Dissertation for the degree of Master of Science
University of the Witwatersrand
Johannesburg

Dynamic bulk freight train scheduling in an uncongested rail network

Robert Andrew Bennetto

April 2013

Supervisor: Prof. D. Lubinsky

Declaration

I declare that this dissertation is my own unaided work. It is being submitted for the degree of Master of Science at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

[illegible]

R.A. Bennetto
25 April, 2013

Abstract

Many academic works in the train scheduling environment concentrate on optimizing movements of resources through the physical network. To optimize bulk freight lines, algorithms must provide a feasible schedule given the available resources, basic operational constraints and varying demand while ensuring resource allocations that minimise total cost. To be usable the algorithm must run within reasonable time limits. This dissertation focuses on the bulk freight train scheduling problem of full loads without track congestion but extends to cover operational constraints as well as flexible resource allocation and hubs. A problem outline is given wherein the constraints and decision variables are well defined followed by a review of current literature. An exact formation of the problem is given with benchmarking on small data sets. A genetic algorithm is used to solve for schedules on larger problem data sets. The algorithm was successfully implemented on the 60Mt Coal Line in South Africa which provided notable improvements in efficiencies. Discussion and results are provided.

Acknowledgements

I would like to thank David Lubinsky, my supervisor and employer at *OPSI Systems* for the opportunity to work on a project of this scale, for his insights and guidance as well as the time off to put the required pen to paper for this dissertation.

To Transnet Freight Rail, in particular Hennie Muller and Deon van Niekerk, for their continuous feedback and suggestions that drove the prototypes into production with great success.

To my family for their unrelenting support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Rail networks provide a rich supply of mathematical problems of great variety and scale. This dissertation focuses on the problems associated with bulk freight, as opposed to general freight or passenger rail, with a particular problem instance in the South African context being described, modeled, optimized and implemented.

1.1 Export Coal in South Africa

Transnet Freight Rail (TFR) is South Africa's (SA) primary heavy haul freight rail company responsible for 99% of all freight moved by rail. The coal export line accounts for 30% of all volumes moved by the company at around 70mta (million tons per annum). Coal exports are delivered to the Richards Bay Coal Terminal (RBCT), the second largest coal terminal in the world and largest in the southern hemisphere.

The railway line is broken into three portions, RBCT to Vryheid, Vryheid to Ermelo and North of Ermelo. The majority of loading sites are situated north of Ermelo. Through the network more than 1000km separate the farthest loading site and RBCT.

In order to move coal through this network, locomotives and wagons are used. Locomotives are active rail vehicles which are able to pull wagons. Wagons are used to transport coal from a loading site or loading terminal to RBCT. Several classes of locomotives and wagons are found on the coal line. Wagons come in two sizes; Jumbo (83 ton load) and Small (57 ton load).

Locomotive classes are differentiated by their power source. There are AC, DC, Diesel and AC/DC locomotives used on the coal line. Within some of these classes there are additional types which have different power ratings.

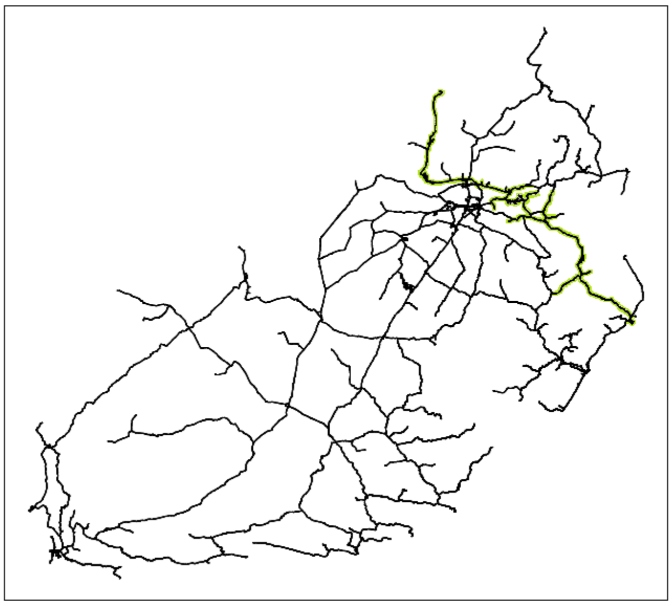

Figure 1.1: South Africa's Rail Network and Coal Export Line

Diesel locomotives are used on portions of track which are not electrified. The *main line* between RBCT and Ermelo is electrified with AC current and the electrified sections north of Ermelo operate on DC current. Ermelo acts as a staging yard where wagons are transferred from DC to AC locomotives or vice versa.

The term *train* is used to describe locomotives which move wagons through the network, between *sidings*, *stations* or *yards*. Trains vary in size across the coal line. Trains on the main line can be between 100 and 200 wagons long while north of Ermelo they may be at most 100 wagons when loaded. There are operational constraints that determine the maximum size of trains that may be operated on a line. This is largely governed by the holding power of the brakes on the train given the track gradients that will be encountered. It may be that a 200 wagon train may be operated if the wagons are empty, but not if they are loaded.

TFR have been in the process of upgrading the braking system used on their wagons, from the standard air-brake system to Electronically Controlled Pneumatic Braking System (ECPBS or abbreviated ECP). The ECP brakes allow for safer operation of trains. Incompatibility between the systems mean that one cannot mix ECP and non-ECP wagons in the same train. This becomes a problem when creating larger trains out of smaller ones, i.e. two 100's combine to give a larger main line train. If their braking systems are incompatible both will have to wait until another 100 wagon train arrives before meeting the departure requirements.

Each of the yards on the coal line have limited capacity. The biggest, Ermelo, can handle at most 2200 wagons simultaneously. Yard processing times are affected by current inventory levels since space to place incoming wagons may be limited and delays begin to occur. Only a few loading sites can take more than 100 wagons at a time while others can take at most 50 small wagons.

The weekly demand at the 50 active loading sites is not static. Several factors ranging from the mining schedules to the previous week's demand determine the demand mix for the coming week. Loading sites have heterogeneous wagon requirements, loading and travel times from their respective hubs. Track and loading site maintenance impose additional constraints on when trains may travel between sites as well as when trains may arrive at certain loading sites. Determining feasible times at which trains should service different loading sites, given the constraints above, is the primary scheduling problem for the export coal line.

1.2 Freight Railway Features and Landscape

In most settings the primary focus in railway optimisation is the problem of moving resources through the network in such a way as to minimise the delay of trains. There are often many resources occupying the line (such as passenger railways in Europe) producing complicated situations which require advanced heuristics to solve [19]. Congested networks are those which operate trains with a high frequency and mix of train speeds often over single line track where train movements need to be carefully planned. The schedule is usually repeatable over some fixed period (typically a day or week) and integration of both freight trains and passenger trains needs to be taken into account.

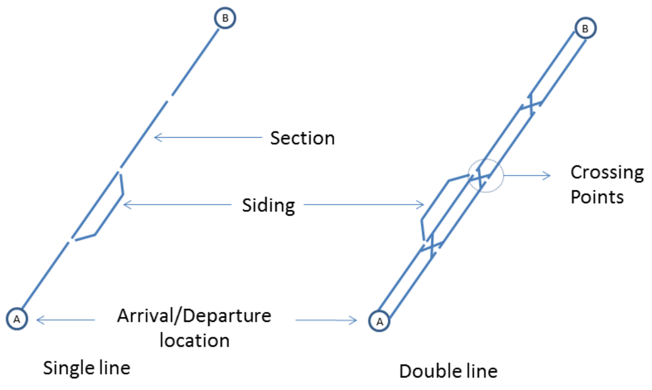

Figure 1.2: Single and double line workings

Figure 1.2 illustrates a schematic view of the differences between single and double line workings. In the single line example, there is only one potential location where trains could pass one another between locations A and B when traveling in opposite directions. However, if all trains are traveling in the same direction, the network would support a maximum of 5 trains *in section* between A and B. The reason for this is the general safety requirement that only one train may be permitted to occupy a section at a time. A section is often defined by the signalling system that monitors a particular section. If a train is occupying a section, a red light will be shown at its starting location, indicating that a following train may not enter. An orange light is shown if there is only one section between the current section and an occupied section.

In the double line working example, a total of 9 trains may simultaneously exist in the network, four in either direction with one stationary in the siding. Crossing points are provided to enable trains to navigate the network should there be some kind of obstruction (such as a broken down train or track maintenance taking place) on the line.

Railways typically use time slots at stations to indicate the times that trains should depart. In some instances, slots represent the physical constraints of the spacing required between trains leaving a siding (for safety/ signaling reasons) or even the electrical grid constraints[1] which play a bigger role in bulk freight than passenger rail. In figure 1.2, station A has a set of departure times in which trains may leave for station B.

In general, a slot will be defined between a pair of sidings or stations. The *Service Design* is the set of all siding pair - slot combinations for a given network, characterising the allowable movement times between all pairs of sidings. In addition to the allowable departure times, the slots may have additional constraints such as the maximum number of consecutive or total slots which may be used in a period of time. The allocation of trains to slots forms the first set of decision variables in the problem.

A task to be performed is triggered by the activation of a slot and is interchangeably referred to as a *job* or *load.* The target number of loads to be completed (also referred to as the demand) is an input to the problem and failure to complete the demand in a given time frame will result in the appropriate penalties being incurred in the objective function.

Historically, the process of selecting slots for departure (which trains will run) has been dealt with independently of the locomotive assignment problem. This is generally referred to as the *Train Timetabling Problem* (TTP) and is concerned with selecting a set of slots for a particular period in which trains may depart from different points. This generates a schedule of planned departure times coupled with accurate arrival times created by calculating the movements of trains through sections and their required passing movements.

In passenger car scheduling, the scheduling is often done in such a way as to take account of the expected demand at different stations and to maximise the profitability of the schedule. In freight networks, the total demand and demand mix can vary significantly from period to period making a fixed weekly schedule impractical.

[1]There is often only enough electricity to support a maximum number trains in a collection of sections. The constraint is met by restricting the rate at which trains may depart a section.

In instances where the demand and departure destination sequence for a set of active slots is known (i.e. the timetable to execute is fully specified) the difficulty in the optimisation is that of assigning locomotives to different departures so as to minimise the cost. This problem is referred to as the *Locomotive Assignment Problem* (LAP) and may present itself as a single or multiple locomotive type variant. In general, the LAP is formulated as a multi-commodity flow problem and is solved using mixed integer programming (MIP) taking into account all the constraints, such as meeting the traction requirements for each load as well as regional constraints.

Yard or hub[2] operations are generally considered separately from the TTP and LAP. The operations around processing incoming wagons[3] into categories are referred to as the *grouping* or *blocking policy.* This is often used in freight networks for processing incoming wagons into different yards, each of which has a common destination. Trains are then dispatched on fixed slots once operational requirements such as a minimum or maximum number of wagons in a yard have been met.

Routing and makeup models [3] are used to determine the train frequency required to meet customer demand given the blocking policies in a network of yards. Integer programming models have been successfully used to model this process in isolation. More recently, models have begun to accommodate solving the TTP and Routing/Makeup Models simultaneously [27].

There are three common domains in which freight related problems are classified:

1. Routing/Makeup Problem (RMP): Determining the required frequency of trains between capacitated siding-destination pairs to meet customer demands (wagon flows)

2. Train Timetabling Problem (TTP): The detailed movements of trains through sections of track given the required frequencies between sidings and tentative departure times from 1 above.

3. Locomotive Assignment Problem (LAP): Allocating locomotive resources in the most efficient way to minimise the cost of executing the schedule provided in 2 above.

Bulk freight refers to environments where resources are dedicated to the task of supporting a particular line or network. Resources are allocated to fixed sets and all wagon sets are loaded to their maximum capacity at a

[2]A yard is the common term for freight station

[3]In American literature, wagons are often referred to as ‘cars’

single siding. The bulk freight problem is not well studied in the literature and requires solving several aspects of existing problems simultaneously to achieve reasonable results. This is discussed further in Section 3.

This list is not exhaustive and there are several related problems pertaining to empty wagon distribution, multiple carrier models and capacity estimation. These related issues will not be covered in detail but may be mentioned where applicable.

1.3 Bulk Freight

Railways that are designed for bulk freight distribution may have centralized hubs or crossdocks where wagons are recombined (or broken up), in line with grouping/blocking policies, into larger (or smaller) sets before completing their next journey. Some reasons for this behavior could be due to different electrical configurations on different sections of track and geographic or loading site constraints which limit the length of trains. The makeup rules involved are generally much simpler as large sets of resources are normally maintained as units.

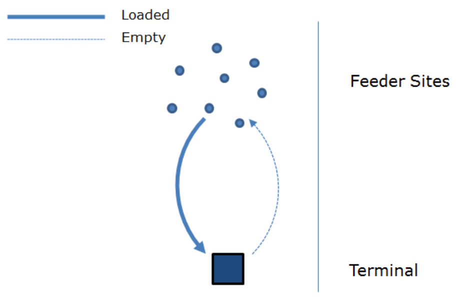

Figure 1.3: Problem Type 1

By its very nature, bulk freight generally deals with quantities on a much larger, fixed scale. For example, a customer may order wagons in units of 100 equating to thousands of tons for a single train.

Sidings where goods are loaded are often referred to as ‘feeder sites’. In bulk export the final destination of goods is generally a port or terminal. Here goods are transferred from the wagons either directly onto shipping vessels or, more generally, onto stockpiles organised by commodity grade which are later loaded onto vessels using reclaiming equipment. Figure 1.3 illustrates the layout of the simplest instance type described in the literature [38] and is described as the *Coal Line Scheduling Problem.*

The more complicated scenario that will be discussed is shown in Figure 1.4 where sets of feeder sites are serviced from different hubs with the service being planned globally for the whole system. Trains which can be sent to feeder sites are a function of the resources available at the source hub which are in turn a function of prior trains which have been planned. A fully integrated resource schedule is required to avoid infeasible plans. Different train configuration rules are often required between different hubs and their associated feeder sites.

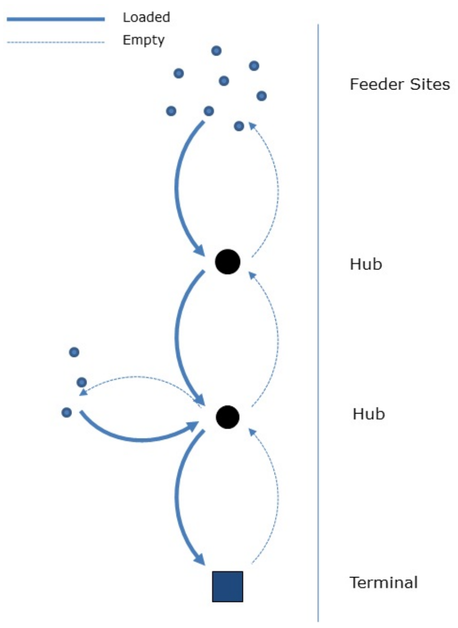

Figure 1.4: Problem Type 2

The general constraints of the bulk freight scheduling problem are listed below:

- Site - Locomotive Class exclusions
  Feeder sites may exclude certain locomotive types due to operational constraints. An example of this is when sites are not electrified, in which case diesel locomotives would need to be dispatched.

- Site - Wagon exclusions
  Wagon types may also be excluded from certain sites, this may be due to the site being unable to accommodate larger wagons due to axle load constraints on the track or have incompatible shunting locomotives at the site itself. Certain locomotives may be unable to couple with wagons due to different braking systems. Similarly, wagons may be incompatible with one another.

- Site operating and maintenance hours
  Sites may have specific operating hours in which loading may occur.

- Site reclaiming times
  After loading has taken place at a site it will be unable to commence loading again for a fixed amount of time. This is referred to as the reclaiming time.

- Site specific loading/reclaiming times
  Each site has a specific loading and reclaiming times. The loading equipment available at each site generally drives the efficiency of the loading and reclaiming time. More efficient equipment generally requires greater capital investment and as a result is generally correlated with larger customer demand.

- Multiple loading points within a site
  This allows for more than one train to be loaded at a time.

- Loading sites may share a siding
  This prohibits more than one train being loaded at a time between multiple sites.

- Rapid vs mechanically loading sites
  Mechanically loaded site are generally allowed to perform drop loads where wagons are left at a siding to be loaded (due to the long loading time) and locomotives return at a later stage to collect the wagons. Locomotives and wagons do not decouple at rapid loading sites since loading is normally completed quickly.

- Resource states
  Locomotive empty (coupled with empty wagons), loaded (coupled with loaded wagons) and light (no wagons coupled) travel speeds between all pairs of sites[4].

- Regional locomotive constraints
  Individual locomotives may be bound to operate within certain areas regardless of their class for convenience. For example, a diesel locomotive has no operational constraints prohibiting it from traveling to any area. The locomotive may be required to remain within a certain area (and be potentially under-utilized as a result) so that it can be relied upon should another train break down in the area and require assistance.

- Locomotive - Wagon exclusions
  Locomotives may have incompatible braking systems with certain wagon sets.

- Wagon - Wagon exclusions
  Wagon sets may have incompatible braking systems with one another.

- Minimum headways between dispatches or arrivals
  Safety regulations require a minimum distance be kept between trains.

[4]The travel time matrix is not typically symmetric in this setting

The time between trains is referred to as the headway and is the minimum time that needs to be kept between trains running in the same direction either departing or arriving at a station.

- Maximum slot usage constraints
  In order to treat the problem space of a bulk freight network independently of all other network activities, there may be restrictions on the number of slots that may be used in order to accommodate other traffic passing through a common area. For example, a maximum of 3 consecutive slots may be used before at least one gap must be left to accommodate other traffic that may be leaving from a particular yard or station. The spare slots may also be required as "catch up" capacity.

- Planned track occupations
  The term *occupation* is commonly used in South Africa to describe maintenance that is being performed on a section of track between sidings. A *Total Line Occupation* refers to the full closure of a section of track, prohibiting any movement of resources until a certain time, as opposed to a fixed delay. As such, a single occupation may impact the travel between several sidings. In practice, a train would not depart a station knowing it was running into a total line occupation unless there was significant confidence that the occupation will be lifted by the time the train arrives at that section of track.

- Hub capacity constraints
  Congestion within hubs/yards is common in practice. Due to the large volume of resources that are being moved around, hubs may not have the capacity to absorb all trains entering the hub if for some reason they are unable to depart trains. A total line occupation on one side of a hub would cause this to happen. The physical constraints of hub capacity need to be respected to ensure that the schedule being produced is executable. There are certain feeder sites which may exhibit the same property whereby there may be multiple loading stations in which case the number of wagons in the siding may be constrained to a certain total. The capacity constraint has been defined to handle the maximum total number of wagons as well as the maximum total number of empty and loaded wagons.

- Different possible train configurations between feeder-hub, hub-hub and hub-terminal legs
  Due to the track configuration and geographic constraints, different train types are operated on different sections of track. A typical example of this would be between the feeder sites, hub and terminal. Smaller trains are used on the feeder leg and larger trains on the ter-

minal leg. This results in the hub combining the wagons arriving from feeder sites into larger groupings which are then sent to the terminal. Similarly, the hub will also split wagons coming up from the terminal into smaller, empty sets, which are then sent to the feeder sites. In instances such as this, it might be that certain locomotives can operate on both the terminal and feeder site legs, but this is not generally the case.

