Simultaneous disruption recovery of a train timetable and crew roster in real time

Cameron G. Walker\textsuperscript{a,*}, Jody N. Snowdon\textsuperscript{b}, David M. Ryan\textsuperscript{a}

\textsuperscript{a}Department of Engineering Science, University of Auckland, Auckland, New Zealand
\textsuperscript{b}Optimal Decision Technologies, New Zealand

Abstract

This paper describes the development and implementation of an optimization model used to resolve disruptions to an operating schedule in the rail industry. Alterations to the existing train timetable and crewing roster are made simultaneously in real time—previous treatments in the literature have always decoupled these two problems and solved them in series. An integer programming model is developed that constructs a train timetable and crew roster. This model contains two distinct blocks, with separate variables and constraints for the construction of the train timetable and crew roster, respectively. These blocks are coupled by piece of work sequencing constraints and shift length constraints, which involve variables from both blocks. This unique parallel construction process is then used as the basis of a method to deal with the resolution of train disruptions in realtime. Favourable results are presented for both the combined train/driver scheduling model and the real-time disruption recovery model.

© 2004 Elsevier Ltd. All rights reserved.

1. Introduction

In this paper, we consider the problem of recovering in real time from a disruption to a planned train schedule, which involves adjusting both the train and crew timetables. This research was conducted in association with a rail operator who resolves schedule disruptions manually, relying entirely on human decision-making. The train controller, using a train controller diagram (TCD) of the final train timetable, as well as a ruler and a number of different coloured pencils, traces the course through time of each train over its set schedule, making alterations to the current timetable as follows.

* Corresponding author.

E-mail address: cwalker@math.auckland.ac.nz (C.G. Walker).
When there is a delay to a train service, the train controller identifies which trains are affected by the delay and alters the schedule of the affected train with the highest priority first (where its priority is decided by the train controller based on the type of train, or the impact of its delay). The train controller then plots the new schedules for the remaining trains in order of priority so as to accommodate the new schedule of the high priority train. All this must be done in real time, and any changes to the schedule logged into a database for auditing reasons, along with a code identifying the cause of the disruption. Any changes to the driver shifts are made as a consequence of this process—for example the location of a train-crossing may be altered due to a disruption, which in turn results in the locomotive engineers switching trains at this new location. Such a change to driver shifts must continue to satisfy operating rules and conditions on driver duties.

Recovery involves two related processes, the first involving the determination of a revised or amended train schedule and the second involving the adjustment or repair of the associated driver duties. Since each of these processes is affected by the solution of the other, we attempt here to construct the train timetable and crew schedule simultaneously. In order to do this, we initially develop a model that links the two underlying problems of train and driver scheduling. This model is a first step towards the disruption recovery model. While this model could be used to generate train timetables and driver schedules from scratch, our purpose is to develop the composite model that considers changes to a pre-existing timetable and a set of corresponding driver shifts. Schedule disruption is more severe where single-track sections are present, as is the case in a large proportion of the New Zealand rail network. The performance of this model is detailed here for the South Island Coal route, a small isolated single-track sub-network of the New Zealand rail network that is suitable for testing such a model.

Based on this initial composite model, we then develop an approach for disruption recovery in realtime. The objective of our recovery model is to minimize the deviation from the existing schedule while incurring as little cost increase as is possible from the adjusted crew shifts. Thus we hope to alter the existing schedule by as small amount as is possible. In particular, we can expect the driver shifts to change very little, although the location of some train crossings may be affected. This means that our goal of producing good solutions to a train disruption in real time is achievable, as we can speed up the solution process by hot-starting from the previous solution. Also by limiting the period into the future for which the schedule is resolved, we can further reduce the time taken to produce solutions. From a practical point of view there is little point solving too far into the future, as it is quite possible that a new disruption will occur before the end of this new schedule is reached. Ideally the solver should be running continuously with the new disruption information being entered as it becomes available. The width of the time-window into the future for which we solve can be adjusted dynamically depending on the number of trains running and the solution speed required. We trial this model on the Wellington Metro line—a city line that runs from 6 a.m. through to midnight. This problem provides us with a predetermined time-window width of 18 h.

The layout of the paper is as follows. Section 2 is a general background to the train-scheduling problem with some references. In Section 3, we give more specific detail concerning the rules for scheduling in the New Zealand setting. In Sections 4 and 5, we outline the construction of the combined train/driver scheduling model, with the complete model being presented in Section 6. In Section 7, we present the disruption recovery for resolving disruptions to the schedule in real time. In Section 8, we explain the solution process. In Section 9, we discuss our results, and finally, in Section 10, we give conclusions and mention possibilities for future work.
2. Background

Throughout the 1960s, 1970s, and 1980s literature relating to rail optimization has been relatively limited. Compared to the bus and airline industries, optimization was generally overlooked in favour of simulation or heuristic-based methods. Throughout the 1990s this trend has changed, with the use of optimization techniques becoming more prevalent in the rail industry. Cordeau et al. [1] point to stronger competition, privatization, deregulation, and increasing computer speeds as reasons for this change.

