RAIRO-Oper. Res. 49 (2015) 413–434
DOI: 10.1051/ro/2014049

RAIRO Operations Research
www.rairo-ro.org

RAIL SCHEDULE OPTIMISATION IN THE HUNTER VALLEY COAL CHAIN

Gaurav Singh[1], Andreas T. Ernst[1], Matthew Baxter[1] and David Sier[1]

Abstract. This paper describes a method for scheduling trains on the Hunter Valley Coal Chain rail network. Coal for a particular ship is railed from different mines to stockpiles at one of the Port's terminals. The coal producers decide which mines will supply each order in what proportion, so there is no flexibility in the allocation of mines to cargoes. We are presented with a list of tonnes of coal which need to be transported from specified load points at mines to specified stockpiles at the port. The operators of the rail network provide a number of paths, with specified arrival and departure times, that can be used for coal movement. The requirement to assign coal trains to these existing paths makes this rail scheduling problem different to most of those discussed in the literature. In this paper we describe the problem in detail, demonstrate that it is very large making it difficult to solve with commercial MILP solvers, and show that our Lagrangian heuristic is able to produce high quality solutions in a reasonable amount of time.

Keywords. Mixed integer programming, coal supply chain, rail scheduling, lagrangian relaxation.

Mathematics Subject Classification. 90B06, 90B35.

1. Introduction

The Port of Newcastle in Australia is the world's largest coal export port, with a throughput in excess of 100 million tonnes per annum. The coal exported at the port is produced in the Hunter Valley region and must be carried by rail from

Received February 12, 2013. Accepted September 26, 2014.

[1] CSIRO Mathematics, Informatics and Statistics Private Bag 33, Clayton South MDC, 3168 Victoria, Australia. gaurav.singh@csiro.au

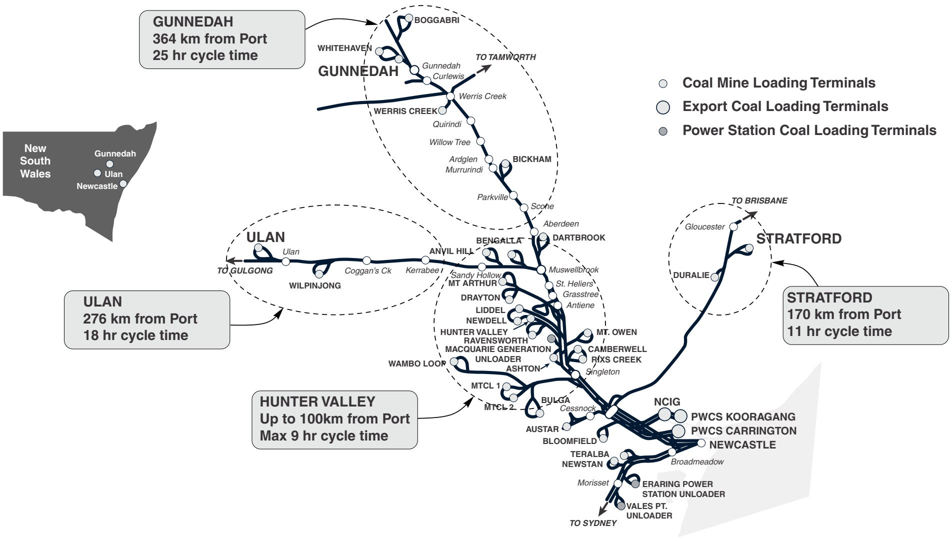

FIGURE 1. The Hunter Valley Coal Mines, Rail Infrastructure and Port of Newcastle [1].

the mines for distances up to 450 km. The Hunter Valley Coal Chain Coordinator (HVCCC) is responsible for coordinating the actions of 14 coal producers operating 40 mines, 3 Coal Loading Terminals at the port and 2 different rail operators. The size of the system can be seen in Figure 1 which shows the rail network for the Hunter Valley Coal Chain with four sub-regions, Gunnedah, Ulan, Hunter Valley and Stratford, each containing several mines. Figure 1 also presents some approximate statistics on these sub-regions to illustrate complexity of the coal chain. The Hunter Valley Coal Chain takes its name from the Hunter Valley sub-region which was the first region developed over the 200 year history of mining in the area.

This paper describes the operational train scheduling problem in the Hunter Valley Coal Chain. The emphasis of this paper is on describing this application and showing that the problems are too large to solve using commercial MILP solvers. We also describe a Lagrangian optimisation method which produces good heuristic solutions in an acceptable amount of time.

1.1. THE DEMAND FOR COAL BRANDS

The demand for coal is managed via contracts for coal brands that are blended from coal products from different mines. Ships arrive at the ports to load orders for a number of brands. An arriving ship provides data for the number and tonnages of brands to be loaded onto it, and consequently on the tonnes of products that have to be delivered from the different mines whose products are used to blend the brands.

Each brand on a ship is associated with a stockpile in the stockyard at the port. The coal products for a given brand must be railed from the mines in order to build its stockpile in time for loading onto the ship when it arrives. Individual stockpiles are built for each coal brand on a ship. Ships cannot berth at the port and start loading until all of the coal is ready in the stockpiles dedicated to that ship. This makes efficient and timely railing of coal essential to the smooth operation of the whole supply chain. The average capacity of a ship loading at Newcastle is 84 kt and it takes around 5–10 days to stockpile all the coal for a ship.

The shipping operators place the orders for coal brands in advance of a ship's arrival so the orders are fixed by the time operational train scheduling takes place. The coal producers will also have decided which of their mines will supply coal products for the orders, and in what proportions. Therefore there is no flexibility in the allocation of mines to the brands making up ship cargoes. A separate planning process is used to determine where in the stockyard stockpiles are to be built and the order in which they are stacked. This planning process considers a time horizon of up to two weeks and produces (amongst other outputs) the amount of coal from different mines that is to be railed each day. The rail scheduling problem considered here is the shorter term planning process that takes as input the tonnages to be moved from specified mines to particular stockpiles at the port. The typical planning horizon for this rail scheduling problem is a 24 h to 48 h period.

1.2. Railing coal from mines to ports

As can be seen from Figure 1 the train network has a relatively simple tree structure. There is a main trunk line with short loops to each of the mines. At the mines, trains are loaded at load points and may then wait for some time until there is an available return path (time slot) on the main line back to the port. While the real situation is slightly more complex, for the purposes of this paper only one train may be on any loop at a time and it can wait at the end of the loop to start its return path for as long as necessary.