- Locomotives and wagons retained as sets of units
  In many problems, such as the LAP, due to the variation in the size of trains that are departed, much time is spent selecting locomotives that best suit a particular load. In bulk freight scheduling, the size of the loads is largely fixed over particular corridors. This enables the operations to run more smoothly as the there is less un-coupling and re-coupling of locomotives to create the best possible locomotive set (or *consist*) for a load. Similarly, wagons are generally kept together in sets whose smallest size generally matches the smallest number of wagons that may be requested by a customer or moved through a particular corridor. The problem of assigning resources to trains can then be handled at a set level rather than an individual resource level. This has computational implications which are very favorable.

Bulk freight lines such as that illustrated in Figure 1.4 and described in Section 1.1 exhibit a unique characteristic in that the schedule is not as much a function of the capacity of the line but rather of the resources themselves. The majority of the railway lines support traffic in both directions and in some places even have an additional passing loop. Due to the uniformity of the sizes of the trains (in terms of mass, not actual resource mix) most trains move at the same speed, so orchestrating the passing of a slower train only occurs when deviations occur due to unplanned resource failures.

There are many instances in which trains may deviate from a perfect plan. Trains may not depart when planned because of drivers not being available, equipment not being present or even resource failure to name a few. The moment a train is unable to depart as planned (to a certain degree) the current schedule becomes invalid.

A heterogeneous fleet of locomotives and wagons and the variable demand means that having a periodic schedule is not possible in a setting where the tightest constraint are the resources themselves. The volume of commodity being moved is typically the chosen Key Performance Indicator (KPI) resulting in trains being dispatched on a *run when ready* basis. This means that if a train is ready prior to its planned departure time that it will depart rather than wait in the interests of freeing those resources sooner for another trip. A schedule that supports this behavior is required that can utilize different slots in different periods and take advantage of the availability of different resource classes at different times given their prior trip allocations.

Finally, the different travel, loading and reclaiming times and resource constraints at feeder sites have a large impact on the slots which can be feasibly selected to run on any day in the planning period. Greedily selecting sites which process trains quicker will result in resources bottlenecking in the future. Demand also needs to be consumed in a manner that matches the tightness of different constraints and future expected constraints, based on the current decisions, since resources will be recycled several times within a single planning period.

1.4 Solution Approach

A model was built to describe the constraints, relationships and objectives of the bulk freight train scheduling problem. Initially an exact approach using branch and bound to solving the model was tried but did not scale to larger problem sizes.

A heuristic method was developed in an attempt to solve larger problem instances. While the performance of the heuristic seemed acceptable for medium sized problems, the Genetic Algorithm implemented as the final solution was able to outperform it in a few minutes on much larger problem instances and provide schedules which expert train planners could not find fault with.

Due to the ease with which infeasible schedules could be produced, a direct meta-heuristic approach was not used. Instead, a *graph builder* or *scheduler* was used to interpret the permutations produced by individuals in the population and create schedules that best matched the order of jobs while still maintaining a feasible schedule. This indirect approach is not uncommon in the literature and saves having to develop problem specific constraints at the chromosome level as well as concentrating the search on feasible regions of the search space.

1.5 Dissertation Outline

The literature review, Chapter 2, covers the areas where the majority of academic research has been conducted with respect to two scheduling problems; the train timetabling problem and the locomotive assignment problem (Sections 2.1 and 2.2 respectively).

Section 3 provides a formulation of the mathematical model for the bulk freight scheduling problem. Section 4.1 outlines the computation complexity of solving the formulated model and attempts to solve small problem instances. Benchmarks provided by the exact search are used in Section 4.2 to test a heuristic branch-and-bound procedure.

The details of the Genetic Algorithm used to solve bulk freight scheduling problem for realistic problem sizes are given in Section 4.3. Comparison to smaller data sets and performance on larger data sets is provided.

Implementation details of the solution at TFR, resulting volume increases and discussion are provided in Section 4.4.

Chapter 2

Literature Review

There is a plethora of different sub-problems in railway optimisation. Problems are commonly considered to fall into one of three horizons [3] of optimization, namely:

- Strategic (>1 year)
- Tactical (1 week - 6 months)
- Operational (< 1 week)

The strategic domain is often concerned with large capital expenditure, such as purchasing locomotives or constructing new sections of track, which requires a long lead time on the decisions being made. Tactical decisions relate to the time table for the next period and the operational tier is concerned with the immediate window of execution. Within each of these domains there exist several sub-problems which are often specific to particular case studies.

Courdeau [19] provide an excellent survey of the freight railway scheduling landscape up to and including 1998. Major works were classified by their planning horizon, problem type, model structure and solution approach. Models are grouped into major categories either falling under the freight routing or scheduling domain as given in Table 2.1.

Freight Routing	Scheduling
Analytic Yard Models	Analytic Line Models
Blocking Models	Train Dispatching Models
Routing and Makeup Models	Locomotive Assignment Models
Freight Car Management Models	

Table 2.1: Courdeau [19] model grouping

Freight routing models focus on the movement of wagons through the network rather than the trains themselves [19]. The objective is generally to minimise the cost associated with moving freight through the network given that it will need to be classified into blocks of similar freight at different yards that have a common destination. There are limits on the amount of work that can be done in different yards.

Bodin *et al* [6] (1980) were the first to formulate the blocking problem as a mixed integer programming problem (multi-commodity flow problem) with the additional constraints of yard capacity and block size constraints. It is noted that solving for the optimal blocking strategy within a yard can lead to a globally sub-optimal solution due to yard congestion. As a result, the formulation across all yards in the network is solved yielding the distribution of classification work to be done at different yards throughout the network. Since the work of Bodin *et al* [6], many variants have been solved using dynamic programming, branch-and-bound, Lagrangian relaxation and other heuristics. In the context of bulk freight scheduling these models are not applicable in their traditional form as there is no wagon reclassification or blocking required in bulk freight, since all resources already operate in fixed sets.

Train dispatching models are concerned with minimizing train delays or deviations that occur due to the required meet/pass movements of trains in the network according to a planned schedule. It is common to represent the problem as a MIP and to solve this using a heuristic decomposition. The models have been extended to cover variable velocity trains and generally incorporate the required headway between trains. Carey and Lockwood (1995) [17] modeled a relatively small single line network consisting of 10 trains and 10 links and reported good results using a branch-and-bound procedure. Recent works have modeled this problem more generically using the Job-Shop Scheduling framework [37].

The assumption of independence between problem landscapes leads to suboptimal solutions for problems which in reality lie across multiple domains. Many of the works prior to the work of Gormon (1998) [27] dealt with the optimization problems related to freight rail in isolation from one another. Not only was Gorman the first to tackle reasonably sized compound problems but also the first to do so using evolutionary algorithms with hybrid schemes.

Gorman's [27] approach to solving the weekly routing and scheduling problem was to discretise the time horizon into hours and use a simple genetic algorithm (SGA), with binary encoding, to determine the selection of trains to operate. Once an individual in the population has been generated, its objective function value is determined by solving the traffic-assignment problem for the resulting selection of services. Gorman notes a result from Davis [21] is that problem specific operators or directed operators can be used in genetic algorithms without loss of generality. As such, Gorman uses a *tabu search* (TS) to increase the search performance.

Gorman [27] also implements a *bit slide* mutation operator in the genetic algorithm (GA) which randomly moves an existing bit to the left or right of its current position, analogous to moving an existing train to either an hour earlier or later. The TS is implemented by copying an existing solution in the population, modifying it using the TS, and then re-evaluating the objective function. Both solutions are added back to the population with the normal selection operators determining which individuals will be preserved in the following iteration of the GA. The TS takes advantage of the information in the demand-flow aspect of the problem such as adding in additional trains when demand has gone un-serviced.

It was reported that as the problem size grew, the performance of the genetic algorithm degraded when used on its own. The convergent performance of the GA was matched by the GA-TS within 6% of the original number of iterations and instances which had never seen a feasible solution under the pure GA were experiencing 100% success rates under the GA-TS. Gorman notes that the solution quality under the GA-TS also deteriorates slightly as the problem sizes increase. A 3.8% decrease in operating cost was reported when compared to existing operating plans to the results of the GA-TS solutions.

The remainder of this literature review deals with the TTP and LAP as separate problems which is a reflection of the way it has been handled in the literature. There is one exception when the LAP does not need to be solved in practice, namely the *coal train scheduling problem* as coined by Liu and Kozan [38] where resources maintain a fixed 1-to-1 mapping with trains in the network. This paper has some characteristics that closely resemble the bulk freight scheduling problem and is discussed extensively at the end of the TTP section.

2.1 Train Timetabling Problem

Line activities refer to general traffic between different locations, normally some distance apart. This includes overtaking/priority rules for passing and dispatching conditions. *The dispatching decision is similar to a machine scheduling problem where trains correspond to jobs and machines to track sections*[1] [2]. It is possible to formulate this problem as an integer program but this is limited to only being able to solve problems with a small number of sections with the objective on minimizing the total travel time [43]. Many works have been published in recent years using different heuristics to solve the Job Shop Scheduling Problem (JSSP) which include genetic algorithms [40], guided local search [4], memetic algorithms (MA) [18], hybrid genetic algorithms [31] and hybrid ant-colony optimisation [33].

More complicated, problem-specific heuristics have been built up over many years to solve the timetabling problem starting with the benchmarks set by Cai and Goh (1994) [13] in terms of performance. The authors found that they experienced $O(NK)$ [2] average running time where N is the number of trains and K is the number of passing loops. The algorithm presented did not extend to cover double line workings. Cai and Goh (1998) [14] presented an extension of this algorithm that was implemented as a real time train dispatching system in an Asian railway.

Higgins *et al* [30] (1997) model the single line scheduling problem using a non-linear MIP and solve it to optimality in several instances using a branch and bound approach. Their model is targeted mostly for use in the live environment as a decision support tool to train monitors. They however demonstrate the capability of the procedure for evaluating the strategic implications of additional trains given a network of stations.

The model is later extended to iteratively determine the number and location of sidings on a single track corridor given a fixed demand [32]. In the same year, Higgins *et al* [31] examined the use of different heuristics such as GAs, TS and two hybrid algorithms to solving the train scheduling problem by comparing the results of the heuristics to the optimal obtained from the methods presented in [30]. They reported that the genetic and hybrid algorithms were within 5% of the optimal for at least 90% of the test cases. The GA found the optimal solution 50% of the time with the TS being less successful only finding the optimal 10% of the time but performed better on larger problem sizes when time constraints were enforced. The hybrid algorithms performed the best on average but at a computation cost of 7

[1]The *machine scheduling problem* is referred to as the Job Shop Scheduling Problem (JSSP) in current literature.

[2]The worst case performance is given as $max\{O(N^4K), O(N^3K^2)\}$.

times more than the GA.

Caprara *et al* (2002) [16] present a particularly well cited paper on the train timetabling problem in which they present a proof that any TTP can be transformed into a *Max-Independent Set Problem* (MISP) demonstrating that it is NP-hard. Two graph theoretic models are presented in this paper, the first of which is not solved for two reasons; firstly, it would not scale to large size problems without heuristic decompositions, and secondly, the heuristic decomposition resulted in too many lagrangian profit variables making it impractical for real world instances.

The second graph model presented models the track constraints through the nodes of the graph instead of the arcs, resulting in far fewer lagrangian profits in the decomposition. The model presented allows for the removal of trains from the schedule. An iterative two phase heuristic is applied. Firstly, trains are ranked in descending order of their lagrangian profit and are scheduled one by one. Since all other existing trains in the solution are fixed, the optimal insertion for each preceding train (T_p) that is scheduled can be found by finding the set of minimum cost arcs to accommodate T_p.

The second phase is a refinement heuristic which, in an arbitrary order, compares the actual schedule times to the ideal schedule times for each train in the schedule. If any of the scheduled times do not correspond, a new path is calculated which is maximal with respect to the actual profit (not the profit which was maximal with respect to the lagrangian weights). The old path is then updated with the new one. The heuristic then attempts to add back any unscheduled trains which may have been left off due to previously negative lagrangian multipliers which may now be positive due to the updates in the previous phase of the heuristic. A complicated subgradient optimization procedure updates the lagrangian multipliers in the next relaxation to be solved.

The model allowed for different penalties on lateness for different trains. As a result, high speed passenger trains received higher penalties than freight trains in their computational experiments. They reported gains of between 0.3% and 20.8% over a greedy heuristic. Run times were very acceptable on the hardware of the time (500Mhz Workstation) taking at most 11 minutes on real world data. Not surprisingly, the largest gains over the heuristic were on artificial data sets which were designed specifically to test large congested instances. The largest data sets took around 92 hours to solve.

Caprara *et al* (2006) later present an updated model which uses a similar formulation but with stronger constraints on overtaking which dominate existing constraints in the model by assuming that trains move at a constant speed between stations [15]. Argument is given that the finer grained details of train movements in the operational landscape do not need to be incorporated in the planning model and that the schedule produced is still feasible. The benefit of the additional constraint set is that the dominated constraints can be excluded and the subgradient optimization procedure has a much improved running time as a result without affecting the solution quality significantly.

Kwan and Mistry [34] (2003) present a co-evolutionary strategy for solving the timetabling problem for the UK passenger network. Train Operating Companies (TOC's) bid for track time and their objective is to produce a schedule that best accommodates all of the role players through their soft constraints whilst meeting all hard constraints. An example of a soft constraint is that of arriving at a particular station within a certain window. Train schedules are revised on a 6 monthly basis and the bidding for new trains to operate on the network commences around 18 months prior to the schedule being published.

The strategy employed in solving the problem stems from there being three sets of decision variables in the problem, namely: departure times, scheduled run times and resource options at a station. Each of these decision variables is deemed a *species* and is evolved while holding other decision variables fixed. Collaborating species, against which the individual species is being evolved, are chosen using tournament selection with a tournament size of two. Other species are then selected for evolution. The type of algorithm used to evolve individual species is a function of the underlying data structure used to encapsulate the decision variables. A simple genetic algorithm [26] is used for species that implement a binary encoding. Departure time decision variables are represented as an integer array with each increment representing a 30 second period. An adaptive step-size mutation Evolutionary Strategy (ES) [29] with a gaussian mutation operator is used to solve for the departure time decision variables.

At the time of publication, UK division *Railtrack* were using custom train software which implemented a Simulated Annealing (SA) meta-heuristic to solve the timetabling problem. Results on the co-evolutionary strategy showed that it outperformed the SA significantly and bested it on all evaluation criteria except for one which was a soft constraint. Run times were fractionally slower than the SA but negligible when compared to the cost savings achieved. The strategy used was not tested against an exact formulation on a small test instance.

Semet and Schoenauer (2005) [42] present a memetic algorithm to solve the problem of handling small perturbations in real world scheduling problems. The memetic algorithm is a hybrid algorithm consisting of an evolutionary strategy with a mathematical programming tool, namely, CPLEX[3]. A mapping function, or *scheduler*, is used to translate the permutation of requested trains within the evolutionary algorithm into a valid schedule, by inserting the trains in their requested priority into an initially empty schedule, one by one. This means that each train that is inserted is locally optimal but the train itself could have a sub-optimal route overall. The authors note that implementing a more complicated method to insert trains optimally would have come at far larger computational cost but also that optimal schedules may exist in a portion of the solution space which is inaccessible by the permutation coupled with the heuristic insertion. They cite development and computational cost as well as a larger search space as motivation for not extending the heuristic insertion algorithm.

The Evolutionary Algorithm (EA) proceeds as normal but after the population reaches a convergence criterion the best solution is passed to CPLEX as an initial starting solution. This high quality starting solution allows CPLEX to start from an area of the search space which allows its branch-and-bound algorithms to prune much larger portions at a time. There is no iterative feedback between CPLEX incumbent solutions and the EA. Using the EA as a seed to CPLEX allowed for achieving near optimal results in 4 as opposed to 24 hours. The authors reported that the average performance of the hybrid algorithm was superior to CPLEX alone in almost all instances by approximately 14%.

Tormos *et al* (2008) [44] constructed a GA to solve the TTP for the Spanish Manager of Railway Infrastructure (ADIF) which was packaged and is being used successfully by ADIF. Unlike the indirect GA representation used by Semet and Schoenauer [42] (which uses a *scheduler* to interpret the genes and construct feasible schedules) Tormos *et al* [44] use a direct chromosome representation widely used in project scheduling. A list of activities with precedence constraints is used as the gene structure and parsed into feasible schedules by resolving conflicts according to the priorities specified in the genes. The authors note that many permutations are equivalent in that they will produce the same schedule since differences only arise when conflicts are resolved.

A feasible initial population is formed by a novel heuristic scheme. Each next activity to be added to the schedule across all trains is biased according to the *regret* of not including that activity. The bias is calculated by measuring the current deviation against the optimal running time of the train.

[3]CPLEX is IBM's 'High-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming'

Thus, trains which are deviating as the schedule is being built get their movements probabilistically prioritized over other trains. This method is referred to in the literature as *Regret-Based Biased Random Sampling* (RBRS).

Uniform crossover operators are used but modified to keep the relative precedence constraints when merging activities from two parents. The mutation operator randomly selects a new position for an activity within the allowable portion of the gene that will not violate the precedence constraints. Standard tournament selection with a tournament size of 2 is used as the selection operator.

Tormos *et al* [44] present their results by comparing the final solutions obtained by the GA against two seeding mechanisms, the RBRS scheme presented, and a pure random train selector. Both run for an equivalent amount of time. The RBRS outperformed the random selection scheme by around 5.7% across all test cases. Since the GA uses the RBRS scheme as part of its initial seeding we would be very surprised if it performed worse than the RBRS on average. However, the authors report that the improvement of the GA over the RBRS scheme is proof of its efficiency in solving railway problems, a claim which is not fully substantiated or supported. The authors did not compare the performance of the GA, given a RBRS generated population, against a hill-climbing or random search algorithm given the same starting solution. The GA reduced deviation in the schedule by an average of 8% over the best of the RBRS results across the test instances reported.

Kozan and collaborators have been working on the train dispatching models over many years. Staring with optimal branch-and-bound techniques already mentioned (1996) [30] for single line workings, heuristic benchmarking (1997) [31], capacity determination (2006) [9], modeling the problem as a JSSP (2008) [10], extending the model to a *Blocking Parallel-machine Shop Scheduling Problem* (BPMJSS) (2009) [37], inserting trains in an existing timetable (2009) [11] and addressing the adjusting of timetables to handle perturbations and unnecessary overtaking (2009) [12].

More recently, Liu and Kozan [38] (2011) have addressed the problem of "optimizing a coal rail network under capacity constraints". This paper is of particular interest to us as it most closely resembles a subset of the problems we face in bulk freight scheduling. The model presented is an extension of that presented in [37] where jobs are represented by trains, single line tracks are modeled as machines and double line sections as parallel machines. A constructive heuristic inspired by the *shifting bottleneck procedure* (SBP) is described in [36] which constructs a feasible initial solution to the BPMJSS problem. The feasible solution is then improved through the use of the Tabu

Symbol	Definition
n	number of jobs
m	number of machines
J_i	job i $(i = 1, 2, ..., n)$
M_k	machine k $(k = 1, 2, ..., m)$
h_k	the number of units on machine k
u_{kl}	the l^{th} unit of machine k $(l = 1, ..., h_k)$
o	index of sequence position of operation in one job $(o = 1, 2, ..., m)$
s_{ilk}	the starting time of job i on the l^{th} unit of machine k
p_{ilk}	the processing time of job i on the l^{th} unit of machine k
C_{max}	the maximum completion time or makespan
r_{iolk}	$= 1$, if the o^{th} operation of job i requires the l^{th} unit of machine k $= 0$, otherwise
x_{ilk}	$= 1$, if job i is assigned to the l^{th} unit of machine k $= 0$, otherwise
y_{ijlk}	$= 1$, if both jobs i and j are assigned to the l^{th} unit of machine k and job i precedes job j (not necessarily immediately) $= 0$, otherwise
w_{ijolk}	$= 1$, if the o^{th} operation of job i requires the l^{th} unit of machine k and job j is scheduled on this same unit as its successor (not necessarily immediately) $= 0$, otherwise
L	a very large positive number

Table 2.2: BPMJSS Notation

Search (TS) meta-heuristic.