2.1. Train crew scheduling

The train-scheduling problem is more difficult than its counterpart for buses, for example, because the set covering formulations are considerably larger. One reason for this is that the relief opportunities are closer together, meaning that the train-scheduling problem has more pieces of work to consider. A piece of work is a term used in the bus scheduling literature to define the time period in a bus timetable between two successive relief opportunities. Other difficulties can arise due to the geographical size of the network. For instance, driver passengering (usually termed paxing or deadheading) is often required in the case of the train-scheduling problem.

Despite the greater complexity of the train-scheduling problem, it nevertheless shares many characteristics with the bus-scheduling problem, and so it is of no surprise that many of the approaches to train scheduling in the literature involve the modification of an existing model for buses. Techniques used include limited subsequence and Lagrangian relaxation, as well as the adaption of existing bus crew-scheduling systems such as IMPACS [2] and HASTUS [3]. The IMPACS team augmented their solver with column generation in the mid to late 1990s to develop TRACS II [4], increasing the size of the solvable problems. TRACS II is used for both bus and train crew scheduling. In the case of train scheduling in New Zealand there has been work done in conjunction with Tranz Metro that lead to a set partitioning base model with embedded column generation [5,6]. This work provides optimal crew pairings given a fixed train timetable (because of the dense nature of passenger services this approach is acceptable). (See [7] for a more comprehensive discussion of the literature.)

2.2. Train crossing resolution

One unique difficulty, when scheduling in the rail context, is crossing resolution. On single-track sections of the network that are traversed in both directions, sidings large enough to hold a train are placed alongside the track at various points to enable train crossings and overtakings. The problem of deciding where trains should cross or overtake to optimize a given objective is called the train crossing problem (TCP).

It has been noted that the problem of scheduling trains is analogous to the problem of scheduling jobs on a set of machines, with the trains corresponding to the jobs and the sections of track being identified with the machines. In the job-scheduling context many of the problems are known to be NP-hard, although for some simpler problems linear programming techniques have been successfully implemented. The early attempts at linear programming approaches to these problems occurred in the 1960s. The objectives investigated included “minimize the latest job time,” “minimize the average job
time,” and “minimize the total machine idle time”. Szpigel applied similar mathematical programming techniques to the train-scheduling problem, solving a small problem with a modified version of the branch-and-bound algorithm [8]. Szpigel’s objective was to “minimize the weighted average of train travel times”.

At the time Szpigel’s approach was of little use, due to limited computing power. Throughout the 1970s heuristic approaches were more popular, and continued to be preferred, in conjunction with complete enumeration techniques, throughout the 1980s and into the 1990s. However, computer power has increased dramatically in the last 25 years and it is now realistic to attempt approaches such as Szpigel’s to solve complex train scheduling problems very efficiently.

Throughout the 1990s and into the new millennium research output in the rail industry has increased dramatically. One of the popular problems considered using optimization techniques is the generation of periodic train timetables that minimize passenger-waiting time [9–11]. For a detailed review of these and other related problems we direct the interested reader to [12] or, for a general survey of the optimization problems considered in train routing and scheduling, to [1].

2.3. Simultaneous vehicle and crew scheduling

Attempts to consider the vehicle and crew scheduling problems simultaneously have been made in the bus and airline industry, although the vehicle scheduling aspect of the problem differs significantly from what we are considering.

In the airline industry Stojković and Soumis [13] have constructed an optimization model for simultaneous operational flight and pilot scheduling. The problem is formulated as an integer non-linear multicommodity flow problem, which they solve with a Dantzig–Wolfe decomposition and a Branch and Bound method. Their model is designed to modify one-day planned individual activities when a disruption or alteration to an existing schedule has been made. Cordeau et al. [14] have employed Benders decomposition to simultaneously route aircraft and roster their crew at minimum cost, in the situation where the flight legs are to be flown by only one type of aircraft. They employ column generation techniques and a Branch and Bound method to solve their problem.

Haase et al. [15] have used column generation together with Branch and Bound in the bus industry to simultaneously schedule vehicles and crew, in the case where there is only one depot with a homogeneous fleet of vehicles. Their approach is very similar to that of Klabjan [16] in the airline industry, and has proved successful in solving larger problems than previous exact solution techniques in the bus industry [17]. Many heuristic or optimization-based heuristics have also been employed to address this problem in the bus industry [18,19].

As mentioned in Section 2.2, the problem of scheduling vehicles in the rail industry is made more difficult by the presence of single-track sections in the rail network. In the rail industry it is necessary to decide the exact location where services travelling in opposite directions will cross, or where a faster service travelling in the same direction will overtake. There are instances of optimization models for rail that account for both the routing and scheduling of vehicles together (to ensure that feasible timetables can be constructed) [20–22], but we have found no examples of train scheduling and crew rostering being included in the same optimization model.
3. Rail scheduling in New Zealand

In New Zealand there are specific rules that govern the construction of the train timetables and associated driver shifts.

3.1. Shift rules

*Book on* is the amount of time elapsed between the beginning of the shift and the first driving activity. *Book off* is defined in a similar way. The duration of the Book on and Book off is a standard 25 min, although this may alter due to the nature of the first and last activity in the shift. A train which requires shunting or turning may require a longer book on time, whereas passenger trains generally require less than the 25 min. The maximum length of a given shift is a function of its start time, as is the maximum amount of time spent driving the train (known as the *footplate*). These times vary from 10 to 11 h for shift lengths, and from 8.5 to 9 h for footplate times. Each shift over 4 h must contain a *personal needs break* (PNB) during which the driver is idle. The time and location at which the PNB is taken is not subject to strict rules, although there are so called *soft rules*, which the scheduler will try to adhere to when planning the PNB. Basically it is desirable to have the PNB placed as near to the middle of the shift as possible, so that the outward and homeward footplate times are relatively equal. It is also preferable to have the PNB coincide with a train crossing as this provides a more robust train timetable.