The situation is a little more complex for the coal terminals at the ports. This is illustrated in Figure 2. When a train arrives at a terminal it waits for access to an inloading receival (or dump) station. However waiting at the terminal is generally considered less desirable than waiting at the mine's loadpoint loop. As the train moves through the dump station, doors on the bottom of the wagons are opened and the coal falls onto a conveyor belt feeding machines that stack the coal onto the stockpiles in the stockyard.

The stockyard has a number of rows, which are areas where stockpiles can be built. The stacking and reclaiming yard machines travel on (single) tracks between the rows, so only certain machines can serve the stockpiles in a given row. Further, when a machine is in place and serving a stockpile in a row, it can restrict access

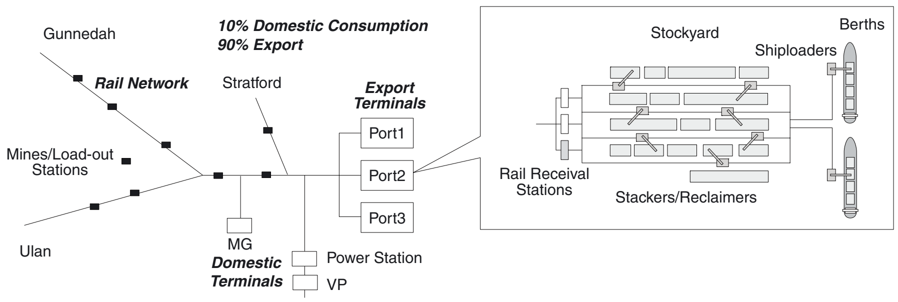

FIGURE 2. A schematic of coal chain [2].

to other stockpiles in that row, or adjacent rows. We need to schedule access to the stockpiles in the rows so that delays arising from machine conflicts between the dump stations, stackers and reclaimers are minimised.

As part of this process, trains scheduled to bring coal from the mines have nominated dumpers and stackers at the terminal to move the coal onto the stockpiles. There will be a number of valid dumper-stacker combinations for an arriving train and, where possible, the schedulers will try and assign a *preferred* dumper-stacker combination for the unloading.

1.3. Using fixed train paths on the rail network

The rail network used by the Hunter Valley Coal Chain for coal export is also used for domestic coal deliveries, other freight services, and passenger trains.

The rail operators with responsibility for managing and optimising the overall network provide a number of train paths assigned for export coal movement to the HVCCC. The train paths provide time slots on the network at different times based on train speeds, required train separation distances, and other operational factors.

The paths start or end at a port at a fixed time. A *forward* path starts at a port at a specified time and provides a slot for a train to travel all the way up the valley. A *return* path provides a slot for a train to travel all the way down the valley, finishing at a port at a specified time. In this way, a path can be used to move a train to/from any mine that is serviced by the path. Each roundtrip "consumes" one forward and return path. Paths going to further mines such as Ulan are less frequent than to closer mines in the Hunter Valley.

The mine load points have more flexible arrival and departure times. The HVCCC schedulers specify: (a) the load point and time at which a train travelling on a particular forward path will exit the path and leave the main network; and (b) the load point and time from which the train will enter a particular return path so as to arrive back at the port at the port arrival time specified for the path.

A train is not on the main rail network while it is loading at a mine or unloading at port, so it will not affect other trains on the network during these operations.

The requirement that we use set paths at set times makes this quite a different problem to most rail scheduling problems in the literature. We are not required to consider any of the track sections between the port and the load points because the set of paths provided to the HVCCC is guaranteed to prevent track conflicts.

If this was the entirety of the problem then we would have a train assignment problem (trains to brands) combined with a path assignment problem. However we must still consider conflicts at the ports and load points when developing a schedule using the available paths.

1.4. Background literature

The rather extensive train scheduling literature is largely focussed on situations in which the scheduler has full control of the rail infrastructure and schedules are developed to optimise factors such as freight or passenger throughput, or the utilisation of trains and other infrastructure.

As noted, the requirement to use existing paths with specified arrival and departure times makes this rail scheduling problem different to most of those discussed in the literature. However, those problems provide the background to the path utilisation model considered in this paper, so we outline approaches to some of them here.

Cordeau *et al.* [3] provide a survey of common rail transportation problems. They propose a classification of models and associated algorithms for each problem group. The review focusses on routing and scheduling problems. Routing models concern the operating policies for freight transportation and rail fleet management. Scheduling models address the dispatching of trains and the assignment of locomotives to carriages or wagons. Lusby *et al.* [4] give a more recent review of models and methods used to assist planners in finding train routes. The authors survey work on train timetabling, train dispatching, train platforming, and train routing problems and discuss track allocation from a strategic, tactical, and operational level.

Salido *et al.* [5] describe a railway capacity model for solving periodic train scheduling. The model uses topological and rail schedule data to set up a constraint optimization problem. The variables are train frequencies and the arrival and departure times of trains at stations. The constraints relate to user requirements, the railway infrastructure and rules for rail traffic coordination. In this case topology refers to factors such as stations, tracks, distances between stations, and traffic control features. The schedules relate to factors such as the arrival, departure and stopping times of trains at the stations, junctions and crossings in the network.

Ahuja *et al.* [6] consider the locomotive-scheduling (or assignment) problem in which a consist, or group of locomotives, must be assigned to each train in a preplanned train schedule so as to provide each train with sufficient locomotive power to pull the train from its origin to its destination. The authors describe a mixed integer programming (MIP) formulation of the locomotive-assignment problem.

The MIP formulation results in very large problems that cannot be solved to optimality or near optimality in acceptable running times using commercially available software. The authors use a mix of problem decomposition, integer programming, and very large-scale neighborhood search, to develop a solution technique to solve the problem in shorter (around 30 min) times.

D'Ariano *et al.* [7] study a real-time rail traffic management system for helping controllers to manage disruptions to schedules. The authors use a branch-and-bound algorithm for sequencing train movements and setting routes to ensure smooth train behaviour and limit train delays. A local search algorithm is used to try and find an optimal rerouting for trains affected by schedule disruptions.