The notation (Table 2.2) and mathematical formulation for the BPMJSS is given as follows:

$$\textit{Minimise } C_{max}. \tag{2.1}$$

Equation (2.1) is the objective function which is to minimise the makespan.

$$\sum_{l=1}^{h_k}\sum_{k=1}^{m} r_{iolk}(s_{ilk}+p_{ilk}) \leq \sum_{l=1}^{h_k}\sum_{k=1}^{m} r_{i,o+1,l,k}s_{ilk} \;\; o=1,2,...,m-1, \forall i. \tag{2.2}$$

Equation (2.2) ensures the starting time of the $(o+1)^{th}$ operation is no earlier than the finish time of the o^{th} operation of job i.

$$s_{ilk} \geq s_{jlk}+p_{jlk}+L(y_{ijlk}-1) \;\; \forall i,j,k,l. \tag{2.3}$$

Equation (2.3) restricts that both jobs i and j are processed on the l^{th} unit of machine k and job i precedes job j (not necessarily immediately).

$$s_{jlk} \geq s_{ilk}+p_{ilk}+L(y_{jilk}-1) \;\; \forall i,j,k,l. \tag{2.4}$$

Equation (2.4) restricts that both jobs i and j are processed on the l^{th} unit of machine k and job j precedes job i (not necessarily immediately).

$$y_{ijlk}+y_{jilk} \leq 1 \;\; \forall i,j,l,k. \tag{2.5}$$

Equation (2.5) restricts that the conditions in (2.3) and (2.4) are exclusive.

$$\sum_{l=1}^{h_k}\sum_{k=1}^{m} x_{ilk}=1 \text{ and } x_{ilk}+x_{jlk}-1 \leq y_{ijlk}+y_{jilk} \;\; \forall i,j,k,l. \tag{2.6}$$

Equation (2.6) restricts that each unit can process at most one job at a time.

$$\sum_{l=1}^{h_k}\sum_{k=1}^{m} r_{imlk}(s_{ilk}+p_{ilk}) \leq C_{max} \;\; \forall i. \tag{2.7}$$

Equation (2.7) restricts the completion time of the m^{th} (i.e. last) operation of each job to be no earlier than the makespan.

$$s_{ilk}, p_{ilk} \geq 0 \;\; \forall i,l,k. \tag{2.8}$$

Equation (2.8) satisfies non-negativity condition.

$$\sum_{j=1}^{n}\sum_{l=1}^{h_k}\sum_{k=1}^{m} r_{jolk}s_{jlk}w_{ijolk} \geq \sum_{l=1}^{h_k}\sum_{k=1}^{m} r_{i,o+1,l,k}s_{ilk}, \;\; i \neq j; \;\; o=1,2,...,m-1; \;\; \forall i. \tag{2.9}$$

Equation (2.9) defines the blocking conditions and satisfies the starting time of successors on the same machine should be greater than or equal to the starting time of successors of the same job, for each operation.

The authors note that that a key aspect of this model is that is has the ability to block a machine if it cannot start the next job on another machine immediately after its current job. This is required to correctly model the true nature of line capacity when scheduling trains through a network.

Liu and Kozan [38] describe the Tabu Search operations used in their computational experiments as an exchange (an *E-Move*) and remove/re-insert operator (an *I-Move*). The E-Move swaps two jobs in the permutation sequence while the I-Move moves a job from position a to b. The sets of moves between these two operators are not mutually exclusive and redundancy checks are performed to avoid spurious operations. The unrestricted neighborhood of the TS is defined as a function of the possible operations with respect to a pre-specified job sequence, the computational complexity of which is $O(n^2)$ since there are $(n-1)^2$ possible E-Moves and I-Moves, not taking into account duplicates. The authors note that this neighborhood is too large and they reduce it to only consider E-moves between adjacent jobs in the current sequence. This reduces the neighborhood size to $O(n)$ with reasonable constants.

The algorithm developed was used to determine the feasibility of a dedicated export line in Australia. The line modeled had 41 sections and demand from 3 mines (with 9, 24 and 10 loads each) totalling 43 trips that needed to be made. The loading time at the mine was assumed to be 4.3 hours and 5 hours offloading at the terminal. Seven trains were made available to complete the work. A train is defined in this context as a fixed allocation of locomotives (4) and wagons (280) which remain as a unit for the duration of the whole schedule[4]. Headway between trains is ignored, presumably due to the limited number of sections.

There is a problem in the way in which results are presented in this paper. The authors present a "random" permutation for the order in which the mines will be serviced by the terminal which is not random at all. The authors concede that *there is no doubt that the initial random train schedule is not ideal* but do nothing to rectify the situation. Results stating the 'increase in capacity' (stated at 38.81%) of the algorithm are compared against the 'not ideal' baseline solution. This is an attempt to overstate the performance of the algorithm and perhaps the difficulty of the problem faced.

Given the estimate of the lower bound on the makespan assuming no track, terminal or mine conflicts occur is 135.12 hours, we can calculate, on average, the time spent loading or unloading as $\frac{43}{7}(4.3 + 5) = 57.12$ which

[4]This assumption is relaxed in the bulk freight scheduling model presented in Section 3.

suggests that around 42% of a train's time is spent either in the terminal or at a mine. The baseline solution used by the authors forces immediate queuing at the mines in an absolute worst case scenario since all trains depart immediately after one another (within their section constraints) having been allocated to do all the work for the first, second and then third mine. As a result, trains arrive at an interval governed by the longest section traversal time between the trains en route from the terminal to mine 1. This means that the second train which arrives waits for the first to load[5], the third waits for the second and first to load and so forth. It can be seen from the train diagrams presented their paper that the queue length grows to a size of 7 at the first mine with the first train actually offloading and returning for its second load from mine 1 to find a queue of 4 trains still waiting to be loaded. There are two offloading points in the terminal which allow for the faster processing of trains than at any individual mine.

There is no simple way to derive an accurate estimate of how over-inflated the baseline solution is given the information provided by the authors. A more thorough analysis would have measured the performance of the algorithm against a sufficiently large sample of random permutations.

It is also unfortunate that the authors did not consider a *hot start*[6] of resource positions. It is clear that conflicts at the mine are the tightest constraint in this problem space given that there are 7 trains moving through the network and only 3 mines to send them to. The moment 4 trains are en route to mines we would expect at least 1 conflict (since the travel time to a mine is less than the loading time at the mine). The authors did not comment on the systematic waves of resources queuing to load and then queuing to offload throughout the period as a result of the poor resource starting positions. Instead they chose to state that a 'near-optimal' result had been achieved with massive efficiency increases over what was a worst case scenario baseline example. There is no question that the timetable provided may be 'near-optimal', however, it is far from the actual capacity of the line being studied due to careless data configuration.

The problem presented in this last paper most resembles the bulk freight scheduling that is examined in greater detail in this dissertation. Real-world bulk lines need to take consideration of additional penalties caused when excessive queuing takes place. Due to the size of the trains, stopping several resources on the line back-to-back due to poor scheduling does allow for the resources to complete the remainder of their tasks in the time originally

[5]Since the loading time at the mine is far greater than any single section traversal time

[6]It is not realistic to expect all resources to start in exactly the same location at the same time. A typical solution is to iteratively take the positions at the end of one schedule and use them as the starting positions of the next until a steady state is reached.

planned.

As an example, if a train is forced to be delayed by a few hours outside a mine, this will cause the crew to run out of working hours on the return leg to a hub or terminal. Strict labour laws govern that drivers may not work beyond a set number of hours and as a result this will require a new set of crew to be dropped off at the mine, potentially causing a further delay in the trains departure back to the hub or terminal.

In a less ideal (but not uncommon) situation, the same crew return with the train from the mine, resulting in a crew exchange being performed on the main line potentially blocking other trains moving through the network. The model presented in Section 3 attempts to address some of the more practical constraints which are required when modeling a bulk freight network.

2.2 Locomotive Assignment Problem

Another area of railway optimisation which has received great attention is that of locomotive assignment. This problem is concerned with providing enough locomotive power for a particular train and allocating locomotives in such a way as to minimise the *deadheading* or *light* locomotive movements, thus reducing costs and increasing utilization. Locomotives are the single most expensive rolling stock item and as such, poor utilization is heavily frowned upon.

A *consist* is the term used to describe a collection of locomotives coupled together. *Light* locomotives are locomotives which are traveling through the network without any freight cars or wagons attached. A *Deadheading* locomotive is a locomotive which is part of a consist but it not actively involved in the hauling of freight. This may occur when locomotives are being repositioned in the network and are 'hitching a ride' with another train to get to their destination at small cost to the railway. Additional time and operational costs may be required to add deadheading locomotives to an existing consist (before departure and then on arrival).

Light locomotive movements may be required to balance the distribution of locomotives in the network in order to meet future planned trains if the service is imbalanced, which may be a result of tonnage moving more in one direction than another [23]. Light locomotives are also used when picking up loads from sidings that may have long loading or processing times; likewise, light locomotives would have returned to a hub after dropping the load off. Intelligent assignment of locomotives would increase the utilization of light locomotives by ensuring that if a locomotive dropped off a load in one siding, that there would be another load nearby that it could collect, thus reducing the time spent running light.

Florian *et al* (1976) [23] formulated the Locomotive Assignment Problem (LAP) as a mixed integer program and achieved satisfactory results for medium sized problems and disappointing results for larger problems using Benders decomposition [5]. A critical assumption made in the model proposed by Florian is that a fixed schedule of departure/arrival slots for the period already exists, i.e. the train timetable has already been solved independently of the LAP as well not having to select locomotives from an existing fleet.

Given a set of jobs (movements between various stations and depots) to be completed on a day[7] Booler [7] describes the locomotive scheduling prob-

[7]To be repeated cyclically

Symbol	Definition
n	number of jobs
m	number of locomotive classes
A_k	be the set of jobs which can be worked by locomotive of class k
y_{ki}	$= 1$, if job i is allocated to a locomotive of class k $= 0$, otherwise
x_{kij}	$= 1$, if a locomotive of class k works job j immediately after completing job i $= 0$, otherwise
c_{kij}	the cost incurred when $x_{kij} = 1$

Table 2.3: Booler's Notation for the multiple class LAP

lem as one of "finding the set of working locomotives which can complete these jobs at a minimum cost". In Booler's formulation, each job may have a fixed starting time or a range of starting times. Other characteristics of the job must also be specified, such as an arrival and destination point, the duration and type of job. A range of starting times must be specified as a list of starting times with fixed intervals. The type of the job determines the compatibility with different locomotive types.

Booler's notation is given in Table 2.3 and the formulation is summarised below:

Minimise

$$z = \sum_{k=1}^{m} \sum_{i \in A_k} \sum_{j \in A_k} c_{kij}.x_{kij}. \tag{2.10}$$

Subject to

$$\sum_{j \in A_k} x_{kij} - y_{ki} = 0 \;\; (i \in A_k)\,, \;\; k = 1, 2, ..., m. \tag{2.11}$$

$$\sum_{i \in A_k} x_{kij} - y_{kj} = 0 \;\; (j \in A_k)\,, \;\; k = 1, 2, ..., m. \tag{2.12}$$

$$\sum_{k=1}^{m} y_{kj} = 1, \; j = 1, 2, ..., n. \tag{2.13}$$

where

$$x_{kij} = 0 \text{ or } 1 \text{ and } y_{ki} = 0 \text{ or } 1. \tag{2.14}$$

For the given values of the variables y_{ki}, the equations (2.10) - (2.12) form m assignment problems, one for each locomotive class. The assignment problems are linked by the n equations (2.13) which ensure that each job is allocated to a locomotive.

Booler uses a similar formulation to [23] except that a heuristic method is used whereby feasible integer components of the formulation are first found by choosing arbitrary values for variables y_{ki} and the resulting m assignment problems (2.10) - (2.12) is solved optimally. The values of y_{ki} are then tested for improvements by solving the dual of the existing model. Booler reported that the results were encouraging and perhaps worth pursuing. Booler later proposed Lagrangian Relaxation (1995) [8] as an effective alternate means to solving the same formulation.

Wright [47] built on the works of Florian *et al* and Booler by applying stochastic heuristics to solving the locomotive scheduling problem. A more elegant formulation was developed by Wright which does not consider the complex costs associated with allocating a locomotive to a single train, but rather the cost of connecting two trains i and j, c_{ij}.

Three heuristics were tested for the given formulation. Wright does not allow for a range of departure times and uses a cost function which penalizes missed departures by a constant for each midnight that is passed before the trip is made. The number of minutes taken to travel light is weighted by a constant allowing for specification of the variable cost of the schedule. Of the three algorithms tested by Wright the Simulated Annealing approach yielded the best results, outperforming his deterministic methods (Wright's variation of the standard Hungarian Algorithm [46]) on all but one problem instance.

Symbol	Definition
i, j	train indicies
t	is a locomotive type index
c_{ijt}	is the cost of train j following train i, with train i assigned locomotive type t
x_{ijt}	is a zero-one variable which is one if train j follows train i and train i is assigned locomotive type t
δ_{it}	$= 1$, if train i may legally be operated by locomotive type t $= 0$, otherwise

Table 2.4: Wright's Notation for the LAP

An exact solution is derived for the locomotive scheduling problem by Forbes *et al* [24]. This was done through the same methodology developed for the multi-depot bus scheduling problem [25]. Forbes *et al* [24] show that the bus scheduling problem with one depot and multiple depots are equivalent to the locomotive assignment problem with one and multiple locomotive types respectively.

The multiple-depot bus scheduling problem has been formulated as a specialized form of the transportation problem with side constraints, namely the multi-commodity flow problem where the commodities are locomotive types and each train is represented by two nodes, for the start and end of a trip. The problem is solved directly as an integer programming problem and the authors present their results on the same data used by Wright [47]. The notation used by Forbes *et al* is given in Table 2.4 and formulation provided below:

Minimise

$$\sum_{i,j,t} c_{ijt} x_{ijt}. \tag{2.15}$$

Subject to

$$\sum_{j,t} x_{ijt} = 1, \ \forall i. \tag{2.16}$$

$$\sum_{j} x_{ijt} - \sum_{j} x_{jit} = 0 \ \forall i, t \text{ for which } \delta_{it} = 1. \tag{2.17}$$

$$x_{ijt} \text{ and } c_{ijt} \text{ exist only when } \delta_{it}\delta_{jt} = 1. \tag{2.18}$$

Constraints (2.16) ensure that each train is assigned to a unique successor. Constraints (2.17) then ensure that a train is assigned the same locomotive type as its successor.

Two sets of directed arcs, within train i from its start node to the end node, and between trains from the end node of train i to the start node of train j, ensure that the flow of locomotive types, x_{ijt}, given constraints (2.16), service all trains only once. Forbes *et al* specify a cost function that takes into account key factors such as; running light, the cost of operating train i with locomotive type t and the cost of operating locomotive type t.

The authors note that the problem is NP-Hard in its integer form but by relaxing the integrality requirements of (2.15) the resulting linear program is solved and then transformed back to an integer solution using a branch-and-bound procedure with five prioritized strategies. To solve the relaxed version of (2.15) Forbes *et al* use the same method as Wright [47] by dropping the locomotive class restrictions, leaving a much easier assignment problem to solve.

Forbes *et al* [24] solved the model on the same data used by Wright [47]. It is interesting to note that in almost half of the problem instances tested, that the gap between the solution found by the linear relaxation and the final optimal solution was zero. In the remaining instances, the optimal was within 1.1% for all instances.

Ziarati *et al* (1997) [48] present a model driven by a case study for Canadian National Railway Company. The model is complex in that it requires multiple locomotive classes to be assigned to meet power requirements for each train as well as meeting other constraints such as: critical service stops that need to be made, time windows and power requirements for the coming week. The complexity of the problem was added to by 26 locomotive types, each with different power ratings. The solution approach presented is two fold; a *Dantzig-Wolf* decomposition [20] followed by heuristic branch-and-bound based on a depth first search.

In order to accommodate the large problem size (1249 locomotives and 1988 train segments) the authors split the week into a sequence of overlapping two-day periods and solved the resulting 7 problems individually. The authors note that the average gap between the integral relaxation and the final solution was quite high at 5.46%. Run times for each of the two-day overlaps were around 30 minutes. A three day rolling solution was attempted but with much longer run times at 4 hours per solution, resulting in around 21 hours of computation for a week's schedule.

The results were compared with the current allocations in use at Canadian National Railways and resulted in 7% and 7.5% savings in locomotives and power consumption for the 2-day and 3-day overlap scenarios respectively. The additional runtime of the solution for the 3-day overlap is well justified as a 0.5% improvement equated to $2million in annual savings at the time of writing. Several years later, Rouillon *et al* (2006) [41] present an improved branching strategy over the depth first search originally used by Ziarati *et al* and was able to find solutions which had savings of an additional $20,000 per day over the solution of Ziarati *et al* at triple the computation time.

Noble *et al* [39] present a variant of the LAP which has similar problem aspects to those already discussed [23] [48] except that in the instance solved for PTC (Public Transport Corporation) no light locomotive movements were permitted for operational reasons. The authors acknowledge the attempts of others to solve the problem but note that formulations used required complicated decompositions to solve heuristically. The authors attempted to solve the problem using a straight-forward formulation but found that the optimality gap of 30% could only be reduced to 20% by using several tactics.

The locomotive allocations are expressed as as special ordered sets (SOS1[8]) of type 1, converting the problem from selecting the number of each locomotive class to assign to a train, to a problem where the locomotive permutation from the SOS1 must be chosen. Sets are expressed in the order of their minimal coverage to satisfy the train requirement up to the maximum number of 4 locomotives permitted on any particular train. Binary variables were provided for each train service representing the number of valid permutations for that service. By reformulating the problem the resulting relaxation provided a solution within 0.5% of the optimal which was found within a

[8]SOS1 is a set of ordered variables where at most one can take on a strictly positive value.

second of CPU time.

Ahuja *et al* (2005) [1] present a richer model for solving the LAP. Their model considers the cost of splitting a consist of locomotives and having to reform them, an operational complexity which is not only time consuming but causes logistical difficulties as well. The consistency of allocations throughout the period is also taken into account, meaning that a train that departs between a origin/destination pair every day of the week at the same time should ideally be allocated the same locomotive consist. This makes it easier for the dispatching manager to remember the rules of the schedule.

The MIP model presented encapsulates the preferences of keeping locomotives together in direct train-to-train connections by adding additional arcs directly between trains that bypass ground nodes in the model which represent the *busting* of locomotives into a pool from which they are then later rebuilt into consists. The model is a multi-commodity flow problem where the locomotives are commodities with several side constraints. The authors note that for real world problem sizes of around 8800 nodes and 30200 arcs (resulting in 200000 decision variables and 67500 constraints), CPLEX 7.0 was unable to solve the integer relaxation of the problem without incorporating the weekly allocation consistency constraints.

An analysis of the scheduling requirements for the week revealed that 94% of trains being run were either 5, 6, or 7 day a week trains. The authors accordingly chose to solve the problem in two stages by assuming that all trains than run with a frequency greater than 4 by creating a representative day in which all run on that day. The solution is then translated back into a weekly problem where trains that are not required on those days specifically can then be assigned to the lower frequency trips in a second stage solve which may require that additional locomotives be added in order to generate a feasible solution.