3.2. Train rules

Each train is subject to a departure window that constrains the start time of the train. In some instances it is also useful to constrain the arrivals of trains at terminals and intermediate stations. The sharing of locomotives by train services may make it necessary to link the departure of one train with the arrival of an earlier one, to allow time for the decoupling and coupling of the locomotive. The loading and unloading of freight may provide additional constraints on some trains, as should any scheduled stops that the trains may have to adhere to.

Trains are given priorities $\rho^k$ that represent the cost of train $k$ being idle for one unit of time with respect to other trains. This priority is based on an estimate of the true cost (calculated from such factors as expense to start the train moving again), and also on the inconvenience caused by having the train wait. The train priorities are used when resolving train crossings to help decide which train should take the siding and wait. Upon the completion of a train crossing or overtaking, the train standing in the siding is required to wait a small period of time before continuing on its journey for safety reasons.

4. The train timetable

For a train timetable to be considered feasible it must fulfil all the physical requirements inherent in the real-world problem. Trains must traverse the sections of the network in logical order, and must cross at sidings or stations. Scheduled stops and crossing delays must be observed. If locomotives are shared, enough time must be given for transferring the engine from one train to another. We
introduce constraints for all these physical requirements to ensure that our model produces feasible solutions that can be implemented in practice.

We begin the description of our model with some notation. The following parameters are defined: 

\( \tau_i^k \) denotes the duration of piece of work \( i \) by train \( k \); 
\( \sigma_j^k \) denotes the duration of a scheduled stop for train \( k \) after piece of work \( i \); 
\( \delta_{jk} \) represents the minimum separation time between the arrival of train \( j \) and the departure of train \( k \), where trains \( k \) and \( j \) share the same locomotive; 
\( l_k \) denotes the earliest start time for the first piece of work on train \( k \) and \( u_k \) denotes the latest start time for the first piece of work on train \( k \).

The following decision variables are defined for the train timetable: 

\( a_k^i \) denotes the start time of piece of work \( i \) on train service \( k \) and 
\( s_k^i \) is the amount of time train \( k \) remains idle upon the completion of piece of work \( i \).

For the purposes of this discussion, let us suppose that the pieces of work on train \( k \) are arranged sequentially according to its route and numbered \( 1, 2, \ldots, n_k \), and that the number of trains to be scheduled is \( m \).

For each train operating on the network, we have to ensure that each piece of work \( i + 1 \) is not started until piece of work \( i \) has been completed. Furthermore, any stop scheduled after the completion of piece of work \( i \) must also be observed before the commencement of piece of work \( i + 1 \). This time constraint is illustrated in Fig. 1.

In this figure, the amount of time between the commencement of piece of work 1 and piece of work 2 on train \( k \) is equal to the duration of piece of work 1 (\( \tau_i^k \)) plus the duration of the scheduled stop at the completion of piece of work 1 (\( \sigma_i^k \)) plus some idle time (\( s_i^k \)).

This means that the following set of constraints must be satisfied for \( k = 1, \ldots, m \):

\[
da_{i+1}^k - a_i^k = \tau_i^k + \sigma_i^k + s_i^k, \quad i = 1, 2, \ldots, n_k.
\] (4.1)

Each train has an earliest and latest departure time associated with it, so for \( k = 1, \ldots, m \) we must also include constraints of the form

\[
l_k \leq a_1^k \leq u_k.
\] (4.2)

If trains \( j \) and \( k \) share a locomotive we must ensure that sufficient time (\( \delta_{jk} \)) is scheduled for the transfer of the locomotive to train \( k \) after the completion of the last piece of work for train \( j \). Thus, if \( a_n^j \) is the start time for the last piece of work on train \( j \), we must also add a constraint of the form

\[
da_1^k - a_n^j \geq \tau_n^j + \sigma_n^j + \delta_{jk}.
\] (4.3)
The most difficult feature of the train-scheduling problem is to decide the location of crossings and over takings. In Fig. 2, we have the example of an illegal train crossing. One of train $j$ or train $k$ must wait at their station until the other train has passed.

A legal train crossing is enforced by including the following constraints:

$$a^j_i - a^k_i - \gamma_{il}^{jk} M \geq \tau^k_i - M + \mu,$$

$$a^k_i - a^j_i + \gamma_{il}^{jk} M \geq \tau^j_i + \mu,$$

where $\mu$ is a small constant representing the crossing delay and $\gamma_{il}^{jk}$ is a binary variable set to 0, if train $k$ waits for train $j$, or 1, if train $j$ waits for train $k$. The constant $M$ is chosen to be sufficiently large so that if $\gamma_{il}^{jk} = 1$ constraint 4.4 must be satisfied while constraint 4.5 is always satisfied. This ensures that train $j$ waits $\mu$ minutes after train $k$ has arrived at station A before it makes its departure. If $\gamma_{il}^{jk} = 0$ constraint 4.5 is always satisfied while constraint 4.4 must be satisfied, and train $k$ waits $\mu$ minutes after train $j$ has arrived at station B before it makes its departure. Constraints 4.4 and 4.5 can be added dynamically when needed in the solution process to prevent the model from growing too large (see discussion in Section 8).