Şahin *et al.* [8] describe an integer programming formulation, based on a space-time network, for the train dispatching problem. The goal is to determine detailed timetables over a rail network in order to minimize deviations from a planned schedule consisting of the departure and arrival times of each train in each track section of its route. The authors propose heuristic algorithms that can be applied to solve problems with constraints related to freight rail networks. The heuristics are able to find better solutions than those that can be obtained by a commercial solver in much longer times for a real-world problem of a congested freight railroad.

Coal Supply Chains, being economically significant and challenging to plan and operate, are the subject of much study.

Liu and Kozan [9] propose a metaheuristic method, based on Best-Insertion and Tabu Search algorithms, to optimise a coal rail network in Australia. The rail network consisted of single and double track sections with sidings (crossing loops) to allow trains to overtake or pass on long single tack sections of the network. The coal train scheduling system is modelled as a blocking parallel-machine job-shop scheduling problem with some minor constraints. The authors include a case study showing the use of their method to optimise the train schedules in a network based on a complex real-world coal rail system in Australia.

A paper by Lusby *et al.* [10] is more closely related to the coal chain rail scheduling problem presented here. The authors use a set-packing approach to the problem of routing trains through railway junctions. The goal is to find a set of nonconflicting train paths that will allow several hundred trains per day to travel through a number of junctions. A set of train paths is termed nonconflicting if no two trains attempt to claim the same piece of track simultaneously.

The requirement that at most one train can claiming any track section at any given time is observed using a constraint for each track section in a sequence of time intervals. These constraints are represented by two matrices:

(1) One containing rows for each time interval for each track section and columns for each path. The elements in this matrix indicate whether a particular time interval on a section of track is part of a given path.
(2) Another containing rows for each train and, again, columns for each path. A value of 1 in this matrix indicates whether a particular column is a path for a particular train.

Specified parameters are used to denote the benefit received from assigning a particular path to the solution and a binary decision variable is used to govern the inclusion a path in the solution. The objective function in the formulation maximizes the total benefit in routing the trains. Two sets of constraints are used to ensure that all trains receive a path and that at most one train can claim any track section during each time interval.

As noted above, the operators of the Hunter Valley rail network provide a number of rail paths through the network at different times. So our scheduling problem, of assigning trains to these paths, has similarities to the set-packing problem described above. However, our problem is distinctive in that some scheduling of port infrastructure is required. As such this problem can also be related to shared-resource machine scheduling.

There are also several papers that have dealt with aspects of planning and scheduling specifically in the Hunter Valley Coal Chain. Boland and Savelsbergh [1] provide an overview of the various optimisation problems that arise in this application. While Welgama and Oyston [11] use a simulation study to look at ways to increase the throughput of the supply chain.

Singh *et al.* [2] provide a broad description of the operations of the Hunter Valley Supply Chain and its planned expansion. This paper looks at how to optimally expand the infrastructure in the supply chain to meet expected increases in future demand. The paper looks at train scheduling but only at a level sufficient to estimate the capacity using different infrastructure configurations. The train capacities are expressed in terms of the number of trains or tonnages per day that can be processed by various parts of the train network.

2. Rail scheduling on the Hunter Valley Coal chain

Ship arrivals at the terminals provide orders for coal. All the coal for the orders on a given ship must be stockpiled at the terminal before a ship can berth for loading. The coal producers specify which mines will provide the coal components that will be blended to build the coal brands in the stockpiles. Trains must be scheduled to bring the coal from those mines to the terminals where nominated dumpers and stackers are used to move the coal from the trains onto the stockpiles.

The rail schedulers construct daily train schedules to determine which trains are to be sent to the mines, using the available train paths, so that their arrivals at the terminals best meet the planned dumping and stacking operations. The train trips are subject to a number of system constraints such as load point capacity limits, turn-around and refueling delays, and train size limits for individual mines.

In the next few sections we describe the key components of the model, including some definitions, constraints and objectives.

2.1. Roundtrips

The railing model described in this paper uses the idea of a train "roundtrip" as the basis for scheduling the coal deliveries.

A roundtrip is a train journey starting and finishing at a parking area[2] on the rail network. On a roundtrip, a train travels out to a mine on a forward train path, loads with coal at the mine's loading point, may wait for an appropriate return train path, travels back to the terminal on a return train path, and finally goes back to the parking area where crew changes and maintenance may be carried out. Each roundtrip has a value determined by the amount of coal delivered, the idle time spent waiting for access to resources, and the importance of the delivery of different coal types to meet shipping demand.

In what follows, a roundtrip is specified by the following attributes:

- a component(coal type at a mine) that will be used to build the brand in a stockpile,
- the train used for the journey,
- the forward train path,
- the mine load point,
- the idle time spent waiting at the load point,
- the return path,
- the dump station selected, and
- the stacker used to stack the coal onto the nominated stockpile.

The attributes are subject to a number of hard constraints that must be observed when constructing roundtrips:

(1) Every train is run by its operator and the component specifies which operator's train is to be used for a particular roundtrip.
(2) The component also specifies the mine that is supplying the coal and the stockpile where coal will be stored. This in turn specifies the load point for the roundtrip and restricts the dumpers and stackers that can be assigned to that roundtrip. In particular:
    (a) The capacity limit of a load point specifies a set of possible train sizes.
    (b) Because not every stacker can access every stockpile, the dumper and stacker combination assigned to a roundtrip must be valid for the stockpile.
(3) Only forward and return paths that match the junction of the load point in the roundtrip can be used.
(4) The start time of the return train path for the roundtrip cannot occur before the train has finished travelling to the mine and loading at the loadpoint. The time to load a train depends on its size and the capacity of the loadpoint.
(5) The time between the arrival at the load point and the departure of the train must be within a given time window. This is both a practical restriction on the amount of idle time for a train and a way to limit the problem size. Changing the maximum idle time at the loadpoint can significantly increase or decrease the number of possible roundtrips.
(6) An appropriate amount of time must be allowed for crew changeover between two subsequent journeys of a train. This is simply modelled using an enforced "idle" time at the end of a roundtrip.

[2]Similar to a marshalling yard.

(7) Trains, loadpoints, dumpers and stackers cannot be used during their pre-planned maintenance periods or any other activities. In general we talk about *outages*, though some of these might for example be generated implicitly by pre-assigned train trips that the schedulers want to fix as input to the optimisation.