The integer relaxation of the daily representative problem was solved in a few seconds but a feasible integer solution could not be obtained even after a substantial amount of time (72 hours). The authors suggest that the problem complexity lies with the fixed charge variables associated with light travel and connection arcs. To solve the problem, two heuristics were used to approximate the desired behavior on the train-to-train connections and light travel movements.

In the first heuristic, the constraints related to fixed charge variables were removed from the problem and the IP-relaxation solved. Costs were inflated on certain arcs to discourage consist busting. The arc with the highest number of locomotive flows on a train-to-train connection was then fixed and the resulting relaxation re-solved. If the solution is feasible and the cost increase less than a fixed parameter amount, then the arc is fixed and the process repeated, otherwise the connecting arc is rejected and the next largest tried. The iterative selection of train-to-train connections stops when either there are none remaining to choose from or the desired number has been reached (a parameter specified in line with the business rules).

To solve the problem of light locomotive movements, the authors first solve a minimum-cost flow problem (a reduced version of the larger space-time network) by ignoring the time dimension to establish candidate arcs between pairs of locations (arcs) where light movements may be required to restore balance. If a positive flow of light locomotives is found on an arc then candidate arcs are introduced into the main space-time model at 8 hourly increments. As with the first heuristic, the fixed charge variables are ignored and the relaxed IP solved with all candidate light arcs. Any arcs with zero locomotives after the first solve are deleted from the model. The smallest flow is then removed and the resulting LP relaxation solved again. If the cost increases by more than a fixed parameter the arc is fixed in the solution, failing which, it is permanently deleted. The process repeats until all light arcs have been examined.

A final MIP is solved sub-optimally to determine which locomotives are active and which will deadhead as a result of the current allocations. Although finding an optimal solution was not possible in a reasonable amount of time, the authors noted that *very good* solutions were found early on in the CPLEX run and as a result they could terminate the procedure without confirming optimality in favor of commencing a *very large-scale neighborhood* (VLSN) *search algorithm.*

VLSN is done through CPLEX by defining the neighborhood; for each locomotive type as the set of moves for which the schedule for all other locomotive types does not change. The stopping criterion is thus when for all locomotive types within the current solution, no local change can be made that results in a better solution keeping all other locomotive type schedules fixed. Even though this produces a large MIP, the authors report that a local optimum is converged to within a few seconds.

Once the set of allocations has been obtained for all high frequency trains, the remaining daily problem can be solved. As already mentioned, the representative daily problem (of the whole week) is transformed back into a week. The lower frequency trains are then added to the solution iteratively by type, starting with the most available in descending order, each time solving a single-commodity flow problem on the full weekly space-time network. Finally, VLSN is again applied to the weekly space-time network for each locomotive type (solving, again, the single-commodity flow problem) until no improvement can be found holding all other locomotive type schedules fixed.

The results of a study conducted at CSX Transportation (a major U.S railroad company) are presented in which over 400 locomotives[9] could be saved over the solution provided by the in house software used at CSX. The CSX solutions did not consider any light locomotive movements and had much higher levels of consist busting. Ahuja founded Innovative Scheduling in 2000, a company specializing in the operations research domain by developing custom solutions, and by 2007 had designed three locomotive-orientated decision support tools for CSX.

Vaidyanathan *et al* (2007) [45] extended their work using a subtle modification to the model presented in 2005 [1]. The formulation used is very similar except instead of modeling each individual locomotive as a commodity in the multi-commodity flow formulation they model the consist in the network. This has several advantages, namely the reduction of consist busting (a preferred characteristic) as well as a massive reduction in decision and constraint variables, allowing previously unsolvable instances (due to a failure to converge) to be solved in a matter of minutes. The authors argue that although this reduced problem complexity may produce solutions which are at a greater cost than their previous formulation, the robustness of the schedules produced and ease of implementation are much higher.

Consists to choose from are added as decision variables to the model with at most p of each consist being allowed in the model. From an implementation perspective, the number and variation of consists should be kept lower as it makes for a more practical and easier schedule to implement. Other constraints such as locomotive region restrictions as well as special locomotive unit requirements within the consist (for signaling systems) are also taken into account.

The authors present the details of several case studies where key variables were modified such as train speed, connection time and transport volume. The impact on the cost of the schedules produced were measured and presented.

[9]CSX had a fleet of 3316 locomotives at the time

Chapter 3

Problem Formulation

The following sections will detail the data requirements and model structure for solving the Bulk Freight Train Scheduling Problem (BFTSP).

3.1 Model Formulation

The BFTSP and all associated constraints are formulated as a multi-commodity flow problem, where locomotive and wagon sets are the commodities, using an MIP model similar to that presented by Ahuja *et al* [1] [45]. Two major differences in the model presented here are firstly, not all train arcs have to be selected and secondly, wagons and locomotives are modeled as separate commodities. Constraints govern the minimum number of trains that need to be run in the period. A number of additional constraints are added to ensure that the resulting selection of slots is executable.

We start by defining the *space-time* network which is used to model the flow of resources. The network consists of *arcs* and *nodes*. We denote the set of all nodes to be N and all arcs to be A. Each arc has an associated cost C_{ij}, indicating the cost of moving from node i to node j.

As presented by Ahuja *et al*, we use five types of nodes, *ground nodes*, *train departure nodes*, *train arrival nodes*, *source nodes* and *sink nodes*. We use three types of arcs in this model; *train arcs*, *ground-train arcs* and *ground-ground arcs*. Each arc is defined between two nodes. As the names suggest, train arcs, denoted $TrArc$, link a train departure and train arrival node (or vice versa), ground train arcs connect ground nodes to train departure or arrival nodes and ground-ground arcs connect two ground nodes, denoted $GTrArc$ and $GGArc$ respectively.

It is worth emphasizing that each node i has a specific time and location associated with it, denoted N_i^t and N_i^{loc}. Ground nodes represent a

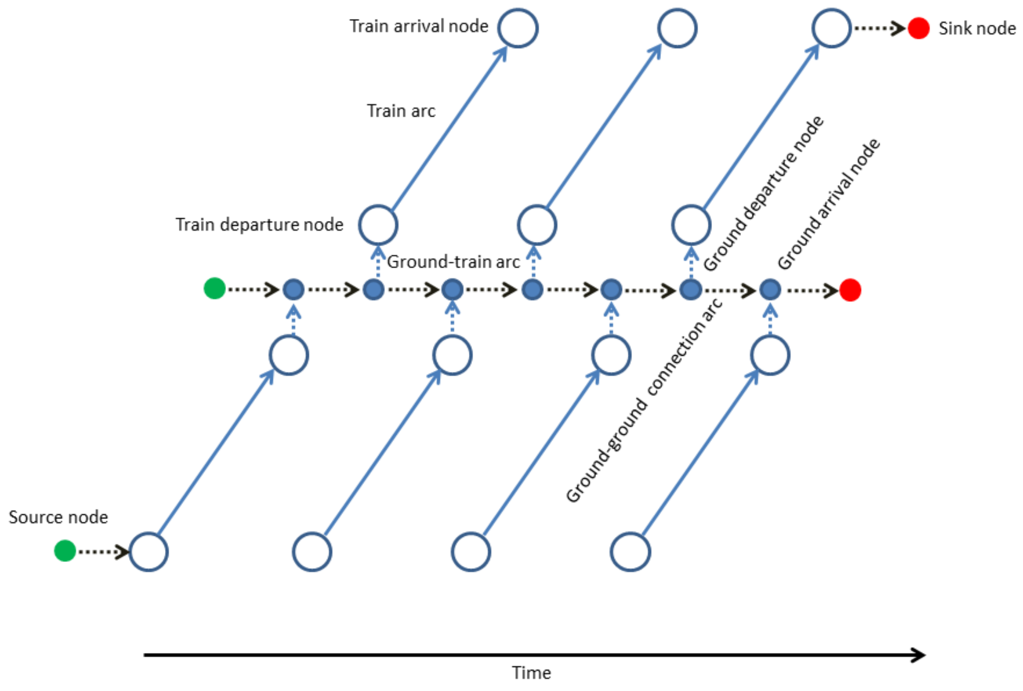

Figure 3.1: Space-time network example

collection of flows through the network (such as in a yard or hub) whereas ground departure and ground arrival nodes are specific to a particular train. Ground nodes therefore directly model the capacity of locations and the ground departure and arrival nodes serve to either decrease or increase the total commodities in these absorbing locations.

Figure 3.1 provides an illustration of a sample space-time network. Each of the nodes has a location and time in the network. The horizontal axis in Figure 3.1 represents time and the vertical axis a relative ordering of connectivity of nodes. Arcs are all directed and no arcs travel back in time, unlike the models presented in [1] and [45] as we do not wish to create a cyclic schedule, the reason for the addition of source and sink nodes.

Source and sink nodes indicate the starting and potential end positions of resources in the network. The sink nodes provide a selection of exit points from the network for resources. Source nodes and sink nodes are the only nodes on the network that do not have both incoming and outgoing arcs. Source nodes have exactly one outgoing arc and sink nodes have exactly one incoming arc.

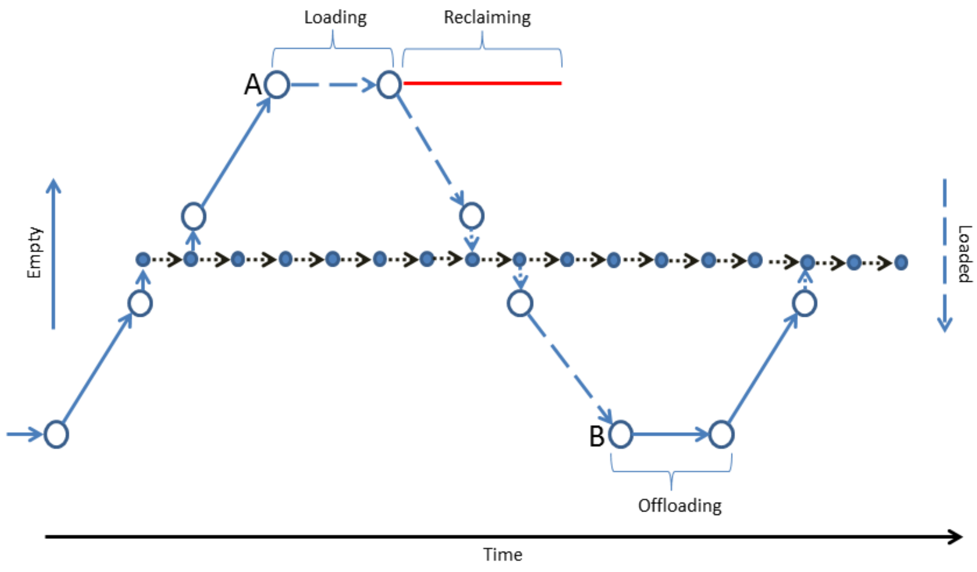

Figure 3.2: Empty wagon set flow in the space-time network

We adapt the models presented in the literature [1] [45] which deal with the locomotive allocation problem to extend to wagon set allocation. Locomotives do not change state (empty to loaded or loaded to empty) in the course of the schedule. Wagons may change state several times. In order to still correctly model the flow of both empty and loaded wagons in the network, each wagon will be represented by two instances, one loaded and one empty. Node constraints will enforce that the same resource must be used throughout the network, except at particular nodes where the opposite state will be required.

Figure 3.2 gives an illustration of the flow of an empty wagon set in the network. It is important to note that in this diagram there are two distinct resources that are being used, since empty wagons sets and loaded wagon sets are modeled separately. Specific arrival nodes carry the constraint of enforcing a state switch to the alternate resource, labeled A and B in the diagram. Wagons change from an empty to loaded state at A and from the loaded to empty state at B. All nodes in Figure 3.2 except A and B have the constraint specified by Equations (3.6) and (3.7). A and B are subject to Equations (3.8) and (3.9) which enforce the resource switching. These equations are elaborated on in the constraints section of this chapter.

Locomotives do not contribute to the capacity constraints since they require very little space to be placed. The sum of wagon set sizes over any single ground node are considered in order to ensure capacity constraints are adhered to. When creating the space-time network, ground-ground nodes are sorted in ascending order by their times for each location and are linked sequentially using *GGArc*. This ensures that all flows within a site have to pass through the same arc from one time point to the next. This can be seen in both Figures 3.1 and 3.2 as all ground nodes and linked sequentially in increasing time order at the same location. The ground nodes enable wagon capacity constraints to be placed on a location.

Symbol	Definition
i, j	node indices
σ	site indicies
N	the set of all nodes (excluding source and sink nodes)
N_i^t	the time associated with the i^{th} node, $t \in \mathbb{Z}^+$
N_i^σ	the site σ associated with the i^{th} node
k	train configuration index, representing all distinct valid train combinations
A	the set of all arcs, $a \in A \Rightarrow a = (i, j, k)$
c_{ijk}	is the cost associated with the k^{th} type of train between nodes i and j
$I[i]$	the set of arcs entering node i
$O[i]$	the set of arcs leaving node i
S_σ	the σ^{th} site
S_σ^P	the processing time[1] at the σ^{th} site,$\in \mathbb{Z}^+$
S_σ^R	the reclaiming time at the σ^{th} site,$\in \mathbb{Z}^+$
S_σ^B	the number of parallel processes permitted at the σ^{th} site, $\in \mathbb{Z}^+$
S_σ^e	the maximum empty wagon capacity at site σ, $\in \mathbb{Z}^+$
S_σ^l	the maximum loaded wagon capacity at site σ, $\in \mathbb{Z}^+$
S_σ^{tot}	the maximum wagon capacity at site σ, $\in \mathbb{Z}^+$
S_σ^d	the demand requested by site σ
S_σ^{load}	the amount loaded at site σ in tonnes per train
n	the number of locomotives
L	the set of locomotives $L = \{l_1, l_2, ..., l_n\}$
m	the number of wagon sets
W_e	the empty wagon sets $W_e = \{w_{e_1}, w_{e_2} ..., w_{e_m}\}$
W_l	the loaded wagon sets $W_l = \{w_{l_1}, w_{l_2} ..., w_{l_m}\}$
$\|W_m\|$	the size of wagon set m in number of wagons
W_m^{cap}	the capacity of wagon set m in tonnes
S_p^c	an ordered set of all nodes associated with sites in the p^{th} collection, defined in the **Slots** constraints section
M_{pq}^{slots}	the maximum number of consecutive slots between site collections p and q, $\in \mathbb{Z}^+$, defined in the **Slots** constraints section
H_{min}	the minimum headway required between trains, $\in \mathbb{Z}^+$

Table 3.1: Notation used for the BFTSP

3.1.1 Decision Variables

There are four types of decision variables in the model; namely, the arc that is selected for activation (3.1), the locomotive set (3.2) assignments to arcs and finally, empty (3.3) and loaded (3.4) wagon set assignments to arcs. These resource assignments to the particular arcs (or slots) cause the greatest complexity in the model as the greatest number of constraints require validation around these decision variables. We denote the decision variables as follows:

$$TrArc_{ijk} = \begin{cases} 1 & \text{if the } k^{th} \text{ arc is active between nodes } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

$$l_{xijk} = \begin{cases} 1 & \text{if locomotive set } x \text{ is assigned to } TrArc_{ijk} \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

$$w_{e_{yijk}} = \begin{cases} 1 & \text{if empty wagon set } y \text{ is assigned to } TrArc_{ijk} \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

$$w_{l_{zijk}} = \begin{cases} 1 & \text{if loaded wagon set } z \text{ is assigned to } TrArc_{ijk} \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

3.1.2 Constraints

Resource Flow

In order to ensure that resources flow uniquely through the network from arc to arc, we add a constraint that the flow at each node is balanced for every locomotive set, given in Equation (3.5). Each resource is treated as a unique commodity in the multi-commodity formulation. It should be noted that shorthand is used in this context such that: $a \Leftrightarrow (x, i, j, k)$

$$\sum_{a \in I[i]} l_a = \sum_{a \in O[i]} l_a \; \forall i \in N \text{ and } a \Leftrightarrow (x, i, j, k). \tag{3.5}$$

Modeling the wagon flows requires some additional attention. Wagon sets change states at certain nodes. We define the set of nodes at which wagons change from an empty to loaded state as N_{EL} and from a loaded to empty state as N_{LE} where $N_{EL} \cap N_{LE} = \emptyset$. All other nodes where wagons do not change state fall into N_o such that $N_o \cap (N_{EL} \cup N_{LE}) = \emptyset$ and $N_o \cup N_{EL} \cup N_{LE} = N$.

$$\sum_{a \in I[i]} w_{e_a} = \sum_{a \in O[i]} w_{e_a} \; \forall i \in N_o \text{ and } a \Leftrightarrow (y, i, j, k). \tag{3.6}$$

$$\sum_{a \in I[i]} w_{l_a} = \sum_{a \in O[i]} w_{l_a} \; \forall i \in N_o \text{ and } a \Leftrightarrow (y, i, j, k). \tag{3.7}$$

Flow constraints (3.6) and (3.7) govern the flow constraints over wagons in the same way as for locomotives where no state changes do occur.

$$\sum_{a \in I[i]} w_{e_a} = \sum_{a \in O[i]} w_{l_a} \; \forall i \in N_{EL} \text{ and } a \Leftrightarrow (z, i, j, k). \tag{3.8}$$

$$\sum_{a \in I[i]} w_{l_a} = \sum_{a \in O[i]} w_{e_a} \; \forall i \in N_{LE} \text{ and } a \Leftrightarrow (z, i, j, k). \tag{3.9}$$

Flow constraints (3.8) and (3.9) govern the flow constraints over wagons where state changes occur. The constraint enforces that the same wagon set is used but from the the opposite set.

Wagon configuration

There are multiple allowable wagon configurations between nodes i and j, indexed by k. Each configuration has its own total wagon requirement given by Λ_{ijk}. The constraint across all train arcs is given in (3.10).

$$\sum_{y=1}^{m}(w_{e_{yijk}} + w_{l_{yijk}}) = \Lambda_{ijk} \; \forall i, j, k. \tag{3.10}$$

$\Lambda_{ijk} = 0$ for configurations related to light locomotive movements.

Slots

Each p^{th} site collection S^c_p gives the sites (and hence ordered nodes) which represent some geographic constraints. For example, only one departure may occur at a time between S^c_p and S^c_q but another simultaneous departure could occur between S^c_p and S^c_{qq} (perhaps a northbound and southbound train both leaving from S^c_p). Figure 3.3 gives an example of site collections used in the test data given in Section 4.2. In this example trains may arrive or depart at S^c_2 from S^c_1 or S^c_3 simultaneously.