We can deal with an illegal train overtaking in a similar way. If train $j$, (while doing piece of work $l$) is on course to pass train $k$ (while it is doing piece of work $i$) at a point between two stations, we can resolve this infeasibility with the following pair of constraints:

$$a^j_i - a^k_i - \gamma_{il}^{jk} M \geq \tau^l_i - M + \nu,$$

$$a^k_i - a^j_i + \gamma_{il}^{jk} M \geq \nu.$$

Here, the constant $\nu$ represents the section clear delay and $\gamma_{il}^{jk}$ is a binary variable set to 0, if train $k$ waits for train $j$, or 1, if train $j$ waits for train $k$. If $\gamma_{il}^{jk}$ takes value 1 then constraint 4.6 is binding and train $j$ must wait for train $k$. Since, we must enforce the section clearance between the two trains and train $j$ is faster than train $k$ it is necessary to include on the right-hand side of constraint 4.6 the difference in time that the two trains take to complete the particular section ($\tau^k_i - \tau^j_i$). Clearly it is only necessary for the slower train to wait the length of the section clear delay if it is held at the station. This is the case when constraint 4.7 is binding ($\gamma_{il}^{jk} = 0$).
5. Driver shifts

As with the train services, driver shifts can be viewed as a sequence of pieces of work, although in the case of driver shifts the sequencing is determined as part of the solution process. The method used to construct a complete set of driver shifts is as follows.

For each piece of work we must identify the pieces of work that can directly follow with regards to time and location (we will call these pieces of work subsequences). Subsequences are determined in the following way. The first piece of work on a train \( k \) has an earliest and latest departure time \( l^k \) and \( u^k \), respectively. If we let \( \phi \) represent the maximum expected amount of idle time incurred following a piece of work, then \( l^k + \tau^k_1 \) is the earliest possible start time for the second piece of work on train \( k \) and \( u^k + \tau^k_1 + \phi \) is the latest possible start time for this piece of work, where \( \tau^k_1 \) is the duration of the first piece of work on train \( k \). By iterating this process it is possible to determine expected departure windows for each train from each station on its route. If the expected departure window for the train performing piece of work \( i \) overlaps with the expected arrival window of the train performing piece of work \( j \) then we identify piece of work \( i \) as a subsequence of piece of work \( j \). We maintain some maximum width on the departure window so that the difference between the earliest and latest expected start time for any piece of work does not grow too large. This limits the number of subsequences for any given piece of work and improves the quality of the shifts generated (reducing train driver idle time). Note that one piece of work cannot follow another in the same direction unless it is on the same train. This rule prevents the generation of shifts where trains are left idle without a driver.

Once all next subsequences are determined, it is possible to construct all potential driver shifts, excluding personal needs breaks. The first subsequence for any given piece of work is always the next piece of work on the same train, if the service is still running. Note that because we enforce a maximum width for any expected departure window and because the only possible train swap for a driver is with a service travelling in the opposite direction, the number of subsequences for any piece of work is kept small. At this stage we ensure that only legal shifts are generated (in terms of footplate duration, start and finish location, and number of distinct trains driven).

These shifts are constructed prior to the solution of the model. During the solution process a column generator inserts personal needs breaks into these a priori shifts to produce columns for the constraint matrix. Although our aim in constructing this model is to recover from a disruption, it still makes sense to include the financial cost of the recovery in our objective function. Determining the cost of each shift is complicated by the fact that it must be done prior to the construction of the train timetable. Thus the length of the shift is approximated solely by the sum of the pieces of work it covers, ignoring the possibility of idle time. Each driver is guaranteed payment for a minimum number of hours, regardless of how long he works. Furthermore, after a certain point the hourly payment increases to an overtime rate for the driver. These two factors mean that it is desirable to generate shifts of a length longer than the driver minimum payment but short enough to avoid overtime payments. Thus each shift \( S_j \) has cost

\[ c_j = |\omega_j - \Omega|, \]

where \( \Omega \) is a predetermined target footplate and \( \omega_j \) is the sum of the \( \tau_i \) (in effect a lower bound on the true footplate time).
If a binary variable $x_j$ is associated with each $S_j$, such that $x_j = 1$ if $S_j$ is included in the solution and $x_j = 0$ otherwise, then the crewing goal can be expressed by the set partitioning model

$$\text{Min } c^T x$$

s.t. $Ax = e,$

$$x \in \{0, 1\},$$

where $a_{ij} = 1$ if piece of work $i$ is covered by shift $S_j$ and $a_{ij} = 0$ otherwise. The number of rows of $A$ corresponds to the total number of pieces of work specified in the problem.