Note that trains are not to sit idle anywhere in their roundtrip except after loading at the loadpoint. It is theoretically possible for trains to wait in other locations, for example just before the dump stations at the terminals. However there is very little space for trains to wait in these places and idle time in these locations is not allowed at the planning stage, in order to prevent congestion during the implementation of the schedules where some idle time may occur due to delays or other unforseen circumstances.

We also need to try and best meet the following soft constraints:

(1) Among all the valid dumper-stacker combinations, where possible use the preferred dumper-stacker combination.
(2) Components with an earlier delivery date, based on ships arrival time, have a higher priority or weight.

The choices associated with setting up a roundtrip are discrete, except for the idle time spent at the loadpoints. We discretise this time period into one minute intervals. We also note that unlimited idle time would result in an infinite number of possible roundtrips, so a maximum idle time limit is required. The size (and thus complexity) of the model and the quality of the solutions produced is highly dependent on this choice of maximum idle time. We will defer detailed discussion of this point until the results section.

We model the scheduling problem by generating the set of all possible roundtrips and then searching for a subset which maximises the value of the schedule with no resource conflicts.

2.2. Resources

There are a number of resources that are "consumed" by a roundtrip. The selection of an optimal set of roundtrips is largely constrained by the availability of these resources. The roundtrip attributes and associated resources considered here are:

(1) The forward and return paths. Each path can only be assigned to at most one roundtrip.
(2) Dumpers, stackers and load points can service at most one train at a time.
(3) At most one train for a particular stockpile can be unloaded at a given time.

Since most of these resource constraints, with the exception of the path assignment, apply over the whole time horizon we need to create a set of discrete constraints. This is done by considering for each roundtrip the fixed interval over which it is using a resource and then determining the maximum clique sets of this interval graph. For each such clique a constraint is created that at most one of the corresponding roundtrips can be selected. The case where the resource is unavailable due to existing activities or maintenances is already dealt with by not generating conflicting roundtrips.

2.3. Tonnages

Each component represents a demand for a certain number of tonnes to be shipped from a particular mine to a defined stockpile at the terminal. This demand may require multiple train trips to be satisfied and can be completed by one or more combinations of the trains. For example, a component with total tonnage as 10 000 tonnes can be scheduled by two 5000 tonne trains or one 7000 and one 3000 tonne train. In general the tonnages provided as input to the model are chosen so that the they can be satisfied exactly with an integer number of trainloads. However the model allows some of the tonnages to not be satisfied and simply maximises the amount of coal scheduled for delivery to the port.

3. The rail scheduling model

In the next few subsections we present a mathematical model for the rail scheduling problem. We first present the notation used in the formulation, and then the objective function and constraints for the model.

3.1. Notation

Sets

I	The set of all roundtrips. A roundtrip $i \in I$ is constructed with respect to the attribute requirements specified by the shipping orders and uses resources subject to the constraints associated with their availability during the schedule planning horizon.
C	The set of coal components associated with the order for a particular brand of coal on a ship. A roundtrip i will supply (part of the) coal for a specific order component $c \in C$.
R	The set of resources, trains, forward and return paths, load points, dump stations and stackers. A roundtrip i will use particular resources $r \in R$.
T	The set of critical time intervals, t, into which the planning horizon is divided. In general the set of such time intervals, corresponding to times at which a maximum number of roundtrips use the same resource in the interval graph, are resource dependent. However for notational convenience we have dropped the resource subscript and describe the formulation as if all resources used the same (super)set of time intervals.
$I_{r,t}$	The set of all roundtrips $i \in I$ that use resource $r \in R$ at time $t \in T$.
I_c	The set of all roundtrips $i \in I$ such that roundtrip i contributes towards component $c \in C$.

Parameters

d_c The demand (tonnes) for order component c.
u_i The tonnes of coal for component c delivered by roundtrip i.
v_i The value of roundtrip i. The overall value is made up from a number of benefits related to roundtrip attributes such as coal tonnes delivered, idle time, and preferred dumper/stacker allocations.

Decision variables

x_i Whether roundtrip i is part of the scheduling solution.

3.2. The rail scheduling model formulation

We wish to select a subset of the roundtrips, such that the sum of the values is maximised without any resource conflicts. If we create a discrete set of times when the resource conflicts may occur (cliques), then this is an instance of the set packing problem. However we must also limit the amount of coal delivered for any one component; the order must not be over-delivered. So we require that the total tonnes delivered for a component not exceed the amount requested. Unlike the set-packing constraints these constraints do not have 0 or 1 coefficients, indeed because of the varying sizes of trains they become fractional.

Since each possible roundtrip adds a variable to the model, this formulation gives a very large problem size for the real world instances we are trying to solve. Artificially adding restrictions on the idle time of trains makes this number of variables more reasonable, but still in the millions.

$$\text{Maximise} \quad \sum_i v_i x_i \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i \in I_{r,t}} x_i \le 1 \qquad \forall\, r \in R,\ t \in T \tag{3.2}$$

$$\sum_{i \in I_c} u_i x_i \le d_c \qquad \forall\, c \in C \tag{3.3}$$

$$x_i \in \{0,1\} \quad \forall\, i \in I \tag{3.4}$$

The objective function (3.1) maximises the total utility value associated with the roundtrip attributes. The individual utility values are described in Section 3.3 below.

Constraint (3.2) says that a resource r can be used by only one roundtrip in time interval t. That is, we cannot have any resource conflicts.

Constraint (3.3) says that the total tonnes railed by roundtrips carrying coal for component c cannot exceed the total demand for that component.

Constraint (3.4) says that round trip i is either in the schedule or not.

3.3. Scheduling objectives

Our goal is to find the "best possible" schedule, so we need a way of comparing schedules. This is done by assigning a benefit (or utility) to various activities within the schedule and maximising the total utility value, that is, the sum of the individual benefits. Costs or penalties associated with activities have negative utility values.

The overall objective contains a number of sub-objectives. We want to:

(1) Maximize the throughput, that is, the amount of coal delivered to the terminal.
(2) Minimize the deviation from the preferred dumper-stacker combinations.
(3) Minimize the idle time
(4) Use the "correct" or preferred train size for coal pick-up from each of the mines.

Each of these objectives is assigned a weight to reflect its relative importance.