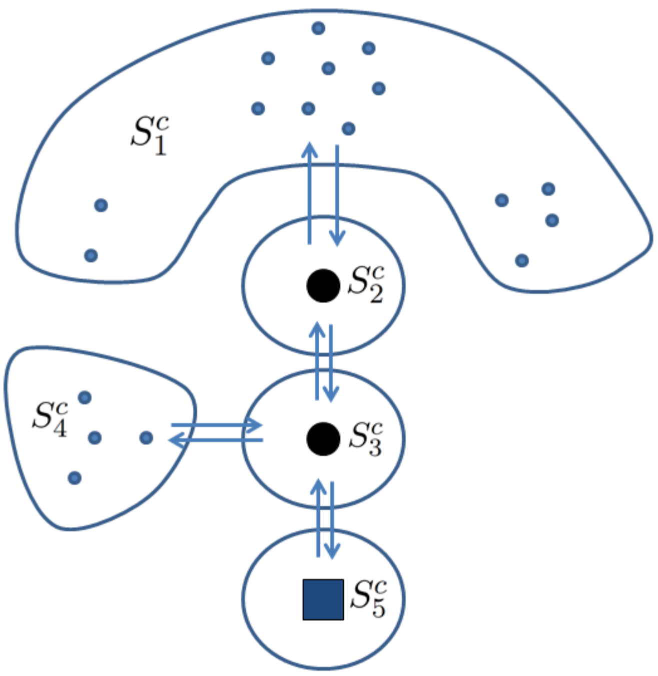

Figure 3.3: Example of site collections for sites and hubs

The constraint for the maximum number of slots, M_{pq}^{slots}, that can be consecutively used, is given in Equation (3.11) and is expressed in terms of the slot collection pairs (S_p^c, S_q^c).

$$\sum_{b=i}^{i+M_{ij}^{slots}} \sum_{\forall k \in TrArc_{bj}} TrArc_{bjk} \leq M_{ij}^{slots} \; \forall (i,j) \in (S_p^c, S_q^c). \tag{3.11}$$

The slot usage constraint only works when used in conjunction with Equation (3.12) since we need to ensure that each $TrArc$ between i and j is used at most once across all possible configurations k. This prohibits the simultaneous departure of two trains with different configurations traveling to the same destination.

$$\sum_{\forall k \in TrArc_{ij}} TrArc_{ijk} \leq 1 \; \forall i, j \in N. \tag{3.12}$$

Windows

A site exclusion window can easily be translated to the set of nodes that fall into that window for the time period at a particular site. Once the set of nodes has been identified, constraints can be placed on the arcs restricting their use. For α^{th} exclusion window at site σ, $SW_{(\sigma,\alpha)}$, with start $SW_{(\sigma,\alpha)}^s$ and end $SW_{(\sigma,\alpha)}^e$, we have Equation (3.13) constraining all departures from the site during that time.

$$\begin{gathered} TrArc_{ijk} = 0 \\ \text{where } N_i^t \in \left[SW_{(\sigma,\alpha)}^s, SW_{(\sigma,\alpha)}^e\right] \text{ and } N_i^\sigma = SW_\sigma, \; \forall i, j, k, \sigma, \alpha, . \end{gathered} \tag{3.13}$$

Similarly for arcs which lead into site σ we also set their decision variables to zero where the arrival node (j) falls within the window. We extend this further to cover the processing time (S_σ^P) at the site. The reason for this is that we would not want a resource arriving and then being loaded if the site is closed. A track occupation window would exclude this additional offset. Equation (3.14) gives the set of arcs that are excluded based on their arrival nodes.

$$\begin{gathered} TrArc_{ijk} = 0 \\ \text{where } N_j^t \in \left[SW_{(\sigma,\alpha)}^s - S_\sigma^P, SW_{(\sigma,\alpha)}^e\right] \text{ and } N_j^\sigma = SW_\sigma, \; \forall i, j, k, \sigma, \alpha, t. \end{gathered} \tag{3.14}$$

Loading, processing and offloading

When resources arrive at a particular site they may perform one of three possible actions; Loading, processing and offloading. Loading and offloading both result in resource switching for wagons. Processing of wagons and all locomotive transactions do not alter their state. Resource state changes are handled as part of the constraints on the nodes. As a result we will keep discussion the processing time, S_σ^P, for site σ which may cover any of the three possible actions which may take place.

The minimum amount of time which must be spent at a particular location is handled by placing a constraint on the resource allocation to arcs entering and leaving a location. For all arcs entering a location at node N_i^σ, we define the set of all outgoing arcs which fall within the processing time at the same location to be A_i^0. Given that any one of the incoming arcs may be selected, we wish to ensure that none of the outgoing arcs is selected until the required time has passed for a resource. We require this constraint to be applied across the resource allocation variables for locomotives and both empty and loaded wagon sets together since resource switching could take place.

The constraints on the locomotive dwell times are given in Equation (3.15) below:

$$\sum_{\forall (j,k) \in A_i^0} l_{xijk} \leq 1 \text{ for } x = 1, ..., n \text{ and } \forall i \in N. \tag{3.15}$$

The constraint that covers both loaded and empty wagon sets is given in Equation (3.16):

$$\sum_{\forall (j,k) \in A_i^0} (w_{e_{yijk}} + w_{l_{yijk}}) \leq 1 \text{ for } y = 1, ..., m \text{ and } \forall i \in N. \tag{3.16}$$

Equations (3.15) and (3.16) ensure that resources cannot be allocated to outgoing arcs within the prescribed processing time. It should be obvious that resource specific processing times can easily replace S_σ^P to define a new set of restricted outgoing arcs A_i^0. This enables locomotives to have a different processing time to wagons when arriving at certain sites. This leads to improved efficiencies when wagons have very long dwell times at certain sites as the locomotives can perform other tasks and return to the wagons at a later stage.

Reclaiming

Each train that is sent to a loading site results in a reclaiming time constraint, i.e. another train may not be sent until the reclaiming time has elapsed after the processing time of the first train. Some sites may have multiple dedicated loading sidings and can process the loading of trains in parallel as a result. We refer to this maximum number of parallel tasks as S^B_σ. The constraint is given in terms of the arrival node j for all train arcs in Equation (3.17) since we wish to restrict the arrival of trains at a siding regardless of their origin.

$$\sum_i \sum_{v=N^t_j}^{N^t_j+S^p_j+S^R_j} \sum_k TrArc_{ivk} \leq S^B_j \ \forall j, t. \tag{3.17}$$

Headway

Headway is required from a safety perspective to ensure that trains do not depart or arrive within a minimum required interval, H_{min}. For simplicity, we will assume that the design of the outgoing slots has already taken care of the constraints on departure nodes, i. However, we need to ensure that trains arriving at destination j are meeting the headway requirements, given in Equation (3.18).

$$\sum_{y=1}^{m} TrArc_{yjk} \leq 1 \text{ where } y \in \left[N^t_j, N^t_j + H_{min}\right] \ \forall i, j, k. \tag{3.18}$$

Region

The region constraint applies to locomotive or wagon set assignments to train arcs. We define two new sets of indicator variables in Equations (3.19) and (3.20) for locomotive and wagon movement restrictions.

$$l_{n,ij} = \begin{cases} 1 & \text{if locomotive } l_n \text{ may travel between nodes } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases} \tag{3.19}$$

$$w_{m,ij} = \begin{cases} 1 & \text{if wagon } w_m \text{ may travel between nodes } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases} \tag{3.20}$$

Equation (3.21) ensures that when travel between any pair of points is prohibited by a resource that the assignment variable will remain zero. It follows that other kinds of constraints (such as resource specific constraints) can be represented by a similar constraint.

$$l_{xijk} \leq l_{x,ij} \ \forall i, j, k \text{ and } x = 1, ..., n. \tag{3.21}$$

$$w_{e_{yijk}} \leq w_{y,ij} \ \forall i, j, k \text{ and } y = 1, ..., m. \tag{3.22}$$

$$w_{l_{zijk}} \leq w_{z,ij} \ \forall i, j, k \text{ and } z = 1, ..., m. \tag{3.23}$$

Several different types of real world constraints such as axle size constraints, resource class restrictions, AC/DC power availability and operational workings can be modeled through such equations.

Site Capacity

For each Site S_σ we define the maximum number of empty, loaded and total wagons permissable in that site: S^e_σ, S^l_σ and $S^{tot}_\sigma \in \mathbb{Z}^+$

It follows that we need to restrict the flow of commodity across all ground nodes, denoted N_G. Due to the flow constraints already provided in (3.6) and (3.7) we can arbitrarily restrict either the outgoing or incoming flow as given in (3.24) and (3.25)

$$\sum_{a \in O[i]} \|w_{e_a}\|.w_{e_a} \leq S^e_\sigma \text{ where } S_\sigma = N^\sigma_i, \ \forall i \in N_G. \tag{3.24}$$

$$\sum_{a \in O[i]} \|w_{l_a}\|.w_{l_a} \leq S^l_\sigma \text{ where } S_\sigma = N^\sigma_i, \ \forall i \in N_G. \tag{3.25}$$

$$\sum_{a \in O[i]} (\|w_{e_a}\|.w_{e_a} + \|w_{l_a}\|.w_{l_a}) \leq S^{tot}_\sigma \text{ where } S_\sigma = N^\sigma_i, \ \forall i \in N_G. \tag{3.26}$$

ECP

The ECP constraints between resources and sites are quite specific. They exist in four forms:

- ECP Wagons may not be moved to Non-ECP sites
- Non-ECP Locomotives may not be coupled with ECP Wagons
- ECP Wagons can only be coupled with other ECP wagons
- Non-ECP Wagons can only be coupled with other Non-ECP Wagons

The first item above is easily dealt with in the creation of the train arcs by simply not creating ECP arcs for sites which cannot support ECP. In Equation (3.1) we have the k^{th} arc between sites i and j. Each of the k arcs represents a feasible resource mix in terms of the ECP constraints given above. We define a variable for each k^{th} arc which indicates if the arc is an ECP or arc or not, given in Equation (3.27).

$$TrArc^{ecp}_{ijk} = \begin{cases} 1 & TrArc_{ijk} \text{ is ECP} \\ 0 & \text{otherwise.} \end{cases} \tag{3.27}$$

We then apply constraints to the resource allocation decision variables (3.2) - (3.4) to ensure feasible combinations are generated. We extend the ECP property to locomotives and wagons and denote the respective sets of ECP locomotives and wagons L^{ecp} and W^{ecp} and non-ECP sets L^{necp} and W^{necp} where $L^{ecp} \cap L^{necp} = \emptyset$, $W^{ecp} \cap W^{necp} = \emptyset$, $L^{ecp} \cup L^{necp} = L$ and $W^{ecp} \cup W^{necp} = W$:

$$w_{e_{yijk}} + w_{l_{yijk}} = 0 \text{ where } y \in W^{ecp} \text{ and } TrArc^{ecp}_{ijk} = 0, \; \forall i, j, k. \tag{3.28}$$

$$w_{e_{yijk}} + w_{l_{yijk}} = 0 \text{ where } y \in W^{necp} \text{ and } TrArc^{ecp}_{ijk} = 1, \; \forall i, j, k. \tag{3.29}$$

$$l_{xijk} = 0 \text{ where } x \in L^{necp} \text{ and } TrArc^{ecp}_{ijk} = 1, \; \forall i, j, k. \tag{3.30}$$

While ECP and non-ECP wagons must remain mutually exclusive as per Equations (3.28) and (3.29). ECP locomotives can be used with non-ECP wagons which is why we only have one constraint for locomotives, Equation (3.30) which prohibits the use of non-ECP Locomotives with ECP wagons.

Demand

The final constraint on the decision variables is that of meeting the requested demand for each site, S^d_σ. Due to the structure of the space-time network, we can easily translate this requirement to be that of having a minimum number of incoming arcs selected for each site, given in Equation (3.31).

$$\sum_i \sum_{j'} \sum_k TrArc_{ij'k} \geq S^d_\sigma \text{ where } j' \in \left\{ j | N^\sigma_j = S_\sigma \right\} \; \forall \sigma. \tag{3.31}$$

3.1.3 Objective

The objective is to simultaneously select train arcs and allocate resources in such a way as to maximise the resource utilization by minimising the dwell time of wagon sets, resulting in the maximum throughput for the network in the minimum amount of time. Since wagon sets are not active resources, a good locomotive allocation will result in good utilization of wagons. The objective function simply consists of the sum of all arcs over which resources flow. We specify the cost of each arc, c_{ijk}, as:

$$c_{ijk} = \tau_{ijk}.(N_j^t - N_i^t). \tag{3.32}$$

where τ_{ijk} is the inflation factor for the k^{th} configuration between nodes i and j.

Light locomotive movements have a $\tau > 1$ as we want to discourage their use in the final solution unless the benefits are sufficiently large. τ_{iik} may also be increased to penalize dwell times at hubs or offloading sites (between ground nodes), encouraging more efficient resource utilization.

Another aspect of achieving maximum throughput is that of measuring the amount that is loaded at different sites. Lane selection can result in larger wagons being sent to a location where smaller ones would have sufficed. There is a trade off between the increased resource utilization and the potential dwell time that would be incurred to ensure a perfect allocation. The amount of wagon under-utilization will be weighted with the β and the final objective function defined as:

Minimise $f(x)$ where :

$$f(x) = \sum_i \sum_j \sum_k \left(TrArc_{ijk}. \left(c_{ijk} + \beta \sum_{y=1}^{m} w_{e_{yijk}}. \left(|w_{e_m}|.w_{e_m}^{cap} - S_{N_j^\sigma}^{load} \right) \right) \right). \tag{3.33}$$

Chapter 4

Test Data

Three data sets are configured and the algorithms given in Table 4.1 applied to the relevant data set. Data configuration in terms of locomotive region constraints, sites, allowable slots, headway requirements, loading, processing, offloading and reclaiming times remain unchanged throughout the data sets. The number of resources available in each scenario (locomotive and wagon sets) as well as total demand and demand mix will vary. The resources will vary in proportion to the increased demand, the final *Large* data set giving a reasonable representation of the amount of data required to complete a practical real world schedule for one week.

Data Set Size	**Exact Search**	**Heuristic B&B**	**Genetic Algorithm**
Small (4 loads)	Y	Y	Y
Medium (30 loads)	N	Y	Y
Large (204 loads)	N	N	Y

Table 4.1: Test Data Set Summary

Figure 4.1 gives the relative position of all sites being considered for Problem Type 2. The thickness of the connecting arcs between sites is an indicator of the number of allowable resource configurations in term of possible wagon set combinations. The associated travel times T_{ij} for all lanes are given in Appendix A.5.

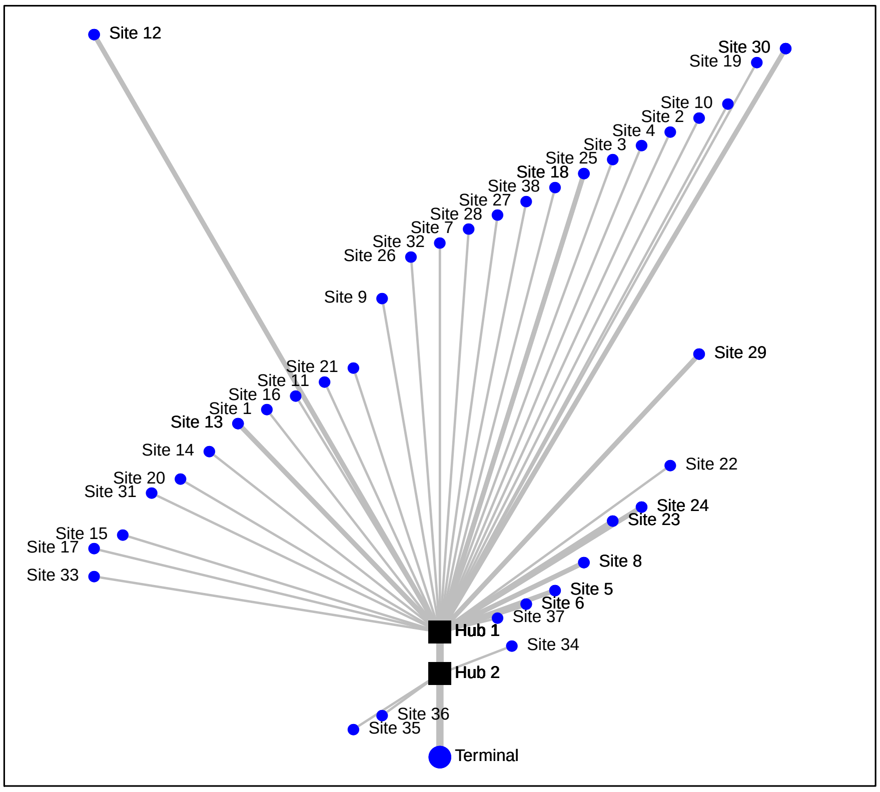

Figure 4.1: Site Lane connectivity and relative position

Concept	**Description**
Site	38 loading sites, 2 hubs, 1 terminal
Region	8 regions in total. Each hub and terminal having their own region
Slot	Slots between pairs of sites are given in A.1. All slots start at midnight
Locomotive Set	50 locomotive sets in total. ECP and class properties are given in A.2
Locomotive Set Region Inclusion	The inclusions for the 8 regions is given in A.2. A diagrammatic representation is given in Figure 4.2
Wagon Set	The 84 wagon sets and attributes given in A.3. Two wagon classes are used, denoted J and S, with 83 and 58 ton capacities respectively
Headway	A 15 minute headway between trains is required when arriving or departure from any site
Wagon configuration	The available wagon configurations between Sites i and j are given in A.4. 103 pairs in total
Travel Time	The 80 distinct travel times (T_{ij})between sites given in A.5
Site Window	9 Sites with 19 windows (for the period of one week) are given in A.6

Table 4.2: Data attribute summary

4.1 Exact Search Results

A program was written in *C#* [1] to handle the building of a schedule and the exact search. A recursive depth-first strategy was used to traverse the solution space, with constraints being validated at each node of the tree to ensure only feasible solutions are considered. The results of 4 runs, each with more loads are given in Table 4.3. The loads chosen in the small problem set can be found in A.7.

Number of Loads	**Nodes Pruned**	**Leaves Inspected**	**Objective Value**
1	4	1	1 803
2	591	517	88 087
3	10 946	4 360	91 385
4	234 471	70 516	98 522

Table 4.3: Exact search statistics

An exponential function of the form $f(n) = \beta^n$ was fitted to the data

[1] *C#* is an object orientated language which uses the Microsoft .Net Framework.

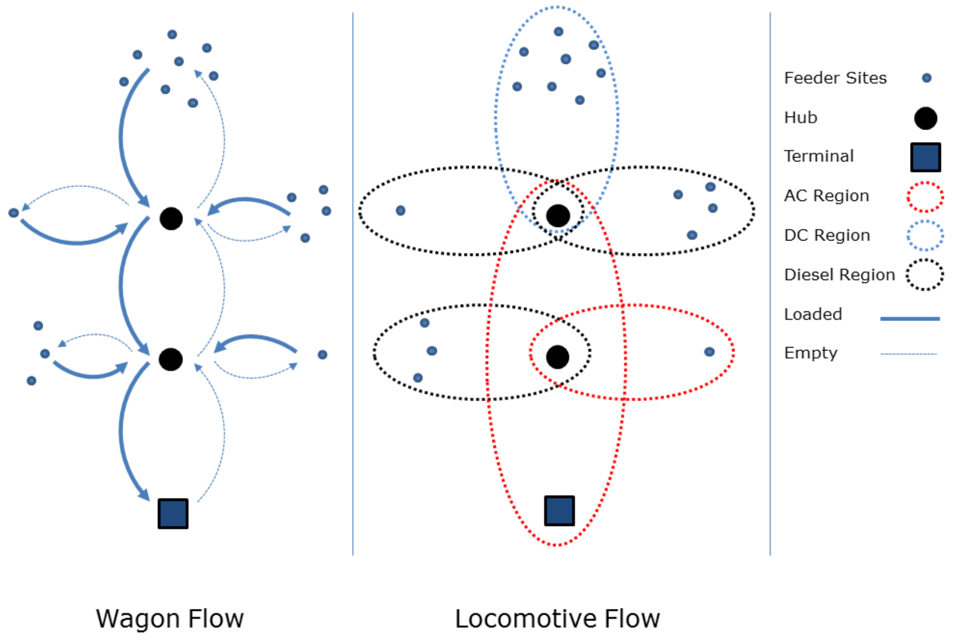

Figure 4.2: Resource flow across all sites

of leaves inspected to estimate the growth in problem complexity for small values of n. A significant parameter estimate was obtained where $\beta = 16.3$. The exact search for 4 loads took approximately 90 minutes to solve. Extrapolating suggests that 5 loads would take around 24 hours to find the optimal solution. This is clearly not a viable method for realistic problem sizes. In addition, the function $f(n)$ will only hold for small values of n since there is a factorial component, related to the order in which loads are completed, which we would expect to dominate for large enough values of n.