To ensure that consecutive pieces of work within driver shifts are time consistent, train swap constraints similar to those for the train timetable are added. Assuming that the $n$ pieces of work in a base shift are numbered sequentially, the following constraint must hold for sequential pieces of work on different trains:

$$a_{d_i - 1}^k - a_{d_i}^j - Mx_i \geq \tau_i^j - M, \quad i = 1, 2, \ldots, n - 1 \text{ and } j \neq k.$$

The subscript $d_i$ gives the position in the associated train’s route where the train does piece of work $i$ from the driver’s schedule, and $x_i = 1$ if the shift $S_i$ is included in the solution and $x_i = 0$ otherwise. The constant $M$ is chosen to be sufficiently large so as to ensure that constraint 5.1 is void if $S_i$ is not included in the solution.

We also include constraints to force the duration of the shifts to be less than the maximum shift length. If we let $I^k$ be the indexing set of all shifts having first piece of work $a_1^l$ and last piece of work $a_n^k$, then this maximum shift length is enforced by including constraints of the form:

$$a_n^k - a_1^l + M \sum_{i \in I^k} x_i \leq f(a_1^l) + \tau_n^k + M.$$  

(5.2)

The function $f(a_1^l)$ returns the maximum allowable length for a shift with first piece of work starting at time $a_1^l$, and the constant $M$ is suitably chosen to ensure that constraint (5.2) is void if $\sum_{i \in I^k} x_i = 0$.

It is still necessary to include the personal needs break somewhere in the shift, and so constraint 5.1 is modified to take care of this. If $I_{p,q}$ is the indexing set of shifts that have piece of work $p$ on train $j$ followed immediately by piece of work $q$ on train $k$, and $I_{p,q}$ is the indexing set of shifts having a personal needs break of length $\Phi$ between activities $p$ and $q$, then the modified constraint is given below.

$$a_q^k - a_p^l - (M + \Phi) \sum_{i \in I_{p,q}} x_i - M \sum_{i \in I_{p,q}} x_i \geq \tau_p^l - M.$$  

(5.3)

Note that due to the set partitioning formulation if a shift is included in the solution, where piece of work $p$ is followed immediately by piece of work $q$, then

$$\sum_{i \in I_{p,q}} x_i + \sum_{i \in I_{p,q}} x_i = 1.$$  

Thus $x_i$ will be 1 for one $i$ in $I_{p,q}$ or $I_{p,q}$ exclusively, and 0 for all other indices. If $x_i$ is 1 for some $i$ in $I_{p,q}$ then $\sum_{i \in I_{p,q}} x_i = 1$ and $\sum_{i \in I_{p,q}} x_i = 0$. This means that constraint 5.3 reduces to

$$a_q^k - a_p^l - M \geq \tau_p^l - M \quad \text{or} \quad a_q^k - a_p^l \geq \tau_p^l.$$
and no time for a personal needs break is scheduled between the activities $a_k^q$ and $a_p^j$. Otherwise
\[ a_k^q - a_p^j - \Phi \geq \tau_p^j. \]

This means that enough time for a personal needs break is scheduled between the two activities.

6. The combined train/driver scheduling model

A large portion of our model decouples naturally into a system of constraints for the train timetable generation, and a system of constraints for the driver shifts. The two non-interacting blocks are linked by piece of work sequencing constraints (5.3) and shift length constraints (5.2). The complete structure of the constraint matrix is shown in Fig. 3, with timetable and piece of work constraints shown above and the shift length and POW sequencing constraints, which link the two blocks, shown below.

The constraints for the full model are given below. Variable, parameter and index definitions are given in Sections 4 and 5.

\begin{align*}
\text{Train} & \quad \text{Driver shift} \\

a_{i+1}^k - a_i^k - s_i^k & = \tau_i^k + \sigma_i^k, \quad (4.1) \\

a_0^k & \geq j^k, \quad (4.2) \\

a_0^k & \leq u^k, \quad (4.2) \\

a_0^k - a_n^j - s_i^j & = \tau_n^j + \sigma_n^j + \delta^j, \quad (4.3) \\

a_l^j - a_i^k - \gamma_{il}^k M - s_i^k & = \tau_i^k - M + \mu, \quad (4.4) \\

a_l^j - a_i^k + \gamma_{il}^k M - s_i^k & = \tau_i^k + \mu, \quad (4.5) \\

a_l^j - a_i^k - \gamma_{il}^k M - s_i^k & = \tau_i^k - \tau_l^j - M + \nu, \quad (4.6) \\

A x & = e, \quad (4.7) \\

a_0^k - a_0^j & \leq f(a_0) + \tau_n^j + M, \quad (5.2) \\

a_q^k - a_p^j & \geq \tau_p^j - M. \quad (5.3)
\end{align*}

The variables $a$ and $s$ are non-negative and real. The variables $x_i$ and $\gamma_{il}^k$ are binary.