3.3.1. Throughput (sub-objective 1)

The most important aim is to schedule as much of the component demand as possible. The throughput contribution to the roundtrip value is based on the tonnage delivered by the trip and is given by:

$$v_i^1 = \frac{u_i}{\mathsf{MaxTrainSize}} \tag{3.5}$$

where MaxTrainSize is the maximum train size in the supply chain. Provided the longer term planning has been done well, it should be possible to deliver all or most of the coal, so that the sub-objectives below can be used to discriminate between otherwise equivalent maximal solutions.

3.3.2. Preferred dumper-stacker combinations (sub-objective 2)

The stockyard managers assign an area in one of the stockyard rows to build the stockpile for a particular order. The managers also assign a preferred dumper and stacker, based on planned yard operations, to be used to build the stockpile.

There may be a number of possible dumper-stacker combinations that could be used to build the stockpile. Using non-preferred combinations is physically possible but may increase the likelihood of congestion and other scheduling issues in the stockyard. The value of a given combination is given by

$$v_i^2 = \mathsf{DumperStackerPriority} \tag{3.6}$$

where DumperStackerPriority has a value of: 1.0 if both the preferred dumper and stacker are assigned to the roundtrip, 0.5 if one of them is assigned, and zero if neither is assigned.

3.3.3. Idle time penalty (sub-objective 3)

A train is considered to be idle when it is waiting in a load point loop after loading. The idle time is the number of minutes during which a train is idle and the idle time penalty is calculated as:

$$v_i^3 = -\frac{\mathsf{Idle\ Time}}{\mathsf{MinutesPerDay}}. \tag{3.7}$$

That is, we arbitrarily scale the idle time for a given roundtrip to be some fraction of a day, and set the corresponding objective term to be negative because we want to minimise the idle time.

3.3.4. Preferred train size (sub-objective 4)

This objective is mainly used as a guide to selecting good roundtrips for the schedule. As mentioned above, the demand specified for each loadpoint comes from a separate planning process. The human planners generally select amounts that are multiples of a particular train size, indicating that this is the preferred train size to be used for meeting the demand. Furthermore, if any trains of a different size are scheduled to collect coal for a particular shipment then it may be difficult or impossible to exactly meet the remaining demand with the available train sizes. Hence we include the additional objective:

$$v_i^4 = \begin{cases} 1 & \textit{if the demand is a multiple of the train size} \\ 0 & \textit{otherwise.} \end{cases} \tag{3.8}$$

3.3.5. Summary of objective terms

The total objective coefficient, v_i, for roundtrip i is calculated as

$$v_i = \sum_j W^j v_i^j \tag{3.9}$$

where j indexes the sub-objective terms described in Sections 3.3.1–3.3.4 above. The planners set the weights W^j for each sub-objective to reflecting its relative importance when generating a schedule over a particular time horizon.

4. A Lagrangian relaxation based algorithm

In this section, we present a Lagrangian relaxation based algorithm to solve the model presented above. The formulation of our problem contains large number of variables and relatively fewer number of constraints, such problems can also be solved by Column Generation Techniques [12, 13]. But there were several reasons to approach this problem with Lagrangian relaxation technique. Firstly, Column Generation techniques can have very slow convergence [14]. Secondly, by using Lagrangian relaxation technique, as presented here, the client did not need to invest in expensive solvers like CPLEX or Gurobi. Finally, for practical problems of this complexity it is more desirable to find close to optimal solutions quickly than to find optimal solutions and as our results will show we do manage to achieve this.

Lagrangian relaxation is a well known and widely used method for the decomposition of large problem. For a detailed discussion on Lagrangian relaxation based algorithms the reader is referred to [15–18].

We relax all the constraints in our model *i.e.* constraints (3.2) and (3.3), by using Lagrangian relaxation techniques and introduce two sets of Lagrange multipliers, one for each type of constraint:

(1) $\{\lambda_{r,t} \;\; \forall r \in R,\; t \in T\}$
(2) $\{\mu_c \;\; \forall c \in C\}$

The Lagrangian Dual can then be obtained as:

$$\begin{aligned} L(\lambda, \mu) = \max_x \Bigg\{ & \sum_{i \in I} v_i x_i + \sum_{r \in R} \sum_{t \in T} \lambda_{r,t} \left(1 - \sum_{i \,\in\, I_{r,t}} x_i \right) \\ & + \sum_{c \in C} \mu_c \left(d_c - \sum_{i \,\in\, I_c} u_i x_i \right) \Bigg\} \qquad (4.1) \\ = \max_x \Bigg\{ & \sum_{i \in I} \left(v_i - \sum_{r \in R} \sum_{t \in T} \lambda^i_{r,t} - \sum_{c \in C} \mu^i_c u_i \right) x_i \Bigg\} \\ & + \sum_{r \in R} \sum_{t \in T} \lambda_{r,t} + \sum_{c \in C} \mu_c d_c, \qquad (4.2) \end{aligned}$$

where $\mu^i_c = \mu_c$ if $i \in I_c$ and 0 otherwise, and $\lambda^i_{r,t} = \lambda_{r,t}$ if $i \in I_{r,t}$ and 0 otherwise.

As this reduces our problem to an unconstrained optimisation problem, it is easy to find optimal solution for $L(\lambda, \mu)$. Clearly, if $\{x^*_i\}$ is an optimal solution for $L(\lambda, \mu)$, then

$$x^*_i = \begin{cases} 1 \text{ if } v_i - \sum_{r \in R} \sum_{t \in T} \lambda^i_{r,t} - \sum_{c \in C} \mu^i_c u_i \geq 0 \\ 0 \text{ otherwise.} \end{cases} \qquad (4.3)$$

Starting with all Lagrangian multipliers as zero we iteratively update the multipliers and therefore an upper bound on the problem is also obtained.

Lagrangian relaxation methods have been strengthened by many approaches and one such approach is the "Volume Algorithm" [19]. This algorithm incorporates in each iteration the direction of movement as a convex combination of the current and previous subgradients. The main idea is to accelerate the solution's convergence by avoiding the usual zig-zag behavior of the subgradient algorithm. We used the COIN-OR[3] implementation of the Volume algorithm to update the Lagrange multipliers.

[3] http://www.coin-or.org/.