4.2 Heuristic Branch and Bound

A common problem with depth-first strategies is that a disproportionately high amount of search time is spent searching the bottom portion of the tree. As the tree size grows (exponentially in terms of number of leaves to explore), it becomes less likely that initial decisions in the search are modified. Understanding the future cost of initial decisions made in the tree can be very difficult, especially when dependencies exist between decision variables.

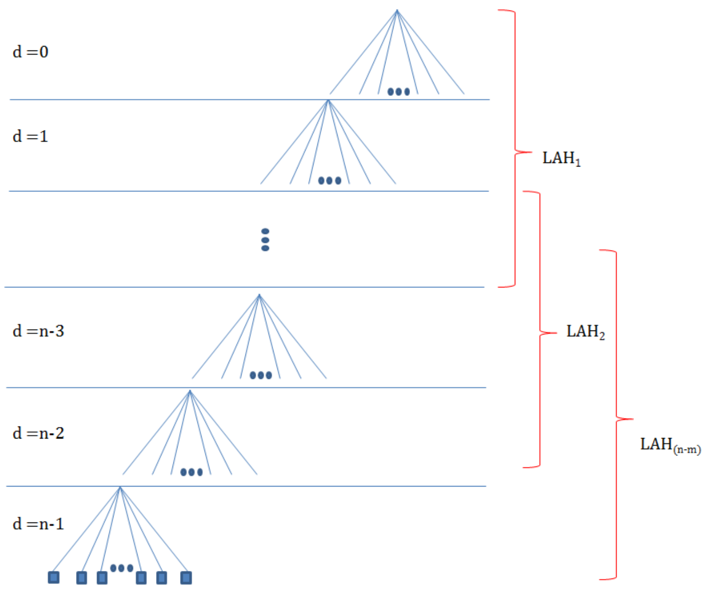

Figure 4.3: Depth First Search with a 'Look Ahead Heuristic'

We describe a 'Look Ahead Heuristic' (LAH) [35] which searches the immediate sub-tree from the current node and finds the best local decision. The depth of the sub-tree is limited to ensure that the "look ahead" is fast. We limit the depth of the local search by parameter a where $1 < a < n-1$. The case where $a = 1$ can be considered a greedy search and $a = n-1$ an exact search, emphasizing the trade-off between computation time and solution quality.

Once the best path for the sub-tree has been found, the first decision of the path is traversed and the next sub-tree searched as a result of the new decision made. This is repeated until the maximum depth of the tree is reached, at which point the solution is returned. A fundamental assumption of this approach is that two decisions, adequately far from one another in the tree, are sufficiently independent.

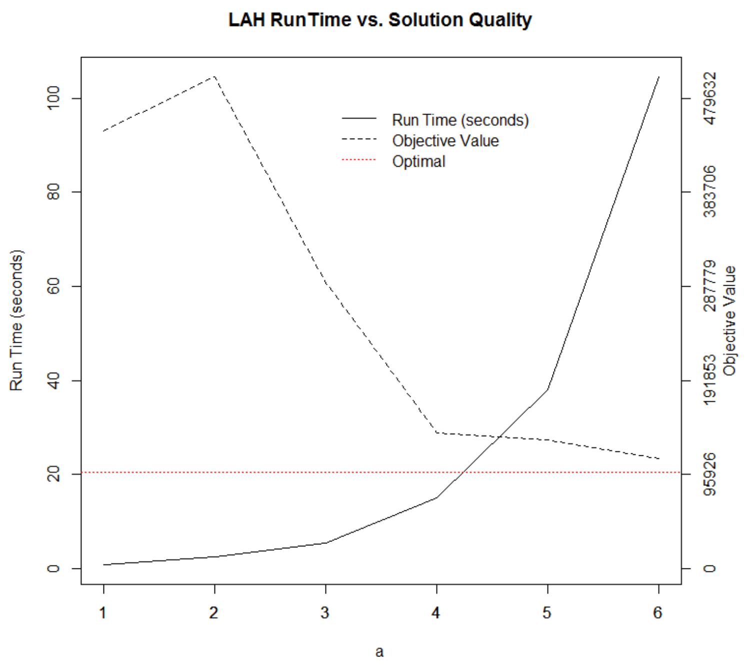

Figure 4.4: LAH Computational Results, small test data set

The results of the *LAH* heuristic are given in Figure 4.4. The largest value of a in the *LAH* managed to get to within 12.5% of the known optimal solution in 2% of the time of the exact search. Interestingly, for low values of a the heuristic performs very badly, making immediately appealing decisions without taking account of the long term cost implications. The maximum depth of the tree in this instance is 45. The nature of the long term dependence between tree nodes makes this heuristic less useful than originally anticipated, but still very interesting in terms of understanding the extent of dependence between decision variables.

As with the depth first search, as the problem complexity increases, the amount of work required at each node of the tree also increases, and total work increases exponentially. This means that although a reasonable bound was obtained quickly for a small problem such as this - for bigger problems a will have to be decreased to bring the running time down to a reasonable value and hence the upper bound is also increased. We would argue that this heuristic does not scale well enough for large problem sizes and is thus not useful in practice.

4.3 Genetic Algorithm

Genetic Algorithms (GA) are a meta-heuristic approach that use a population of candidate solutions to search the solution space. The operations performed by the GA take inspiration from natural processes and fall into three categories, namely:

- Selection
- Crossover
- Mutation

GAs do not require the assumptions of continuity, differentiability or unimodality and as a result have been shown quite robust in several domains [22]. The Simple Genetic Algorithm (SGA) as described by Goldberg [26] uses a binary chromosome encoding which is then mapped to a real number using the formula in equation (4.1).

$$\begin{gathered} x = a + \frac{b-a}{2^l - 1} \sum_{n=1}^{l} y_n 2^{n-1} \ \epsilon [a, b] \\ \text{where } y_n \ \epsilon \{0, 1\} \ \forall n \ \epsilon \{1, 2, ..., l\}. \end{gathered} \tag{4.1}$$

Greater values of l (the number of bits per variable) will increase the precision of the encoding, the size of l should be chosen to be small enough to sufficiently model the domain. In a classical GA setting, an array of real values is produced from the binary encoding which is then used as input to an objective function. The number of parameters in the objective function determines the size of the chromosome required. The objective function returns a real valued number which determines the *fitness* of the candidate solution.

When comparing different solutions the *fitness* is used as the criterion on which selection takes place. There are several types of selection that could be used, the most common being “roulette-wheel selection” and “tournament selection”. The selection operator is used to choose chromosomes of superior quality for use in the subsequent population. Chromosomes selected are then combined using the crossover operator in an attempt to ‘blend’ genetic material between chromosome pairs. Tournament selection was chosen as the selection operator (with a tournament size of 2) as it provides slower convergence and less emphasis on superior individuals, this assists on exploration of the solution space.

The uniform crossover operator is a common choice when working with binary strings for performing crossover. A new uniform random variable is generated, R, the same length as the original chromosome. The probability of performing crossover at point i in the chromosome, also called the crossover rate, is equal to p_c. If $R_i \leq p_c$ then the bits at position i are swapped between two parent solutions. This process produces two new offspring which form part of the new population.

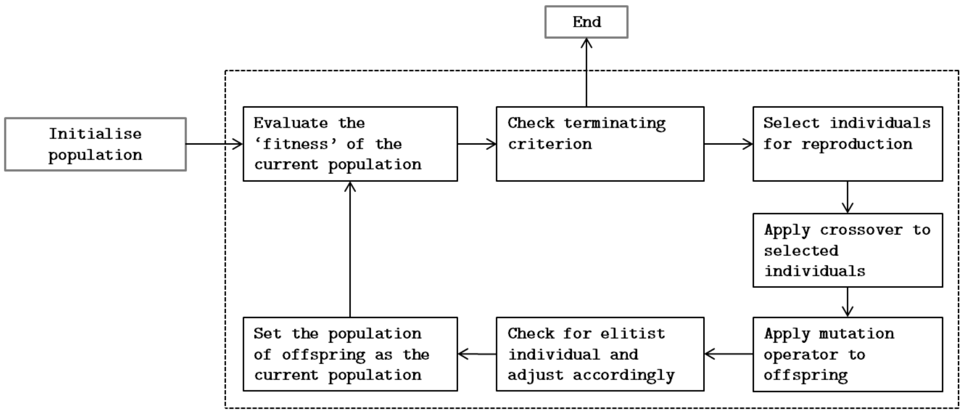
End
Initialise population
Evaluate the 'fitness' of the current population
Check terminating criterion
Select individuals for reproduction
Apply crossover to selected individuals
Apply mutation operator to offspring
Check for elitist individual and adjust accordingly
Set the population of offspring as the current population

Figure 4.5: The Cycle of a Generational Genetic Algorithm

The mutation operator is applied to the new population of candidate solutions once crossover has taken place. This operator is very important for maintaining diversity in the population or even introducing new 'genetic material'. We use the uniform mutation operator which flips a bit in the chromosome with a very small probability p_m. The cycle of an iteration of the SGA is given in Figure 4.5. The elitism operator ensures that if the best solution from the current population is not improved or not selected for reproduction, then it is persisted to the following population. A GA scheme that uses Elitism will never have an objective value that decreases over the course of the run if the objective function is deterministic.

When working with GA's, there are generally three different approaches to handling problem constraints which restrict the feasible search space. The first is to use a parametrization other than a binary encoding which will support the constraint validation of the problem through the data structure itself. This also needs to be extended to cover all GA operations concerned with crossover and mutation.

The second approach is to include penalty terms for any constraints that are broken which outweigh all other cost terms. The criticism of this method is that it increases the sensitivity of the GA and makes it difficult to move through the feasible space of solutions in an efficient way. The third approach is to have a validation step which takes a proposed solution and converts it to a feasible one before evaluation.

Both the first and third approach were tried when solving this particular problem. In the first approach, all resources and tasks were represented in the chromosome structure of the SGA. This meant that the genetic algorithm would have to specify which resources were being assigned to which tasks, as well as the order in which those tasks would be taking place.

Precedence constraints were required on the parameter sets to ensure that resources were not assigned in such a way as to produce a deadlocked schedule. Due to complexity in both the job sequence, resource allocation and unexpected interactions between parameters, the GA produced results with a schedule makespan around double that of the observed operational makespans. So while the schedule was feasible, the representation was too complicated for the GA to produce competitive results.

The second approach was not attempted due to the high number of constraints in the problem. Any infeasibility in the schedule would mean unusable results and given the complexity of a feasible and optimized resource allocation (demonstrated in the first approach) aligned with our intuition that this would not produce good results.

The approach used in the final implementation is in line with the third option above. Control of the permutation component of the problem (the order in which jobs are executed) is given to the GA, a second layer is used to handle the allocation of resources and validation of the schedule by working with the feasible set of possible options. The operators and encoding already discussed were used in this implementation of the SGA. Figure 4.6 gives an overview of the interaction between the model (schedule) and GA (abstract) layers.

Feasibility of the schedule is thus maintained by the schedule builder in the model layer rather than in the GA layer. At a high level, the validation step attempts to adhere to the job sequence provided by the GA and when an infeasible job is requested, the next job in the permutation is attempted. Once a feasible move has been found, the algorithm starts at the beginning of the remaining jobs list and will re-process previously skipped jobs in the

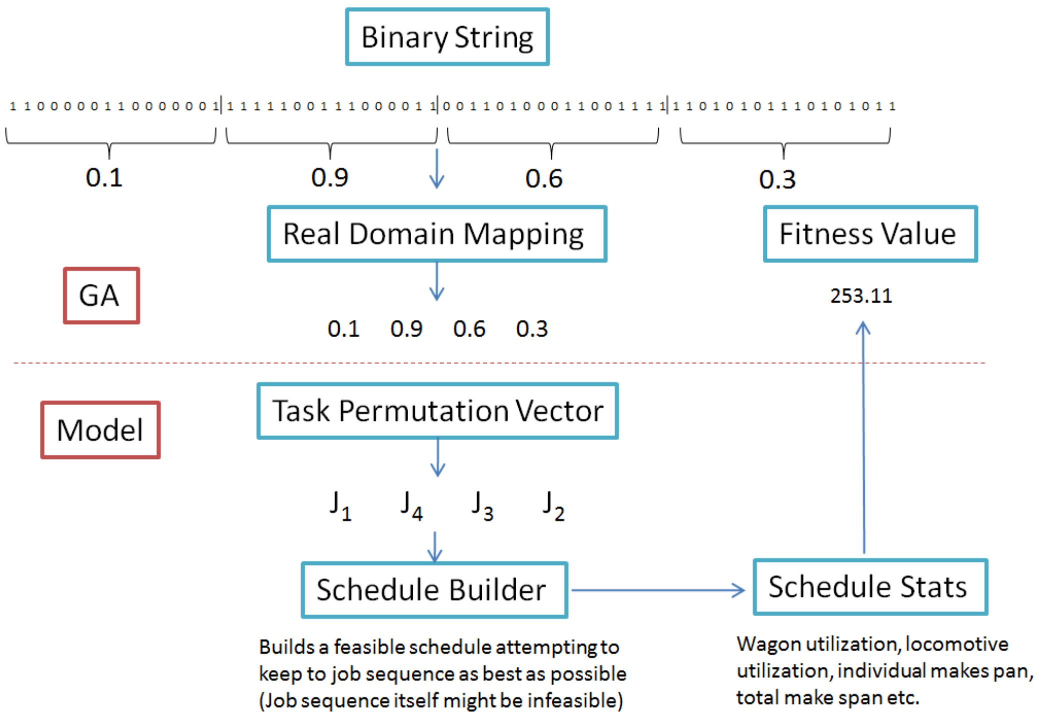

Figure 4.6: Model and Genetic Algorithm Interaction Diagram

event they are now feasible. The SGA used is outlined in Algorithm 1 and the validation step falls within the evaluation procedure called in lines 2 and 7.

This process allows the GA to deal with the priority of tasks at a high level without the complexity of resource allocation for each job. In order for the schedule builder to do this feasibly the schedule is built up incrementally, tracking the movements of all resources through their assigned tasks. Resources are allocated to jobs as part of the evaluation scheme and when no more assignments can be completed, the time horizon is moved forward potentially freeing up resources (from their current tasks). The evaluation procedure stops when the time line cannot be moved up (as all resources have processed their current tasks) and no more jobs can be assigned to resources, potentially due to no more remaining jobs.

Heuristic resource assignment is used in the validation algorithm. While multiple resource combinations may exist, the resource combination that satisfies all constraints for a particular job with the lowest dwell time is selected for the job. While this may lead to sub-optimal results in the resource allocation it does fall in line with the operational principle (first-in-first-out) that is used in practice.

```
initialise population pop;
Evaluate(newPop);
while $i \leq$ Generations do
    newPop $\leftarrow$ Select(pop);
    Crossover(newPop);
    Mutation(newPop);
    Evaluate(newPop);
    UpdateElite(pop,newPop);
    pop $\leftarrow$ newPop;
    $i = i + 1$;
end
```

Algorithm 1: Simple Genetic Algorithm Pseudo Code

A very successful heuristic scheme to restrict the validation step to only consider the top 90^{th} percentile of options, ranked by their waiting time before the job may begin, was deployed in the final GA. The idea behind this is that while certain jobs can be completed exactly according to the priority, we would want to avoid very costly decisions in the hope that they can be deferred to later on. This *repair* of the chromosome was able to drastically improve the quality of starting solutions (where there are many bad decisions) as well as avoid fewer bad decisions in the final solutions which would have taken several iterations to be ironed out.

The average improvement was approximately 8% and 2% in the initial population and final solutions respectively. Only a fifth of the original number of iterations were required to achieve better final solutions on average using this restricting repair operator. These improvements suggest that the GA was very likely to generate bad priorities in solutions throughout the run.

Using the default parameters recommended by Grefenstette [28] we present the results of the SGA run on the small test data set in Figure 4.7.

The SGA was able to out-perform the *LAH* in terms of the best solution found in an equivalent amount of computation time with the solution found being within 5% of the known optimal within 10 iterations.

The second problem set consists of 30 loads, which represents a significant step up on the small test set. Doing an approximate extrapolation

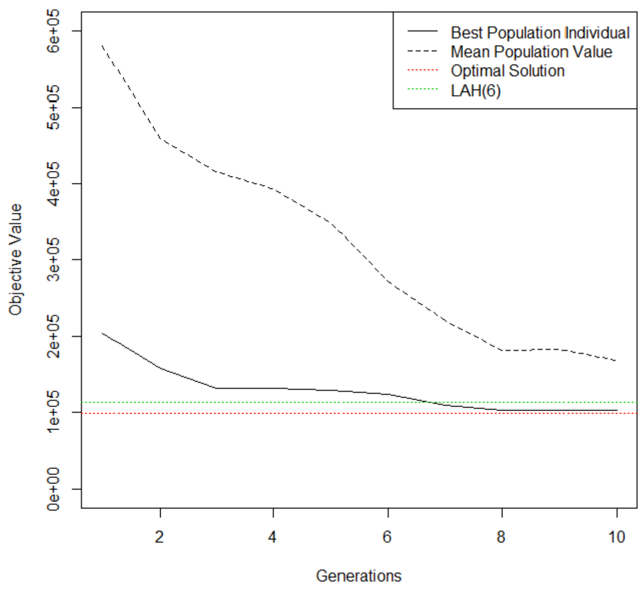

Figure 4.7: GA run results, small test data set

of the expected amount of time the depth first search would run for, given the parameters fitted in the previous model and that the complexity stays approximately the same with a larger mix of demand, the search will take around 1.7×10^{27} years. We will compare the best objective value obtained by the LAH for varied small values of a to the GA for these larger data sets.

The results of the run shown in Figure 4.8 illustrate the variability in the solutions obtained. The LAH performed around 40% better than the random solutions on average. However, there were some random starting solutions in the GA which were able to outperform the LAH, suggesting that it did not scale very well with the problem size. The GA improved the initial solution found by a further 9.2% before converging on a single solution. Due to the amount of search required by the LAH on this problem size, the GA easily outperformed it in terms of solution quality and time taken to generate the solution.

