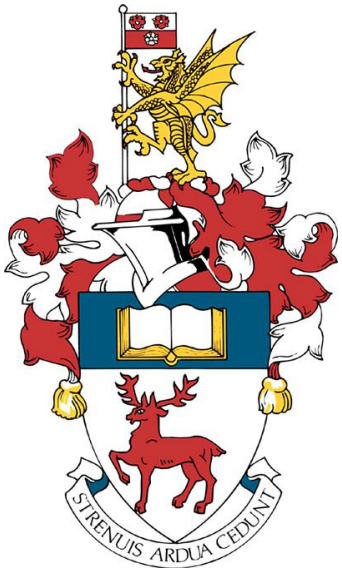
The University of Southampton

Faculty of Social Sciences

Mathematical Sciences



Vehicle Routing and Scheduling with Synchronisation, Time Windows and Skill Levels with Applications in Lifeboat Maintenance

Ruth Walton

A thesis submitted in partial fulfilment for the degree of Doctor of Philosophy

July 2021

University of Southampton

Faculty of Social Sciences

Mathematical Sciences

Abstract

Doctor of Philosophy

Vehicle Routing and Scheduling with Synchronisation, Time Windows and Skill Levels with Applications in Lifeboat Maintenance

by Ruth Walton

The Vehicle Routing Problem (VRP) is one of the most widely studied problems in Operational Research, due both to its complexity and vast number of possible variations and applications. In this thesis we present one such variation of the VRP with synchronisation, time windows and skill levels, designed for the application in the routing of lifeboat maintenance technicians in the Royal National Lifeboat Institution (RNLI), the largest coastal lifesaving charity in the UK. These technicians work in one of several geographical divisions and complete a mixture of planned work, known up to six months in advance, and unplanned work in response to equipment fault and failure which may need to be completed within as little as 48 hours. The objective is to produce routes of minimum cost for these technicians such that all jobs are completed within their respective time window by a technician (or pair of technicians) with the appropriate skill level, and where jobs are completed by two technicians their routes are synchronised to enable them to work together on a job with minimal time spent waiting. The contributing costs include travel, nights away for technicians working far from home and the costs incurred through lateness of jobs.

We formulate an integer program and propose a matheuristic algorithm, comprised of Lagrangian relaxation, brancn and bound and a local search heuristic, to solve the formulation at a divisional level across one or two weeks. We are able to prove that the

subgradient method used with Lagrangian relaxation achieves a good lower bound after just one iteration, which provides an initial solution on which to base the local search heuristic used to find further improvements. We show that this approach achieves usable solutions within a number of minutes, a time which is sufficient for such a method to be implemented by those responsible for technician planning at the RNLI. Such short computation times also allow for the inclusion of unplanned jobs at short notice, as the problem can be resolved with the remaining jobs in a planning period to produce new routes.

Contents

List of Figures

List of Tables

Statement of Authorship

I, Ruth Walton, declare that the thesis entitled "Vehicle Routing and Scheduling with Synchronisation, Time Windows and Skill Levels with Applications in Lifeboat Maintenance" and the work presented in it are my own and have been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

July 5, 2021

Acknowledgements

This thesis has been made possible due to the support of many people, probably too many to list here but I shall try nonetheless. All of these people believed in me and my ability to finish this work, even when I did not believe this myself.

First and foremost I would like to thank Oli, who has been the best and most supportive partner I could hope for, and has put up with countless evenings and weekends of thesis writing. I promise it is nearly over. Next to my family, in particular my parents, whose love and support throughout my life have doubtless lead me down the path on which I now find myself, for which I will always be grateful.

The journey of completing this PhD has also been made possible by many friends. Hattie Hall, the other half of my double act and the best flatmate I could have hoped for for many years, and Thomas Wilson for providing an ever dependable listening ear and voice of reason, and for his cracking sense of humour. I am also grateful for the wonderful office and departmental friendships which made being in the office fun, even on the bad days: Martina Testori, Simos Zachariades, Márton Benedek, Laura Murray, Karl Steinborn-Busse, Lily Clements, Naomi Andrew, Daniel Černín and so many others.

My academic journey would have been impossible without the input of so many Academics over the years. Most notable of course is the support of my Supervisors Stefano Coniglio and Jörg Fliege who have guided me to becoming the researcher I am today, and have always shown more belief in me than I could manage. I would also like to give particular thanks to Christine Currie for her support which helped more than she might realise.