In each iteration we use a greedy heuristic to find feasible integer solutions, thus providing a lower bound on the original problem. The algorithm continues until the gap between lower and upper bound is within acceptable limits or a specified time limit is reached. The details of this implementation are given in the pseudocode description (Algorithm 1).

```
Algorithm: Lagrangian Relaxation based Algorithm
$N \leftarrow 0$
while Time limit not reached or gap is < acceptableGap do
    /* The $volumeHeuristic()$ gives an upper bound and calculates
       $r.relaxedSoln$ & $r.reducedCost$ for all roundtrips $r$.          */
    $volumeHeuristic()$
    $N \leftarrow N + 1$
    for $i \leftarrow 1$ to $\min\{20, N\}$ do
        $S \leftarrow \emptyset$
        foreach Roundtrip $r$ do
            if $(r.relaxedSoln + r.timesUsed/N > 0.05)$ or $(i \leq 3$ and $N \geq 50$ and
            $r.reducedCost < -0.1)$ then
                $S.\text{add}(r)$
                $r.weight \leftarrow r.relaxedSoln + r.timesUsed/N$
            end
        end
        $S \leftarrow randomGreedy(S)$
        if $value(S) > value(H)$ then $H \leftarrow S$
    end
    foreach $r \notin H$ do
        if $H \cup \{r\}$ is feasible then $H \leftarrow H \cup \{r\}$
    end
    if $value(H) > value(H')$ then $H' \leftarrow H$ /* Count how often $r$ is used in
       an iteration's best solution. Used to select subsets for the
       greedy heuristic.                                                  */
    foreach $r \in H$ do $r.timesUsed \leftarrow r.timesUsed + 1$
end
```

Algorithm 1: Pseudocode for the Lagrangian Relaxation based heuristic.

4.1. A greedy heuristic

The greedy heuristic considers a subset of the roundtrips. These are weighted and randomly selected, one at a time, to become part of the solution. Roundtrips which are incompatible with those already selected are removed from the subset. The heuristic continues to select roundtrips randomly until none, that can be feasibly added as a part of the solution, remain from the original subset. The formula used to select the subset and weight the various roundtrips is:

$$p_i = x_i^R + \widehat{x}_i$$

TABLE 1. Attributes of the datasets.

	Dataset A	Dataset B	Dataset C
Components	49	48	49
Forward Paths	391	435	391
Return Paths	462	527	462
Avg. train size (tonnes)	6619.3	6441.9	6619.3
Demand (tonnes)	862 200	855 000	862 200

where x_i^R is the value for x_i from the most recent iteration of the Volume algorithm and $\widehat{x}_i$ is the proportion of heuristic solutions so far which contain roundtrip i (let $\widehat{x}_i = 0$ in the first iteration). A roundtrip is part of the subset considered by the heuristic if $p_i > 0.05$ and is chosen with probability $p_i/\sum p_i$.

After fifty iterations of the Volume algorithm, a small proportion of runs of the heuristic also include, in the subset, roundtrips with reduced costs less than -0.1. This is done to ensure we do not miss including roundtrips that could be scheduled but have an approximately zero reduced cost.

The greedy heuristic is run multiple times for each iteration of the Volume algorithm. As a final step, the algorithm greedily adds any remaining feasible roundtrips (necessarily missing from the subset considered originally) to the best solution. This ensures the heuristic solution is maximal in the sense that it is not possible to add any further roundtrips without dropping some other trips.

```
Algorithm: randomGreedy($S$)
$H \leftarrow \emptyset$
while $S \neq \emptyset$ do
    Select an $s \in S$ with probability $P = \frac{s.weight}{\sum r.weight}$
    $H.add(s)$
    Remove all infeasible roundtrips from $S$ that would clash with $s$.
end
return $H$
```

Algorithm 2: Pseudocode for the greedy heuristic.

5. Computational results

In this section we present results from computational experiments using three different datasets from the Hunter Valley Coal Chain. For each of the datasets we also used two different settings, of 2 and 5 h, for the maximum waiting time at any loadpoint. This gave us a total of 6 different datasets. Table 1 provides an overview of each of the datasets.

Figure 3 shows the distribution of demand from different loadpoints in the three datasets. Each dataset contained 28 loadpoints and 31 trains (of various sizes).

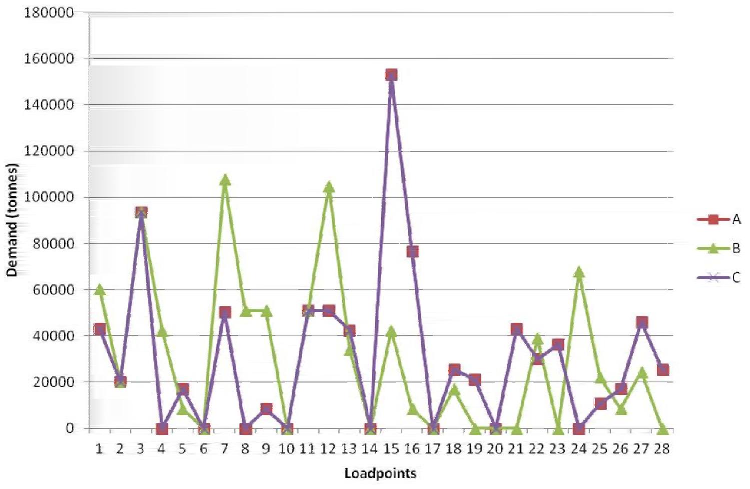

FIGURE 3. Demand in tonnes across all loadpoints.

TABLE 2. Weights of each sub-objective as described in Sections 3.3.1–3.3.4.

	W^1	W^2	W^3	W^4
Dataset A	1.0	0.5	0.5	0.2
Dataset B	0.4	0.5	0.5	0.2
Dataset C	0.4	0.5	0.5	0.2

Datasets A and B are significantly different, while Dataset C differs from A only in the availability of equipment and the use of different objective weights, with A having a larger throughput weight of $W^1 = 1.0$ while C (and B) use a weight of $W^1 = 0.4$. Table 2 provides all the objective weights used in the individual datasets.

As can be seen from Table 3, the integer programming formulations for these instances are very large even when allowing just two hours of idle time at the loadpoints. In particular having over a hundred million non-zero means that the MILP solvers require very large amounts of memory. For our tests we used a Linux server with four quad-core Intel Xeon CPUs running at 2.93 GHz (for a total of 16 cores) and approximately 100 Gb of RAM. Yet even with such a large machine the commercial solvers struggled to get to any solution.

We applied three different methods to the problem instance:

(1) **Lag.Heur:** The Lagrangian heuristic combining a volume algorithm with a greedy repair method as presented in Section 4.
(2) **Cplex:** CPLEX 12.5 was used to solve the ILP presented in Section 3.2.
(3) **Gurobi:** GUROBI 5.0 solving the same formulation.