The third test set consists of 204 loads, in line with what we would expect when dealing with realistically sized problems where a schedule is required for a week. A single run of the GA is illustrated in Figure 4.9. The mean and best individuals in the GA population at each iteration are shown in the left diagram. The right diagram plots the average wagon makespan in the schedule, a metric closely related to minimising wagon waiting time. We

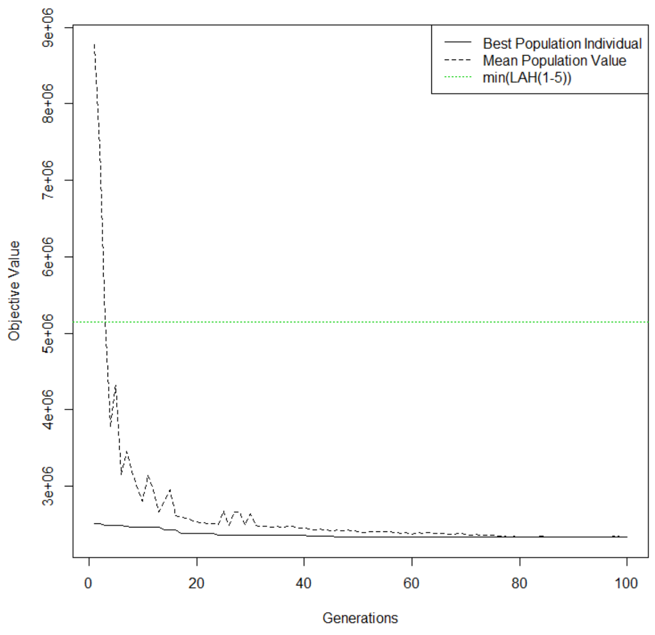

Figure 4.8: GA run results, medium test data set

show the values for each objective function evaluation in this right diagram to contrast it with the objective curve in the left diagram. We find that there is an improvement over the initial population of around 22%, which is substantially larger than that of the medium sized problem. This is not a surprising result considering the larger complexity of the problem.

While the objective function does not explicitly measure the average makespan of the resources it does measure the total waiting time in the schedule. By minimising this term the GA manages to reduce the average makespan for the whole fleet. It is interesting to note that the minimum average makespan in each generation is reduced from 6.8 to 6.3 from the initial to final population. In real terms these kind of improvements have massive cost implications running into the hundreds of millions of Rands in savings per annum when extrapolated.

The objective is heavily biased towards the total tonnage throughput (a large β value in the objective function). This bias becomes clear when looking at the average wagon makespan against the objective cost, shown in Figure 4.10. It is not surprising that the lowest makespan is associated with the lowest objective value since we would expect when converging that

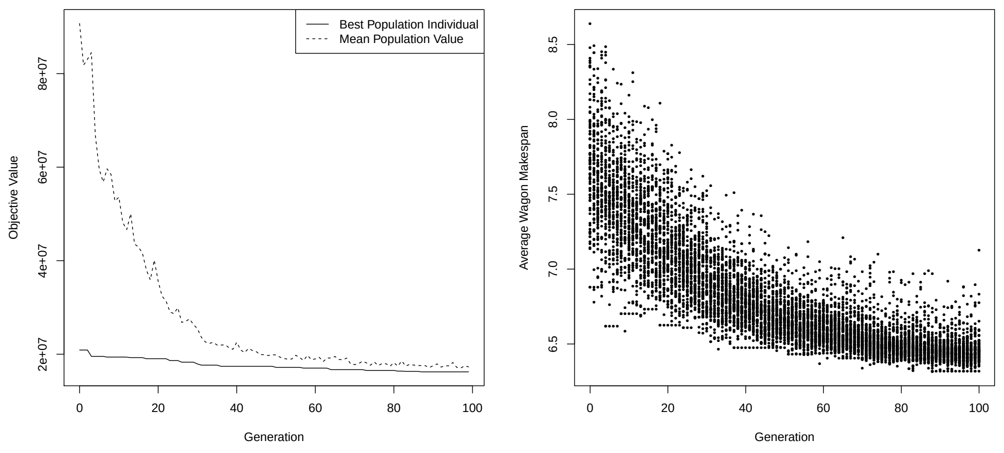

Figure 4.9: GA run results, large test data set

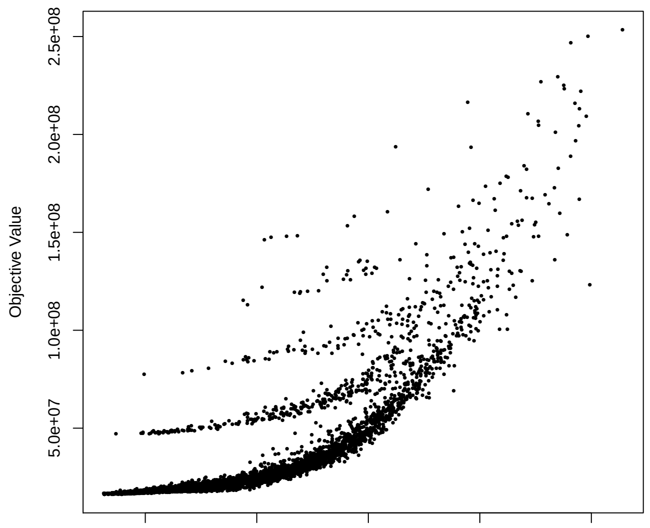

Figure 4.10: Large GA run, all objective function evaluations against average wagon makespan

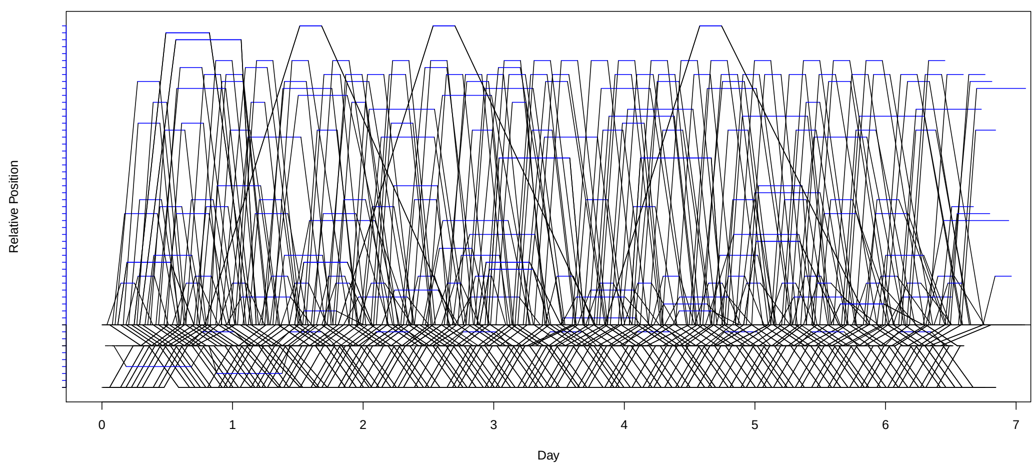

Figure 4.11: Large GA run, best solution train diagram

the greatest amount of search takes place within the neighborhood of the best individual. The distinct bands are the result of the bias towards high tonnages as they differ by a constant amount. One could argue that the bias towards the tonnage could be lowered in favor of a more reasonable tradeoff between tonnage and resource utilization. Often the primary KPI in railways is the throughput of the network with good resource utilization being a driver of the ultimate solution.

An illustration of the final output from the GA is shown as a train diagram in Figure 4.11. All arcs used in the schedule are shown as solid lines and the relative position of sites is given. Points at which wagons are being loaded are shown as blue horizonal lines. The angle of the lines between sites gives an indication of the travel time cost to service a site. Since the relative position is shown, the crossing of arcs should not be interpreted, as normally the case, as the times when trains are expected to pass one another.

Figure 4.13 illustrates that the GA runs are largely unaffected by the quality of the initial starting solutions. Poor solutions, high in objective value, undergo larger improvements from the initial population to the final solution. The improvement is taken as the percentage reduction from the best solution in the initial population to the best solution in the final population (at 100 generations). A significant regression model ($\alpha = 0.001$) was fitted to the data to illustrate the relationship. An average improvement of 14.5% can be expected from a typical GA run on the large data set at 100 generations.

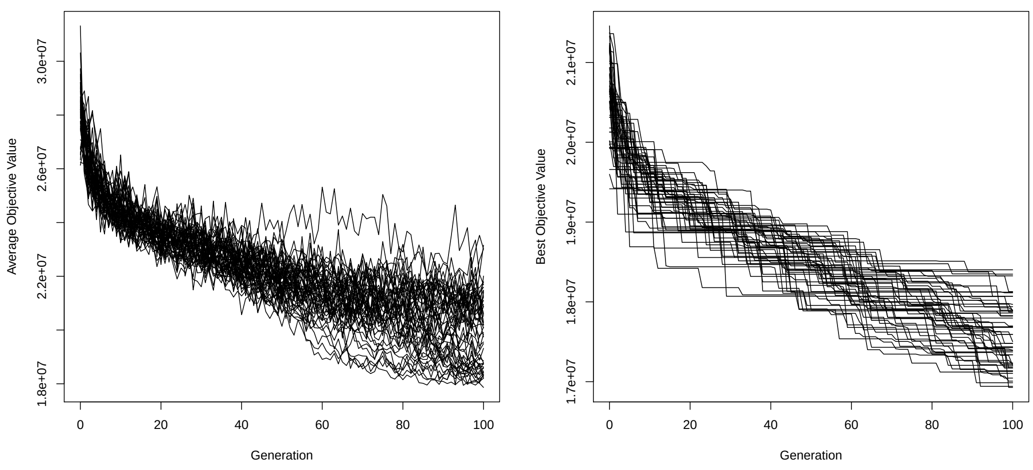

Figure 4.12: Multiple GA runs, average and best performance

The second diagram in Figure 4.13 shows that the starting objective values do not correlate with the final objective values. This suggests that the GA is not only robust in this setting but also that sufficient iterations have been performed to ensure reasonable independence of the final results.

Figure 4.14 affirms earlier comments that the bias term for the scheduled tonnage needs to be reduced as it is creating a significant local optimum which is repeatedly being converged to.

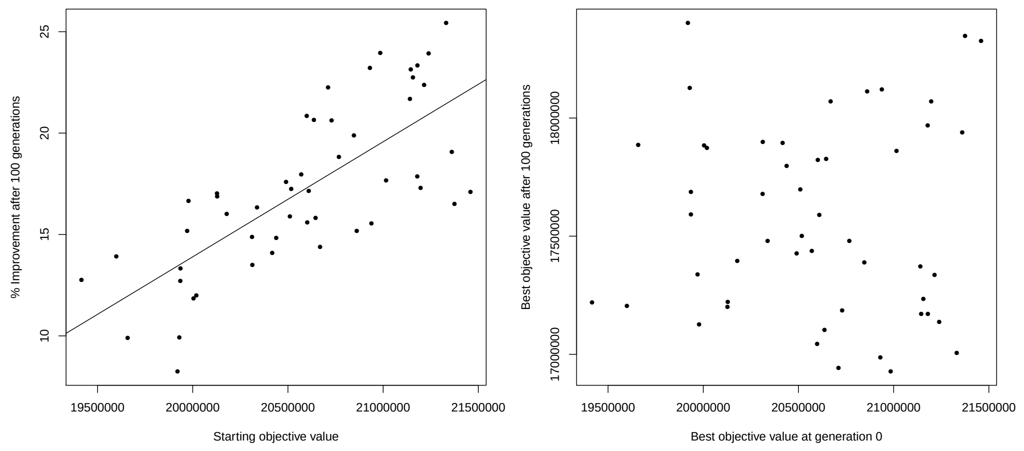

Figure 4.13: GA improvement against starting objective values

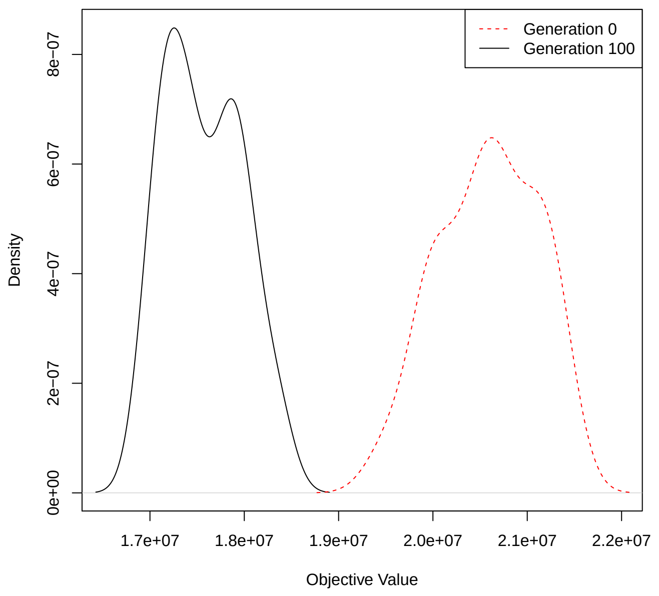

Figure 4.14: Solution distribution at Generation 0 and 100

4.4 Implementation Results

In 2009 Transnet Freight Rail (TFR) went to tender for a solution that could deal with the dynamic nature of the scheduling required for the Coal Link servicing the Richards Bay Coal Terminal (RBCT) in South Africa. OPSI Systems was awarded the tender in late 2009, citing their extensive knowledge in truck scheduling environment and its relationship to the required coal train scheduling as being a large factor in the adjudication process.

The solution was modeled as described in Chapter 3 and the genetic algorithm described in Section 4.3 bundled within a larger application called *PLATO.Rail*. PLATO.Rail is a windows application which provides not only an intuitive front end for the scheduler but also several diagnostics and interfaces to provide insights to the solution outputs. The application also extends to real time resource monitoring, execution and rescheduling which is out of the scope of this dissertation.

PLATO.Rail had a large impact on the total amount of time and effort taken to do the planning, or scheduling, for the coming week. Previous ad hoc systems which were in place would take several hours to create a schedule, sometimes running into the next day before a relaxed schedule was returned. The total run time of the schedule is under 10 minutes in PLATO.Rail which allows planners to tweak the scenario and adjust parameters until a satisfactory result is obtained.

The application went 'live' at TFR in 2011 after a year of prototyping and development. Figure 4.15 gives the three week moving average on tonnages for the Coal Line. The horizonal lines indicate the average weekly throughput for each financial period.

After three years of stagnant performance the Coal Line saw a significant increase in throughput which has been sustained through the year. Although some of the increase is attributed to an increase in resources, previous years also had resource injections but were not able to achieve an increase in annual volumes, suggesting the in order to unlock the potential of the fleet, better planning was required.

A joint entry between TFR and OPSI Systems for scheduling excellence won a gold *Logistics Achiever Award* in 2012 for the PLATO.Rail project and implementation on the Coal Line in South Africa.

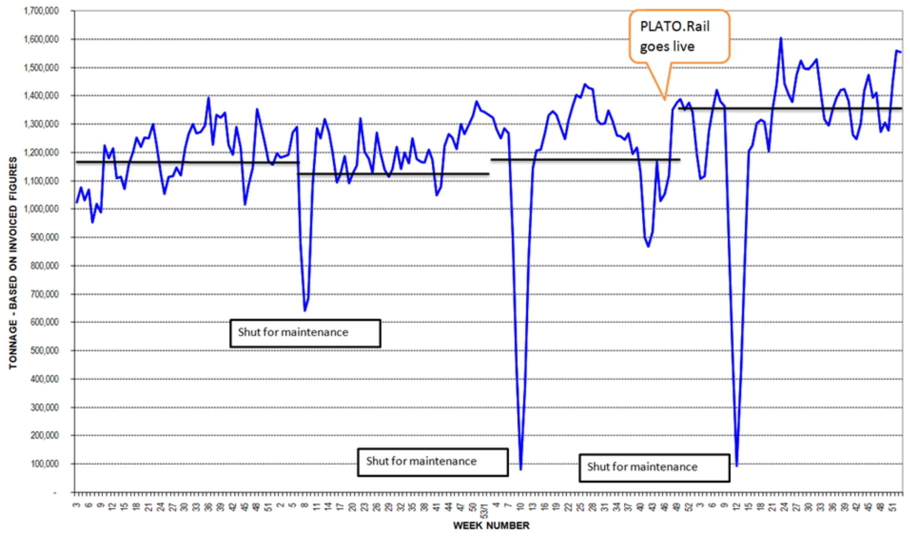

Figure 4.15: Coal Line tonnage: 3 week moving average from 2008/09 to 2011/12 financial year

Chapter 5

Conclusion

Many academic works in the train scheduling environment deal primarily with the optimization of resource movements through the physical network given an existing train schedule. The task of assigning resources to a train schedule is often handled as a separate optimization task. Bulk freight environments often operate without a fixed schedule due to the variability in demand from one period to the next.

An introduction to the concepts of bulk freight train scheduling in a South African context is given in Chapter 1. A review of the literature is given in Chapter 2 with specific attention being given to the Train Timetabling Problem (TTP) and Locomotive Assignment Problem (LAP).

The optimization difficulty in bulk freight train scheduling is to simultaneously determine the train timetable and resource allocation, a combination of TTP and LAP. This is done with respect to additional operational constraints such as reclaiming times, site maintenance, occupations and resource configuration options to name a few. A formulation of the problem is given in Chapter 3 which aims to maximise resource utilization and throughput while minimising the number of light locomotives movements required.

An exact search was written to solve the formulation presented in Chapter 3. The results from the exact search for small problems were used to benchmark a heuristic branch-and-bound. The same methodology was then applied to benchmark a Genetic Algorithm using the heuristic branch-and-bound which is capable of solving realistically sized problems. The final result was that the Genetic Algorithm is able to strike a good trade off between solution quality and computation time.

The Genetic Algorithm discussed in Chapter 4 is being successfully used by Transnet Freight Rail on the Richards Bay Export Line to complete their weekly planning. Implementation details are provided in Section 4.4.

Due to the success of the implementation of the scheduling algorithm employed on the Richards Bay Export Line; *OPSI Systems* was commissioned by Transnet Freight Rail in late 2012 to extend the algorithm to schedule all coal movements in South Africa. The long term view is to reduce loading conflicts at mines through a single, integrated and transparent plan.

Scheduling all domestic and export coal requires fundamental changes to the model presented in this dissertation to support multiple offloading sites, empty wagon distribution and more complex network related decisions. The number of decision variables will also be increased in this model by a factor of approximately 5.

Alternatives to the Genetic Algorithm could be explored for the extended model. One possibility is a multi-phase approach whereby Constraint Programming (CP) is initially used to generate a feasible allocation of trains to mines given the available slots followed by a mixed integer programming formulation (similar to that used by Ahuja *et al* [1]) to provide resource allocation (of both wagons and locomotives) given the fixed set of slots.

Chapter 6

Bibliography

[1] R K Ahuja, J Liu, J B Orlin, D Sharma, and L A Shughart. Solving Real-Life Locomotive-Scheduling Problems. *Transportation Science*, 39(4):503–517, November 2005.

[2] A Assad. Models for rail transportation. *Transportation Research*, 14A:205–220, 1979.

[3] A Assad. Modelling of rail networks: Toward a routing/makeup model. *Transportation Research Part B: Methodological*, 14(1-2):101–114, March 1980.

[4] E Balas, N Simonetti, and A Vazacopoulos. Job shop scheduling with setup times, deadlines and precedence constraints. *Journal of Scheduling*, 11(4):253–262, 2008.

[5] J F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 252(4):238–252, 1962.

[6] L D Bodin, B L Golden, A D Schuster, and W Romig. A model for the blocking of trains. *Transportation Research Part B*, 14B:115–120, 1980.

[7] J M P Booler. The Solution of a Railway Locomotive Scheduling Problem. *The Journal of the Operational Research Society*, 31(10):943, October 1980.

[8] J M P Booler. A Note on the Use of Lagrangean Relaxation in Railway Scheduling. *Journal of the Operational Research Society*, 46(1):123–127, 1995.

[9] R L Burdett and E Kozan. Techniques for absolute capacity determination in railways. *Transportation Research Part B: Methodological*, 40(8):616–632, September 2006.

[10] R L Burdett and E Kozan. A sequencing approach for creating new train timetables. *OR Spectrum*, 32(1):163–193, July 2008.