As discussed in Section 1, this model is not intended as a tool for generating train and crew schedules from scratch. However, if we include costs on the train timetabling variables to penalize train idleness, it could be used for this purpose (with the obvious disadvantage that the set partitioning block could be very large if the timetable has no predetermined structure). In this case, the objective is made up of two components. The first part minimizes the sum of the weighted train idle times, where the weights, $\rho^k$, are just the train priorities outlined in Section 3. A term, $\varepsilon a_i^k$, is also added to the objective to encourage early starts for train services. The value of $\varepsilon$ is arrived at via an iterative evaluation process involving the train scheduler—it should encourage early starts without dominating the rest of the objective function. We also want to encourage uniform length shifts that do not lie beneath a minimum payment threshold or extend into an overtime period, so in the second part of
out objective we cost our shifts in such a way as to penalize any deviation from a pre-specified target (see Section 5 for further discussion). The objective function is given below.

\[
\text{Minimize } \sum_{k=1}^{m} \left( \sum_{i=1}^{n_k} \rho^k s_i^k + \varepsilon a_i^k \right) + \sum c_j x_j.
\]

7. The disruption recovery model

We now use the model developed in Sections 4 and 5 to resolve disruptions to the train schedule in real time. In order to do this, we need to define a new parameter \( \bar{a}_i^k \), the scheduled start time of piece of work \( i \) on train service \( k \) and two new variables \( e_i^k \), the amount by which the \( a_i^k \) precedes the originally scheduled start time and \( d_i^k \), the amount by which the \( a_i^k \) follows the originally scheduled start time.

We also need to introduce a new constraint for every scheduled train departure throughout the train schedule, linking the new train departure times to the scheduled departure times

\[
a_i^k + e_i^k - d_i^k = \bar{a}_i^k, \quad i = 1, 2, \ldots, n_k.
\]

(7.1)

For the train on which the disruption has occurred, we also include a constraint of the form

\[
a_{i+1}^k \geq \beta_i^k + \sigma_i^k,
\]

(7.2)

where \( \beta_i^k \) is the time that train \( k \) completes the disrupted piece of work \( i \). For every other operating train we include a constraint of the form

\[
a_i^k = \psi_i^k,
\]

(7.3)

where \( \psi_i^k \) is the time at which the current activity began. Constraints 7.2 and 7.3 ensure historical consistency in our new train timetable.

Historical consistency in the driver shifts is dealt with at the generation stage. Of course, in the event of a disruption the actual sequence of pieces of work in a driver’s tour of duty is not expected to change significantly, so the solution time is reduced by hot starting from the previous solution, which is now infeasible.
The complete model, obtained by combining the constraints described in Sections 4 and 5, is as follows:

\[
\text{Minimize } \sum_{k=1}^{m} \sum_{i=1}^{n_k} p^k_i (e^k_i + d^k_i) + \sum_{j} c_j x_j
\]

subject to

\[
\begin{align*}
\text{Train Driver shift} & \quad \text{Driver shift} \quad \text{Ax} \\
\alpha^k_i & = \psi^k_i + \sigma^k_i, \quad (7.3) \\
\alpha^k_i & \geq \beta^k_i, \quad (7.2) \\
\alpha^k_i + e^k_i - d^k_i & = \tau^k_i + \sigma^k_i, \quad (7.1) \\
\alpha^k_i + s^k_i & \leq u^k, \quad (4.2) \\
\alpha^k_i - \alpha^k - s_i^k & = \tau^k_i + \sigma^k_i + \delta^k, \quad (4.3) \\
\alpha^k_i - \alpha^k + \gamma^k_i M - s_i^k & = \tau^k_i - M + \mu, \quad (4.4) \\
\alpha^k_i - \alpha^k + \gamma^k_i M - s_i^k & = \tau^k_i - \mu, \quad (4.5) \\
\alpha^k_i - \alpha^k + \gamma^k_i M - s_i^k & = \tau^k_i - \tau^k_i - M + v, \quad (4.6) \\
\alpha^k_i - \alpha^k + \gamma^k_i M - s_i^k & = v, \quad (4.7) \\
\alpha^k_i - \alpha^k + \gamma^k_i M - s_i^k & = e, \\
\alpha^k_i - \alpha^k + \gamma^k_i M - s_i^k & = f(a_0) + \tau^k_i + M, \quad (5.2) \\
\alpha^k_i - \alpha^k + \gamma^k_i M - s_i^k & \geq \tau^k_i - M. \quad (5.3)
\end{align*}
\]

Variable, parameter and index definitions are given in Sections 4, 5 and 7.

To reduce the computational effort required to solve this problem, we do not include all variables and constraints listed above in the initial model. In particular, constraints 4.4–4.7 are removed, along with all the \( \gamma^k_i \) variables. This results in a model that allows trains to cross at any point of the network. After we have solved the relaxation of this reduced model we resolve illegal train crossings in the branch and bound procedure by dynamically adding one of these constraints, with \( \gamma^k_i \) set to 0 or 1, depending on which train waits at the legal crossing point for the other train to pass. By resolving train crossings in this manner we reduce the number of constraints needed in the model and never need to include the \( \gamma^k_i \) variables, which greatly reduces the number of variables in the problem (as these particular variables have four indices). Deciding the location of personal needs breaks is also omitted from the initial model, with these decisions again being resolved in the branch and bound procedure. The dimensions of the problem is further restricted by dynamic column generation. These features of the solution procedure are discussed further in Section 8.

8. The solution procedure

The method used to solve the models described in the previous two sections is branch and bound with column and constraint generation. At each node of the branch and bound tree the associated
relaxation is solved using the primal revised simplex method (RSM), as implemented in ZIP [23]. ZIP is a collection of user routines supported by a powerful solver engine, created primarily for use in solving large 0–1 set partitioning formulations. In the ZIP methodology, the user is responsible for the storage and maintenance of the data structures in which the columns and elements of the model are stored, and for the passing of information from these structures to ZIP.