I would like to thank all those at the RNLI who took time to assist me with the understanding of the problem required to complete this work. I also acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

Chapter 1

Introduction

The Vehicle Routing Problem (VRP) is one of the most widely researched area of Operational Research, with tens of thousands of new works on the VRP and its variation published each year[1]. The applications of this research are widely varied, and cover problems in the private, public and third sectors. In third sector applications such as charities, optimisation of processes is an increasingly important task as such organisations often have limited financial resources or rely on donations, so ensuring this is spent as efficiently as possible in all areas of the organisation's operations is key to maximising what can be achieved with the resources available. This can include Vehicle Routing type problems for organisations which operate large logistics networks, one such organisation being the Royal National Lifeboat Institution (RNLI) which will be the application considered in this work.

The RNLI is a British charity founded by Sir William Henry in 1824. Although it started as a small organisation, the RNLI is now the largest lifesaving charity active around the coast of the UK; it operates over 340 lifeboats across 242 lifeboat stations and since its inception has been responsible for saving more than 140,000 lives. They receive no funding from the UK government, so are entirely reliant on donations to keep these essential services active. They have also more recently expanded their operations to include a Flood Response service, and a Lifeguard service on over 200 beaches around

[1]Google Scholar results for 'Vehicle Routing': 2020 - 30,500; 2019 - 29,400; 2018 - 31,500; 2017 - 31,500

the UK during the summer months. The lifeboats currently in service range from small Inshore Lifeboats to much larger All-Weather Lifeboats, all of which along with life guarding and flood response equipment must be maintained by a network of in house System Technicians (ST). It is the routing and scheduling of these STs which is the focus of this work.

These technicians are divided up in to geographical divisions, and within their division Technicians travel to the required locations to carry out the work. Most of these jobs are known as planned or preventative maintenance, which are known up to six months in advance, but a proportion (30-40%) of work carried out is unplanned, or corrective maintenance. These represent repairs or breakdowns which must be incorporated in to the schedule of planned work, sometimes with only 24-48 hours notice. Given a set of technicians and a set of jobs, the objective is to determine an optimal route and schedule for these STs so that all jobs are completed within the given planning horizon, including the corrective maintenance which may become known with little notice before its deadline.

There are many additional elements to this problem which would ideally be included, and part of the problem at hand is to establish how these can be implemented in a mathematical formulation, and which can be included in the problem whilst maintaining feasibility, and the ability to achieve a solution in a reasonable time. One key element is the ability for emergent work to be able to be incorporated in to existing schedules, ensuring it is completed within its specified time frame. In addition to this, all jobs (both corrective and preventative) have time windows for completion, and some are also subject to precedence constraints. Technicians are skilled in different areas (and sometimes at different levels), and a technician must have the appropriate skills in order to complete a job. In the case where no technician is trained in all the required skills to complete a job, then two technicians are required for this job, and the respective routes of these technicians must be synchronised. Some jobs do not require two technicians to be completed but are still able to be completed by two technicians simultaneously; in such cases it is assumed that the completion times for such jobs will be halved when there are two technicians working on the job.

The solution method presented in this thesis is a matheuristic, combining Lagrangian relaxation and branch and bound with a local search heuristic to achieve a solution. The formulation solved exactly in the first phase is a Lagrangian relaxation of the full formulation, and as such any solutions are not feasible for the full formulation. It does however provide a starting solution for the local search procedure, the first aim of which is to make the solution feasible for the original problem and then implements local searches over a number of iterations to improve the solution.

The possible benefits to the RNLI from Operational Research are not limited to the work described in this thesis. At present the supplies needed for this maintenance work are distributed all over the UK from a single warehouse on the south coast by a fleet of in house delivery trucks; in a related piece of work Coniglio et al. [2017] identify the possibility for savings to be made for the RNLI with the introduction of an additional warehouse. Although such a warehouse would incur running costs, it was found that the potential savings of reducing the distance travelled by delivery trucks would be sufficient to offset these additional costs. There are numerous other ways in which Operation Research could benefit the RNLI, as with many other third sector organisations, but they will not be presented in this work.

This thesis will be structured as follows: Chapter 2 will provide a literature review of the existing work relevant to this problem from the areas of vehicle routing and scheduling, as well as the solution methods used for these. Chapter 3 provides a in depth description of the problem within the context of the application being considered. This is followed by a full MILP formulation of the problem in chapter 4, and then chapter 5 gives a detailed explanation of the approaches used, both exact and heuristic which are ultimately combined to form a bespoke matheuristic for this problem. Chapter 6 will introduce that data relating to this problem that is known, as well as the procedures used to fabricate test data used in all the experimentation carried out, the results of which are presented in chapter 7. Finally chapter 8 will summarise the work presented in this thesis, and make any recommendation for further research surrounding this problem.