[11] R L Burdett and E Kozan. Techniques for inserting additional trains into existing timetables. *Transportation Research Part B*, 43(8-9):821–836, 2009.

[12] R L Burdett and E Kozan. Techniques for restricting multiple overtaking conflicts and performing compound moves when constructing new train schedules. *Mathematical and Computer Modelling*, 50(1):314–328, 2009.

[13] X Cai and C J Goh. A fast heuristic for the train scheduling problem. *Computers and Operations Research*, 21(5):499–510, 1994.

[14] X Cai and C J Goh. Greedy heuristics for rapid scheduling of trains on a single track. *IIE transactions*, 30(5):481–493, May 1998.

[15] A Caprara, M Monaci, P Toth, and P L Guida. A lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics*, 154(5):738–753, 2006.

[16] Alberto Caprara, Matteo Fischetti, and Paolo Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5):851–861, 2002.

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

[18] A Caumond, P Lacomme, and N Tchernev. A memetic algorithm for the job-shop with time-lags. *Computers and Operations Research*, 35(7):2331–2356, 2008.

[19] J F Courdeau. A survey of optimization models for train routing and scheduling. *Transportation Science*, 4(3):380–404, June 1998.

[20] G B Dantzig and P Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[21] L Davis. *Handbook of genetic algorithms.* VNR computer library. Van Nostrand Reinhold, 1991.

[22] Kenneth Alan De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, Ann Arbor, MI, USA, 1975. AAI7609381.

[23] M Florian, G Bushell, J Feland, G Guerin, and L Nastansky. The engine scheduling problem in a railway network. *INFOR*, 14(2):121–139, 1976.

[24] M A Forbes, J N Holt, and A M Watts. Exact Solution of Locomotive Scheduling Problems. *The Journal of the Operational Research Society*, 42(10):825– 831, 1991.

[25] M A Forbes, J N Holt, and A M Watts. An exact algorithm for multiple depot bus scheduling. *European Journal Of Operational Research*, 72:115–124, 1994.

[26] D E Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Artificial Intelligence. Addison-Wesley Pub. Co., 1989.

[27] M F Gorman. An application of genetic and tabu searches to the freight railroad operating plan problem. *Annals of Operations Research*, 78:51 – 69, 1998.

[28] J J Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *Transactions on Systems, Man and Cybernetics*, 16(1):122–128, 1986.

[29] N Hansen and A Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–95, January 2001.

[30] A Higgins, E Kozan, and L Ferreira. Optimal scheduling of trains on a single line track. *Transportation Research Part B: Methodological*, 30(2):147–161, 1996.

[31] A Higgins, E Kozan, and L Ferreira. Heuristic techniques for single line train scheduling. *Journal of Heuristics*, 3(1):43–62, 1997.

[32] A Higgins, E Kozan, and L Ferreira. Modelling the number and location of sidings on a single line railway. *Computers & Operations Research*, 24(3):209–220, March 1997.

[33] K L Huang and C J Liao. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers and Operations Research*, 35(4):1030–1046, 2008.

[34] RSK Kwan and P Mistry. A co-evolutionary algorithm for train timetabling. *Computation, 2003. CEC'03. The 2003*, (July), 2003.

[35] E Lawler and D Wood. Branch-and-Bound Methods: A Survey. *Operations Research*, 14(4):699–719, July 1966.

[36] Shi-qiang Liu and Erhan Kozan. A Blocking Parallel-Machine Job-Shop-Scheduling Model for the Train Scheduling Problem. *The 8th Asia-Pacific industrial engineering and management systems conference*, pages 10.1 – 10.10, 2007.

[37] Shi Qiang Liu and Erhan Kozan. Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers & Operations Research*, 36(10):2840–2852, October 2009.

[38] Shi Qiang Liu and Erhan Kozan. Optimising a coal rail network under capacity constraints. *Flexible Services and Manufacturing Journal*, 23(2):90–110, January 2011.

[39] D H Noble, M Al-Amin, and R G J Mills. Production of locomotive rosters for a multi-class multi-locomotive problem. *Journal of the Operational Research Society*, 52(11):1191–1200, November 2001.

[40] Z Othman, K Subari, and N Morad. Job Shop Scheduling with alternative machines using Genetic Algorithms. Technical Report D, Universiti Teknologi Malaysia, 2007.

[41] S Rouillon, G Desaulniers, and F Soumis. An extended branch-and-bound method for locomotive assignment. *Transportation Research Part B: Methodological*, 40(5):404–423, June 2006.

[42] Y Semet and M Schoenauer. An Efficient Memetic, Permutation-Based Evolutionary Algorithm for Real-World Train Timetabling. *2005 IEEE Congress on Evolutionary Computation*, 3:2752–2759.

[43] B Szpigel. Optimal train scheduling on single track railway. *OR*, pages 343–352, 1972.

[44] P Tormos, A Lova, F Barber, and L Ingolotti. A genetic algorithm for railway scheduling problems. *Studies In Computational Intelligence*, 128:255–276, 2008.

[45] B Vaidyanathan, R Ahuja, J Liu, and L Shughart. Real-life locomotive planning: New formulations and computational results. *Transportation Research Part B: Methodological*, 42(2):147–168, February 2008.

[46] W L Winston and J B Goldberg. *Operations research: applications and algorithms*. Thomson Brooks/Cole, 2004.

[47] M B Wright. Applying Stochastic Algorithms to a Locomotive Scheduling Problem. *The Journal of the Operational Research Society*, 40(2):187– 192, 1989.

[48] Koorush Ziarati, Franqois Soumis, Jacques Desrosiers, Sylvie Gelinas, and Andre Saintonge. Locomotive assignment with heterogeneous consists at CN North America. *European Journal of Operational Research*, 97:281–292, 1997.

Appendix A

Test Data

n	Class	l_n^{ecp}	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	$\sum$
1	DC	1		1	1			1			3
2	DC	1		1	1			1			3
3	DC	1		1	1			1			3
4	DC	1		1	1						2
5	DC	1		1	1			1			3
6	DC	1		1	1			1			3
7	DC	1		1	1						2
8	DC	1		1	1			1			3
9	AC	1			1		1			1	3
10	AC	1			1		1			1	3
11	AC	1			1		1			1	3
12	AC	1			1		1			1	3
13	AC	1			1		1			1	3
14	Diesel	0				1				1	2
15	AC	1			1		1			1	3
16	AC	1			1		1			1	3
17	AC	0			1		1			1	3
18	AC	0			1		1			1	3
19	AC	1			1		1			1	3
20	AC	1			1		1			1	3
21	AC	1			1		1			1	3
22	DC	1		1	1			1			3
23	AC/DC	1			1		1			1	3
24	AC/DC	1			1		1			1	3
25	DC	0		1	1			1			3
26	DC	1		1	1						2
27	AC	1			1		1			1	3
28	DC	1		1	1						2
29	AC/DC	1			1		1			1	3

30	AC/DC	1			1		1			1	3
31	AC/DC	1		1	1						2
32	AC/DC	1		1	1						2
33	AC/DC	1			1		1			1	3
34	AC/DC	1		1	1						2
35	AC/DC	1		1	1						2
36	AC	1			1		1			1	3
37	AC	1			1		1			1	3
38	DC	0		1	1			1			3
39	Diesel	1	1		1				1		3
40	Diesel	1	1		1				1		3
41	Diesel	1			1				1		2
42	AC/DC	1			1		1			1	3
43	AC	1			1		1			1	3
44	AC/DC	1		1	1						2
45	AC/DC	1			1		1			1	3
46	AC/DC	1		1	1						2
47	AC/DC	1		1	1						2
48	DC	1		1	1			1			3
49	DC	1		1	1						2
50	AC/DC	0			1		1			1	3

Table A.2: Locomotive Set Class, ECP and Region Constraints

From Site	To Site	Slot Interval		From Site	To Site	Slot Interval
Hub 1	Site 30	30		Hub 1	Site 22	30
Hub 1	Site 19	30		Hub 1	Site 20	30
Hub 1	Site 10	30		Hub 1	Site 31	30
Hub 1	Site 2	30		Hub 1	Site 24	30
Hub 1	Site 4	30		Hub 1	Site 23	30
Hub 1	Site 3	30		Hub 1	Site 15	30
Hub 1	Site 25	30		Hub 1	Site 17	30
Hub 1	Site 18	30		Hub 1	Site 8	30
Hub 1	Site 27	30		Hub 1	Site 33	30
Hub 1	Site 28	30		Hub 1	Site 5	30
Hub 1	Site 7	30		Hub 1	Site 6	30
Hub 1	Site 32	30		Hub 1	Site 12	30
Hub 1	Site 26	30		Hub 1	Site 38	30
Hub 1	Site 9	30		Hub 1	Site 34	60
Hub 1	Site 29	30		Hub 1	Hub 2	60
Hub 1	Site 21	30		Hub 2	Hub 1	60
Hub 1	Site 11	30		Hub 2	Site 34	60
Hub 1	Site 16	30		Hub 2	Site 27	30
Hub 1	Site 1	30		Hub 2	Site 36	30
Hub 1	Site 13	30		Hub 2	Terminal	60
Hub 1	Site 14	30		Terminal	Hub 2	60

Table A.1: Site slot intervals

m	Wagon Class	w_m^{size}	w_m^{ecp}		m	Wagon Class	w_m^{size}	w_m^{ecp}		m	Wagon Class	w_m^{size}	w_m^{ecp}
1	J	100	1		29	J	100	1		57	J	100	0
2	J	100	1		30	J	100	0		58	J	100	1
3	J	100	1		31	J	100	0		59	J	100	0
4	J	100	1		32	J	100	0		60	J	100	0
5	J	100	1		33	J	100	0		61	S	50	0
6	J	100	1		34	J	100	0		62	S	50	0
7	J	100	1		35	J	100	0		63	S	50	0
8	J	100	1		36	J	100	0		64	S	50	0
9	J	100	1		37	J	100	1		65	S	50	0
10	J	100	1		38	J	100	1		66	S	50	0
11	J	100	1		39	J	100	1		67	S	50	0
12	J	100	1		40	J	100	1		68	S	50	0
13	J	100	1		41	J	100	0		69	S	50	0
14	J	100	1		42	J	100	0		70	S	50	0
15	J	100	1		43	J	100	1		71	S	50	0
16	J	100	1		44	J	100	1		72	S	50	0
17	J	100	1		45	J	100	1		73	J	100	0
18	J	100	1		46	J	100	1		74	J	100	1
19	J	100	0		47	J	100	1		75	J	100	0
20	J	100	1		48	J	100	1		76	J	100	1
21	J	100	1		49	J	100	0		77	J	100	0
22	J	100	1		50	J	100	1		78	J	100	1
23	J	100	0		51	J	100	1		79	J	100	1
24	J	100	1		52	J	100	0		80	J	100	1
25	J	100	0		53	J	100	1		81	J	100	1
26	J	100	1		54	J	100	0		82	J	100	1
27	J	100	1		55	J	100	0		83	S	50	0
28	J	100	1		56	J	100	0		84	S	50	0

Table A.3: Wagon Set Class, Size and ECP properties

Empty						Loaded					
S_i	S_j	Configuration	S_i	S_j	Configuration	S_i	S_j	Configuration	S_i	S_j	Configuration
Hub 1	Site 1	100J	Hub 1	Site 21	100J	Site 1	Hub 1	100J	Site 21	Hub 1	100J
Hub 1	Site 2	100J	Hub 1	Site 22	100S	Site 2	Hub 1	100J	Site 22	Hub 1	100S
Hub 1	Site 3	100J	Hub 1	Site 23	100S	Site 3	Hub 1	100J	Site 23	Hub 1	100S
Hub 1	Site 4	100J	Hub 1	Site 23	100J	Site 4	Hub 1	100J	Site 23	Hub 1	100J
Hub 1	Site 5	100S	Hub 1	Site 24	100S	Site 5	Hub 1	100S	Site 24	Hub 1	100S
Hub 1	Site 5	100J	Hub 1	Site 24	100J	Site 5	Hub 1	100J	Site 24	Hub 1	100J
Hub 1	Site 6	100S	Hub 1	Site 25	100J	Site 6	Hub 1	100S	Site 25	Hub 1	100J
Hub 1	Site 6	100J	Hub 1	Site 26	100J	Site 6	Hub 1	100J	Site 26	Hub 1	100J
Hub 1	Site 7	100J	Hub 1	Site 27	100J	Site 7	Hub 1	100J	Site 27	Hub 1	100J
Hub 1	Site 8	100S	Hub 1	Site 28	100J	Site 8	Hub 1	100S	Site 28	Hub 1	100J
Hub 1	Site 8	100J	Hub 1	Site 29	100J	Site 8	Hub 1	100J	Site 29	Hub 1	100J
Hub 1	Site 9	100S	Hub 1	Site 29	100S	Site 9	Hub 1	100S	Site 29	Hub 1	100S
Hub 1	Site 10	100J	Hub 1	Site 30	100J	Site 10	Hub 1	100J	Site 30	Hub 1	100J
Hub 1	Site 11	100J	Hub 1	Site 30	100S	Site 11	Hub 1	100J	Site 30	Hub 1	100S
Hub 1	Site 12	100S	Hub 1	Site 31	100J	Site 12	Hub 1	100S	Site 31	Hub 1	100J
Hub 1	Site 12	100J	Hub 1	Site 32	100J	Site 12	Hub 1	100J	Site 32	Hub 1	100J
Hub 1	Site 13	100J	Hub 1	Site 33	100J	Site 13	Hub 1	100J	Site 33	Hub 1	100J
Hub 1	Site 14	100J	Hub 1	Site 37	100J	Site 14	Hub 1	100J	Site 37	Hub 1	100J
Hub 1	Site 15	100J	Hub 1	Site 38	100J	Site 15	Hub 1	100J	Site 38	Hub 1	100J
Hub 1	Site 16	100J	Hub 2	Hub 1	100J 100S	Site 16	Hub 1	100J	Hub 1	Hub 2	100J 100S
Hub 1	Site 17	100J	Hub 2	Hub 1	200J	Site 17	Hub 1	100J	Hub 1	Hub 2	200J
Hub 1	Site 18	100J	Hub 2	Site 34	100J	Site 18	Hub 1	100J	Hub 1	Hub 2	100J
Hub 1	Site 18	100S	Hub 2	Site 35	50S	Site 18	Hub 1	100S	Site 34	Hub 2	100J
Hub 1	Site 19	100S	Hub 2	Site 36	50S	Site 19	Hub 1	100S	Site 35	Hub 2	50S
Hub 1	Site 20	100J	Terminal	Hub 2	100J 100S	Site 20	Hub 1	100J	Site 36	Hub 2	50S
Terminal	Hub 2	200J				Hub 2	Terminal	100J 100S	Hub 2	Terminal	200J

Table A.4: Site pair allowable wagon configurations

S_i	S_j	T_{ij}		S_i	S_j	T_{ij}
Hub 1	Hub 2	345		Hub 2	Hub 1	300
Terminal	Hub 2	260		Hub 2	Terminal	332
Hub 1	Site 1	135		Site 1	Hub 1	223
Hub 1	Site 2	267		Site 2	Hub 1	345
Hub 1	Site 3	248		Site 3	Hub 1	275
Hub 1	Site 4	203		Site 4	Hub 1	281
Hub 1	Site 5	107		Site 5	Hub 1	215
Hub 1	Site 6	70		Site 6	Hub 1	302
Hub 1	Site 7	164		Site 7	Hub 1	203
Hub 1	Site 8	142		Site 8	Hub 1	223
Hub 1	Site 9	123		Site 9	Hub 1	123
Hub 1	Site 10	210		Site 10	Hub 1	291
Hub 1	Site 11	121		Site 11	Hub 1	188
Hub 1	Site 12	1047		Site 12	Hub 1	1471
Hub 1	Site 13	224		Site 13	Hub 1	231
Hub 1	Site 14	310		Site 14	Hub 1	169
Hub 1	Site 15	133		Site 15	Hub 1	155
Hub 1	Site 16	135		Site 16	Hub 1	182
Hub 1	Site 17	154		Site 17	Hub 1	202
Hub 1	Site 18	310		Site 18	Hub 1	462
Hub 1	Site 19	400		Site 19	Hub 1	165
Hub 1	Site 20	156		Site 20	Hub 1	185
Hub 1	Site 21	183		Site 21	Hub 1	173
Hub 1	Site 22	139		Site 22	Hub 1	353
Hub 1	Site 23	109		Site 23	Hub 1	181
Hub 1	Site 24	101		Site 24	Hub 1	230
Hub 1	Site 25	200		Site 25	Hub 1	283
Hub 1	Site 26	204		Site 26	Hub 1	267
Hub 1	Site 27	192		Site 27	Hub 1	265
Hub 1	Site 28	166		Site 28	Hub 1	226
Hub 1	Site 29	75		Site 29	Hub 1	120
Hub 1	Site 30	350		Site 30	Hub 1	460
Hub 1	Site 31	97		Site 31	Hub 1	117
Hub 1	Site 32	159		Site 32	Hub 1	217
Hub 1	Site 33	164		Site 33	Hub 1	234
Hub 2	Site 34	151		Site 34	Hub 2	151
Hub 2	Site 35	86		Site 35	Hub 2	86
Hub 2	Site 36	137		Site 36	Hub 2	137
Hub 1	Site 37	20		Site 37	Hub 1	37
Hub 1	Site 38	180		Site 38	Hub 1	210

Table A.5: Site-Site Travel Times

S_σ	Day	$SW^s_{\sigma,\alpha}$	$SW^e_{\sigma,\alpha}$
Site 33	Monday	02:40:00	23:30:00
Site 1	Wednesday	06:00:00	18:00:00
Site 7	Tuesday	06:00:00	16:00:00
Site 10	Monday	00:00:00	18:00:00
Site 12	Monday	00:00:00	23:59:00
Site 12	Tuesday	00:00:00	11:00:00
Site 12	Tuesday	18:00:00	23:59:00
Site 12	Wednesday	00:00:00	11:00:00
Site 12	Wednesday	18:00:00	23:59:00
Site 12	Thursday	00:00:00	23:59:00
Site 12	Friday	00:00:00	11:00:00
Site 12	Friday	18:00:00	23:59:00
Site 12	Saturday	00:00:00	23:59:00
Site 12	Sunday	00:00:00	11:00:00
Site 12	Sunday	18:00:00	23:59:00
Site 32	Wednesday	05:00:00	18:00:00
Site 3	Wednesday	06:00:00	18:00:00
Site 38	Wednesday	06:00:00	18:00:00
Site 17	Thursday	02:01:00	17:00:00

Table A.6: Site Closure Times

	Number of Loads		
S_σ	Small	Medium	Large
Site 1		1	7
Site 2	1		4
Site 3	1	2	11
Site 4	1	4	27
Site 5			2
Site 6	1		2
Site 7			4
Site 8		1	2
Site 9			2
Site 10		3	19
Site 11		1	10
Site 12		1	3
Site 13			3
Site 14		1	2
Site 15		1	13
Site 16		1	5
Site 17		3	17
Site 18			2
Site 19			1
Site 20		1	1
Site 21		1	1
Site 22		1	1
Site 23			1
Site 24			3
Site 25		1	6
Site 26			4
Site 27			3
Site 28			3
Site 29			3
Site 30			1
Site 31			5
Site 32		1	12
Site 33		2	7
Site 34		1	9
Site 35		1	1
Site 36		1	1
Site 37			1
Site 38		1	5
$\sum$	4	30	204

Table A.7: Demand breakdown for all test data sets