Using the idea of limited subsequence outlined in Section 5, legal potential driver shifts are generated before solving the optimization model, with legality depending on shift length, shift origin and destination, and number of train swaps. These shifts do not include personal needs breaks, which greatly reduces the number of shifts generated (from over 410,000 to less than 18,000 in the example discussed in Section 9.1). In fact, we reduce the number of these shifts further (to under 3000 in our example) by removing relief opportunities from the beginning and end of the shifts, which has the additional advantage of encouraging train swaps at locations other than the typically congested driver bases. The performance of the solution procedure is further improved by hot-starting from the solution that was feasible prior to the disruption.

Initially the driver shift variables are ignored in favour of piece of work and idle time variables, when pricing variables for entry into the basis. This provides an ordering of the pieces of work that is time consistent with their aggregation as train services. When there are no viable entering columns to be found from amongst this reduced set the driver shift variables are considered. The shift generation algorithm constructs at most one column from each of the initial potential driver shifts by placing the PNB after the first piece of work that yields a negative reduced cost. If the PNB does not coincide with a train swap a fixed penalty is added to the cost of this column. Initially only PNBs that coincide with train swaps are allowed. At optimality the shift generator is called again allowing PNBs to be placed after any piece of work that is followed by a period of idle time. This solution procedure produces a fractional solution to the combined model, as the driver shift variables can have fractional values. This fractionality, together with any illegal train crossings, is resolved in the Branch and Bound routine.

Fractional cover is resolved using the technique of constraint branching as described by Foster and Ryan [24]. For each piece of work and its subsequence the sum of fractional values of the shift variables containing the pair is calculated. Any piece of work and subsequence that does not have a sum of fractions equal to one or zero indicates fractional coverage. For example, consider the following two sequences of pieces of work representing two shift variables in the solution at fractional value:

\[
\text{Variable 1 at value 0.9: } 1 2 3 4 5 6,
\]

\[
\text{Variable 2 at value 0.1: } 1 2 3 4 5 10 11.
\]

Each pair of activities up until the pairs 5,6 and 5,10 will have a sum of fractions equal to 1. The pair 5,6 will have a sum equal to 0.9, and 5,10 will have a sum equal to 0.1. Consider the choice of resolving the fractional coverage of the 5,6 pair. There are two ways to resolve this fractional coverage: either 5 is followed by 6 or 5 is not followed by 6. This binary decision defines the two possible branches in the branch and bound tree. The branch direction is based, in part, on the magnitude of the sum of fractions for the activity pair. Typically, if the sum of fractions is greater than 0.5 (the 5,6 pair in our example), the so called 1-branch is chosen and driver shifts containing 5 not followed by 6 are removed from the current basis and from the linear programs.
of descendant nodes. The set of shifts removed will clearly contain those shifts having 5 followed by 10. A fractional sum less than 0.5 (the 5,10 pair in our example) implies the 0-branch and driver shifts are removed that contain 5 followed by 10. The particular branch we choose is best determined by considering the real world problem. If the fractioning occurs early within a shift, the tendency is toward choosing the branch that keeps the driver on the same train. Where there is no clear preference in terms of the application, the preference for branching is towards the 1-branch since this forces a pair of pieces of work to occur together and bans all shifts containing either piece of work occurring alone. Conversely, the 0-branch is much weaker, excluding only one piece of work pair from consideration.

Once all of the fractional coverage events have been resolved, the personal needs break location events are considered in time sequential order. These are resolved using constraint branching in the same way as for the fractional coverage. Finally any illegal train crossings and overtakes are resolved by the inclusion of constraints of type 4.4–4.7. The binary variable \( \gamma_{il}^{jk} \) provides two natural branches from this node; either \( \gamma_{il}^{jk} = 1 \), implying train \( j \) waits or \( \gamma_{il}^{jk} = 0 \), implying train \( k \) waits. The default branching direction is determined by the priority weighting of the two trains involved in the crossing, with the default branch allowing the higher priority train to traverse the segment first.

9. Results

The problems described in this section were solved on a Pentium 200 MHz computer with 64 MB RAM. The solution methodology was developed in FORTRAN.

9.1. Results for the combined train/driver scheduling model

Before solving the disruption recovery model, the combined train/driver scheduling model was trialed on the South Island Coal route. This route consists of a main track with 47 stations and sidings, and a spur branching at Stillwater, which is segmented by 5 sidings. This route contains 3 crew bases, one each at Middleton, Westport, and Greymouth. On a typical day 17 trains travel the network between the three terminal stations of Ngakawau, Rapahoe, and Lyttleton.

The linear program initially had 1093 constraints and took 1354 iterations (69 s) to calculate the optimal fractional solution. The shift generator was called 13 times returning a total of 2449 true driver shifts. The branch and bound process took 166 s to find an integer solution with an objective value within 10 percent of the fractional solution. The fractional solution had 14 unresolved train crossings, 69 instances of fractional shift cover and 21 unresolved PNB locations. The fractional shift cover and the unresolved PNB locations were removed from the solution by the first 12 branches. A further 13 branches were required to resolve the 14 illegal train crossings. The linear program at the final node had 78 more constraints than the fractional solution, all of these constraints added during the resolution of train crossings.

