott, M., and Ryan, D. M. (2011). Railway track allocation: Models and methods. *OR Spectrum*, 33(4):843–883.

Lusby, R. M., Larsen, J., Ehrgott, M., and Ryan, D. M. (2013). A set packing inspired method for real-time junction train routing. *Computers and Operations Research*, 40(3):713–724.

Mannino, C. and Mascis, A. (2009). Optimal Real-Time Traffic Control in Metro Stations. *Operations Research*, 57(4):1026–1039.

Mascis, A. and Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517.

Meng, L. and Zhou, X. (2014). Simultaneous train rerouting and rescheduling on an N -track network : A model reformulation with network-based cumulative flow variables. *Transportation Research Part B*, 67:208–234.

Min, Y. H., Park, M. J., Hong, S. P. S. H., and Hong, S. P. S. H. (2011). An appraisal of a column-generation-based algorithm for centralized train-conflict resolution on a metropolitan railway network. *Transportation Research Part B: Methodological*, 45(2):409–429.

National Audit Office (2008). Reducing passenger rail delays by better management of incidents. Retrieved from https://www.nao.org.uk/wp-content/uploads/2008/03/0708308.pdf. Accessed on 17/04/20.

Network Rail (2019). Network Rail Limited Annual Report and Accounts 2019: Strategic report.

Network Rail (2020). Railway Performance. Retrieved from https://www.networkrail.co.uk/who-we-are/how-we-work/performance/railway-performance/. Accessed 17/04/20.

Pachl, J. (2014). Timetable Design Principles. In Hansen, I. A. and Pachl, J., editors, *Railway Timetabling & Operations*, pages 9–42. Eurailpress, 2nd editio edition.

Pellegrini, P., Marlière, G., Pesenti, R., and Rodriguez, J. (2015). RECIFE-MILP: An Effective MILP-Based Heuristic for the Real-Time Railway Traffic Management Problem. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2609–2619.

Pellegrini, P., Marlière, G., and Rodriguez, J. (2014). Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B: Methodological*, 59:58–80.

Reynolds, E. (2020). TTRP Full Results. www.github.com/edwin6/TTRP_Results.

Samà, M., Meloni, C., D'Ariano, A., and Corman, F. (2015). A multi-criteria decision support methodology for real-time train scheduling. *Journal of Rail Transport Planning and Management*, 5(3):146–162.

Samà, M., Pellegrini, P., D'Ariano, A., Rodriguez, J., and Pacciarelli, D. (2016). Ant colony optimization for the real-time train routing selection problem. *Transportation Research Part B: Methodological*, 85:89–108.

Törnquist, J. and Persson, J. A. (2007). N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B: Methodological*, 41(3):342–362.

Törnquist Krasemann, J. (2012). Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances. *Transportation Research Part C: Emerging Technologies*, 20(1):62–78.

Vanderbeck, F. and Wolsey, L. (2010). Reformulation and Decomposition of Intger Programs. In *50 Years of Integer Programming 1958-2008*, pages 431–502. Springer.

Zwaneveld, P. J., Kroon, L. G., and Van Hoesel, S. P. M. (2001). Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128:14–33.