TABLE 3. Size of the models (at 2 h idle time)).

	Rows	Cols	Non–zeros
Dataset A	10 845	2 114 217	129 882 227
Dataset B	13 874	4 783 371	338 651 602
Dataset C	10 918	2 134 562	131 466 242

TABLE 4. Best integer solution – 2 h idle time.

	1 h runtime			10 h runtime		
	A	B	C	A	B	C
Lag.Heur.	240.94	177.52	183.09	240.94	178.70	184.91
Cplex	0.0	*none*	0.0	227.9	0.0	0.0
Gurobi	231.1	157.0	177.0	255.9	157.0	196.4

TABLE 5. Best upper bound – 2 h idle time.

	1 h runtime			10 h runtime		
	A	B	C	A	B	C
Lag.Heur.	272.807	329.866	215.31	259.203	191.7	199.109
Cplex	∞	∞	∞	258.9	∞	∞
Gurobi	∞	∞	∞	258.9	∞	198.9

Each method was executed for 1 h and 10 h of wall-clock time. Since the computer used for the experiments has 16 cores, this in principle allowed the MILP solvers to use up to 160 h of CPU time, though in practice generally they would use significantly less. Tables 4–6 provide a summary of the results.

As can be seen in Table 5, CPLEX was not able to make any progress with datasets B or C; it found a solution to dataset A only after a long wall time and was beaten by both Gurobi and the heuristic. It appears that Gurobi makes more use of heuristics and was thus able to find feasible non-zero solutions within an hour. Gurobi managed to find good upper bounds for datasets A and C within 10 h, but still had not solved the root node for the larger dataset B.

The Lagrangian heuristic on the other hand was very robust and able to produce acceptable quality solutions even within the one hour time limit. Comparing the heuristic to the commercial solver solutions with the 10 h runs it is clear that Gurobi managed to produce marginally better quality integer solutions, though Cplex did not. It should be noted however that our Lagrangian heuristic was not using the multiple cores of the computer, so overall was using less CPU time. Also, as could be expected, the relaxed upper bounds are not quite as tight for the Lagrangian methods as for the linear programming method.

It is interesting to note from Table 5 and Figure 4 that, for the heuristic, longer cpu run times lead to more significant improvements in the relaxed upper bound than in the solutions. This is desirable as it shows that good quality solutions are already found after an acceptable amount of run time.

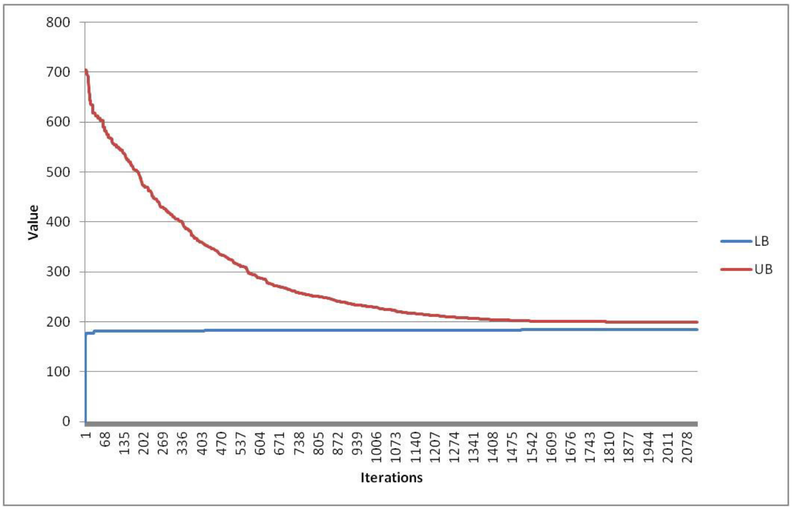

FIGURE 4. Gap between Upper and Lower bounds for dataset C in 10hr run of the Lagrangian heuristic.

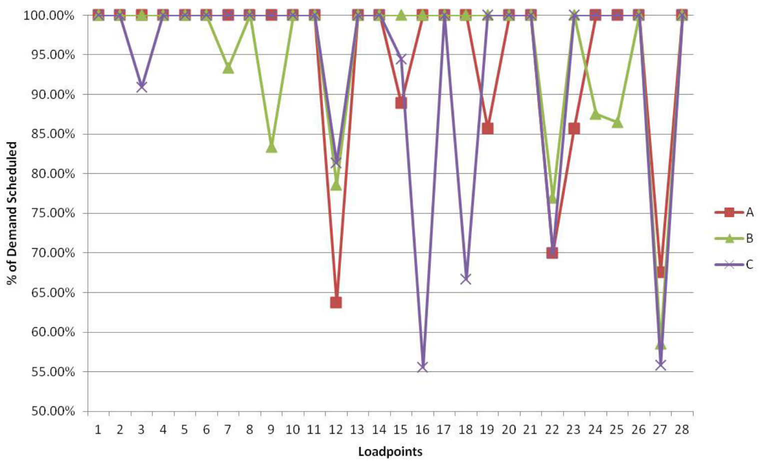

FIGURE 5. Percentage of the loadpoint demand that has been supplied by the best solution found with the Lagrangian heuristic within one hour.

The reasonableness of the solution in meeting practical needs can also be seen in Figure 5. This shows that, for most of the loadpoints, in all three instances all of the demand was met. And, with just two exceptions, the remaining mine loadpoints have at least 60% of the required tonnage scheduled for railing.

Finally we indicate briefly the effect of increasing the amount of allowed idle time at the loadpoints. Table 6 shows the solutions from the Lagrangian heuristics

TABLE 6. Best solutions found with the Lagrangian heuristic when allowing 5 h of idle time at the loadpoints.

	1 h runtime			10 h runtime		
	A	B	C	A	B	C
Upper Bound	426.036	450.56	301.312	262.032	191.781	201.334
Objective (LB)	242.851	174.974	187.797	244.346	177.105	189.74

TABLE 7. Weights of each sub-objective as described in Sections 3.3.1–3.3.4.

	W^1	W^2	W^3	W^4
Dataset D	5	0.5	0.5	0.2
Dataset E	1	0.5	0.0	0.2

TABLE 8. Summary of results with 2 h idle time and 10 h cpu.