The objective of minimizing train idle time impacted significantly on the crew rostering, which is the area where the most significant savings can be made. By reducing the amount of time that a train service is idle at a station we also reduced the total time that crew spent waiting at a station, with the result that we were able to produce a crew roster with one less staff member than the initial roster designed by the associated rail operator. The optimal shift set has 11:07 fewer shift hours and 2:38 less footplate hours than currently operate. The redistribution of work results in a decrease
of 10 min in the average shift length (even though there is one less shift) and a slight increase of 16 min to the average footplate time.

Our solution is shown in Fig. 4. The train routes are numbered and follow the lines drawn on the diagram. The left axis is labeled with the stations on the Southern Coal Route, and the bottom axis represents time. For example, train service 844 leaves from Moana station a little after 8 p.m. on Monday and arrives at Middleton station at about 1 a.m. on Tuesday. The driver shifts overlay the train routes, with each different colour representing a different staff member. Idle time is represented by a horizontal line. To continue the above example, the driver for train 844 from Moana to Middleton started at about 4 p.m. on the 841 service from Middleton to Moana.

9.2. Results for the disruption recovery model

We tested the disruption recovery model on a simulation of the Monday timetable for the Wellington Metro line. On a Monday there are 36 trains travelling on the Metro line. Of these, 14 travel between the Plimmerton and Wellington stations, the remaining going the full distance between Paraparaumu and Wellington. We constructed the initial timetable for the day using our solver. During the solution process the train services were split into 564 pieces of work at possible relief points. In order to test the performance of our solver we chose three trains from different periods of the day (early, middle and late) and introduced a delay at the Muri station of 5, 15 and 45 min duration. The trains which incurred these delays were 6225, 6233 and 6239, all of which traverse the full extent of the Metro line.

The performance of the algorithm under the various scenarios is outlined in Table 1. In terms of the branch and bound process, the fractional coverage issue is not significant in this problem as the sequence of the pieces of work from a shift in the solution are not altered significantly by the delay of a train service. In effect, these issues are ironed out at the beginning of the day when our solver constructs the initial timetable. The personal needs break location is certainly an issue though, especially in the case where a 45 min delay occurs, as this introduces the requirement for
an additional personal needs break (the result of a shift being pushed over 4 h). Illegal crossings are also a major problem, as these can propagate throughout the day. In general one would expect that the bigger the initial disruption, the more work required to fix things. In the main, this is reflected in the results (with the one notable exception of the 5 min disruption to 6233). When we look at the solution performance as a whole, it is clear that, again omitting the case of the 5-min disruption on 6233, if a disruption occurs later in the day, then it will have a shorter solution time. If we consider a particular train in isolation, the longer the duration of the delay, the longer the solution time.

All the times presented in the “Time Total” column of Table 1 give the time taken to solve to optimality. Bearing in mind the processor speed used, it seems reasonable to assert that the solutions are produced in reasonable time. The times range from about 26 s to about 1 min 50 s, with the slowest being for a 45 min delay, and all of the delays for 15 min or less being resolved optimally in under 40 s. In the course of four train-controller shifts we observed that the resolution for a small window of the day’s schedule took about 35 s on average (performed by hand). Given the processor speed of computers currently on the market the approach to train disruption resolution detailed in this paper is definitely viable.

10. Conclusions and future work

The results from our initial tests give encouragement that we have developed technology that is efficient enough to aid in the day to day rescheduling of disrupted trains. Although the average time taken to solve to optimality was slightly under 50 s (which in itself is not an unreasonable time period to spend rescheduling), one must also keep in mind the fact that the speed of the processor used to produce these results is at most one tenth as fast as those currently available on the market.

Of course the network we have considered is still relatively small and there are improvements that can be made. One initial improvement would be to have the software present a list of next best solutions, allowing the train controller to choose from a number of possible schedules. Also, if the performance declines significantly on a larger network it may be necessary to solve for smaller time windows, as opposed to presenting a whole day solution. Resolving a disruption to the timetable a
set distance into the future would involve fewer variables and fewer constraints than resolving for the entire day, and so one would expect the solution time to diminish. In fact, the solution time should be affected directly by the length of time into the future for which we solve. This window would slide forward in time as the day progresses, providing snapshots of the problem to the solver from the current time to some fixed distance in the future. The snapshot that the solver sees would best be considered in two parts. If a disruption occurs within the window but beyond some pre-determined distance into the future it would be ignored. The reason for this is that there is little point in trying to reconcile an infeasibility so near the end of the window, only to change our solution at the next time step, when more of the day’s schedule beyond the disruption is visible to the solver. If the disruption occurs before this point, or if the window slides forward sufficiently for an infeasibility caused by a previous disruption to lie before this point, the solver would resolve the problem for the entire window.

This dynamic approach to disruption resolution presents many possibilities for future research. The width of the window could be altered in response to a change in the density of train disruptions. In a period of relative calm the solver could look further and further into the future, whereas when things are chaotic it may be more important to resolve for the immediate future as quickly as possible. There are also questions as to what the proportion of the window’s bipartition should be. In terms of implementation, the hardest problem to overcome will be the treatment of the driver shifts, as their legality depends on many factors, such as starting and ending at a crew base, which may not be observed in a smaller time window.

References