	Dataset C	Dataset D	Dataset E
Roundtrips	126	126	130
Tons scheduled	778 500	796 600	806 400
Best integer (obj value)	184.91	612.75	243.87
Sub-objective 1	91.6	93.7	94.9
Sub-objective 2	126.0	126.0	130.0
Sub-objective 3	−1.5	−1.7	−4.6
Sub-objective 4	86	82	84

with the increased set of roundtrips generated by allowing additional idle time at the mines. It should be noted that these instances were too large to attempt with Gurobi or Cplex. Comparing the results to those in Table 4 shows that relaxing the constraints in this manner produced slightly improved solutions within a one hour runtime limit. However the gains are not very large. Furthermore, the greater size and complexity of the problems means that within the 10 h time limit we are not able to find quite as good solutions as when restricting ourselves to just two hours of idle time. This is an indication that the Lagrangian heuristic had still not converged even after 10 h of runtime for these larger instances.

In order to analyse the performance of our algorithm on different objective weights, we created two more datasets with identical demand as Dataset C but weights as per Table 7.

As the results in Tables 8 and 9 show, the performance of the algorithm changes as expected with the change of weights. Increasing the weight of sub-objective 1, which is directly related to the amount of coal transported by our algorithm, we get more tons scheduled. Having a positive weight on sub-objective 3 is more desirable as it reduces the amount of idle time on trains without much sacrifice on the tons scheduled.

TABLE 9. Summary of results with 5 h idle time and 10 h cpu.

	Dataset C	Dataset D	Dataset E
Roundtrips	128	128	128
Tons scheduled	799 700	821 300	818 000
Best integer (obj value)	189.74	634.31	249.24
Sub-objective 1	94.1	96.6	96.2
Sub-objective 2	128.0	128.0	128.0
Sub-objective 3	−3.8	−3.6	−8.5
Sub-objective 4	90	89	89

6. Conclusion

In this paper we have described a real world rail scheduling problem that has some interesting characteristics due to the need to fit train trips into given time slots or paths on the main line. These problems can be formulated as very large integer linear programs that are too large to be amenable to solution using commercial solvers such as Gurobi or Cplex. However using the Lagrangian heuristic described in this paper it is possible to get good schedules and also useful relaxed bounds in an acceptable amount of computing time.

Future research will look at improving the convergence of the Lagrangian heuristic and extensions of the problem. This include additional detail encountered in practice such as dealing with maintenance and more complicated constraints on train occupancy of the mines' loadpoint loops. In addition it is also of interest to consider whether multi-core computers can be used to solve these kinds of problems more efficiently in parallel. For example, running different versions of the greedy heuristic in parallel.

References

[1] N. Boland and M. Savelsbergh, Optimizing the hunter valley coal chain. In H. Gurani and S. Ray Mehrotra (eds.), *Supply Chain Disruptions: Theory and Practice of Managing Risk*. Springer London, 2012, pp. 275–302.

[2] G. Singh, D. Sier, A. Ernst, O. Gavriliouk, R. Oyston, T. Giles and P. Welgama, A mixed integer programming model for long term capacity expansion planning: A case study from The Hunter Valley Coal Chain. *Eur. J. Oper. Res.* **220** (2012) 210–224.

[3] J.-F. Cordeau, P. Toth and D. Vigo, A Survey of Optimization Models for Train Routing and Scheduling. *Trans. Sci.* **32** (1998) 380–404.

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

[5] M.A. Salido, F. Barber, M. Abril, P. Tormos, A. Lova and L. Ingolotti, A Topological Model Based on Railway Capacity to Manage Periodic Train Scheduling, in *Applications and Innovations in Intelligent Systems XII, Proceedings of AI-2004, the Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, edited by A. Macintosh, R. Ellis, and T. Allen. Springer, London, 2005, Vol. 2, pp. 107–120.

[6] R.K Ahuja, J. Liu, J.B. Orlin, D. Sharma and L.A. Shughart, Solving real-life locomotive-scheduling problems. *Trans. Sci.* **39** (2005) 503–517.

[7] A. D'Ariano, F. Corman, D. Pacciarelli and M. Pranzo, Reordering and local rerouting strategies to manage train traffic in real time. *Trans. Sci.* **42** (2008) 405–419.

[8] G. Şahin, R.K. Ahuja and C.B. Cunha, Integer programming based solution approaches for the train dispatching problem. Technical report, Sabanci University, 2010.

[9] S.Q. Liu and E. Kozan, Optimising a coal rail network under capacity constraints. *Flexible Services and Manufacturing Journal* **23** (2011) 90–110.

[10] R. Lusby, J. Larsen, D. Ryan and M. Ehrgott, Routing trains through railway junctions: a new set-packing approach. *Trans. Sci.* **45** (2011) 228–245.

[11] P.S. Welgama and R. Oyston, Study of options to increase the throughput of the hunter valley coal chain, in Proc. of MODSIM (2003) 14–17.

[12] A. Capone, G. Carello, I. Filippini, S. Gualandi and F. Malucelli, Solving a resource allocation problem in wireless mesh networks: A comparison between a CP-based and a classical column generation. *Networks* **55** (2010) 221–233.

[13] E. Choi and D.-W. Tcha, A column generation approach to the heterogeneous fleet vehicle routing problem. *Comput. Oper. Res.* **34** (2007) 2080–2095.

[14] H.M.T. Ben Amor, J. Desrosiers and A. Frangioni, On the choice of explicit stabilizing terms in column generation. *Discrete Appl. Math.* **157** (2009) 1167–1184.

[15] L.A. Wolsey and G.L. Nemhauser, Integer and combinatorial optimization. Wiley-Interscience.

[16] G. Singh and A.T. Ernst, Resource constraint scheduling with a fractional shared resource. *Oper. Res. Lett.* **39** (2011) 363–368.

[17] A. Thomas, G. Singh, M. Krishnamoorthy and J. Venkateswaran, Distributed optimisation method for multi-resource constrained scheduling in coal supply chains. *Int. J. Prod. Res.* **51** (2013) 2740–2759.

[18] M.L. Fisher, The lagrangian relaxation method for solving integer programming problems. *Manag. Sci.* **50** (2004) 1861–1871.

[19] F. Barahona and R. Anbil, The volume algorithm: producing primal solutions with a subgradient method. *Math. Program.* **87** (2000) 385–399.