Chapter 2

Literature Review

2.1 Vehicle Routing Problem

There has been a lot of research surrounding the vehicle routing and problem, in part due to the large number of possible applications, but also due to the complexity of the problem. This literature review will provide a summary and comparison of those which are relevant to the problem of System Technician scheduling within the RNLI. The following notation will be used throughout this literature review, unless specified otherwise: N indicates the set of all nodes of the network, where $N = J \cup \{0\}$, 0 is the depot, and J the set of jobs to be completed; A_N is the set of all arcs (i, j) in the network, for $i, j \in N$, and A_J is the set of all arcs between jobs, $\{(i, j)|i, j \in J\}$.

The Vehicle Routing Problem (VRP) itself was first outlined in Dantzig and Ramser [1959], (although referred to as the Truck Dispatching Problem in this case), and is defined as a generalisation of the Travelling Salesman Problem (TSP). With a certain delivery requirement $q_i \in \mathbb{R}_+$ at each point $i \in J$ in the network, if Q, the capacity of the delivery vehicle, is greater than the total item demand (i.e. if $Q \geq \sum_{i \in J} q_i$), then the problem at hand is simply the TSP, where a single route is to be determined. Therefore, generalising to the case with the set K of vehicles of equal capacity Q, where Q and K satisfy the inequalities

$$Q > \max_{i \in J}\{q_i\} \tag{2.1}$$

$$Q < \sum_{i \in J} q_i \tag{2.2}$$

$$Q|K| \geq \sum_{i \in J} q_i \tag{2.3}$$

we have the VRP, in which the aim is to determine the shortest route for each vehicle in K so that demand at all points $i \in J$ is satisfied. Since this, numerous different extensions to the VRP have been considered, at varying length, in an attempt to create a formulation which better captures the complexities of such real life problems. An extensive review of the early work in the Vehicle Routing and Scheduling Problems can be seen in Bodin et al. [1983], and Vidal et al. [2019] provides a comprehensive review of the latest developments in extensions to the VRP family of problems.

2.1.1 Vehicle Flow Formulation

Of the work surrounding the VRP, most approaches fall in to one of two main areas: vehicle flow formulations, which will be discussed in this section, and set partitioning formulations, which will be discussed further in 2.1.2.

The vehicle flow family of formulations make use of two-index binary variables x_{ij} to indicate whether the arc (i, j) is part of the solution (i.e. node j is visited immediately after node i). The full formulation of the problem, as presented by Toth and Vigo [2002], is as follows:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{2.4a}$$

$$\text{s.t.} \sum_{i \in J, i \neq j} x_{ij} = 1 \qquad \forall j \in J \tag{2.4b}$$

$$\sum_{j \in J, i \neq j} x_{ij} = 1 \qquad \forall i \in J \tag{2.4c}$$

$$\sum_{j \in J} x_{0j} \leq |K| \tag{2.4d}$$

$$\sum_{j \in J} x_{0j} = \sum_{i \in J} x_{i0} \tag{2.4e}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - r(S) \qquad \forall S \subset J, |S| \geq 2 \tag{2.4f}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A_N \tag{2.4g}$$

where c_{ij} is the length of arc (i, j), $|K|$ is the number of available vehicles, and $r(S)$ is the minimum number of vehicles required to serve the subset of nodes S. (2.4b) and (2.4c) ensure that each job has exactly one resource arriving to and departing from it, (2.4d) and (2.4e) ensure that the same number of routes depart from and arrive at the depot, which cannot exceed $|K|$, the number of available resources, and (2.4f) are the subtour elimination or capacity constraints. Grötschel and Padberg [1975] prove this form of subtour elimination constraint to be facets of the polytope defined by (2.4b) and (2.4c), meaning it provides a strong lower bound to the linear relaxation, but this comes at a cost: the number of constraints in this formulation is $\mathcal{O}(2^n)$, which results in a formulation that is very computationally demanding. An alternative formulation of subtour elimination constraints is proposed in Miller et al. [1960]: these make use of a new set of continuous variables $u_i \in \mathbb{R}_+$ which indicate the accumulated demand already distributed by the vehicle when it arrives at node $i \in J$.

$$u_j \geq u_i + q_j - Q(1 - x_{ij}) \qquad \forall (i, j) \in A_J \tag{2.5}$$

$$q_i \leq u_i \leq Q \qquad \forall i \in J \tag{2.6}$$

Replacing (2.4f) with (2.5) and (2.6) gives a new formulation of the VRP. The advantage of this, known as the MTZ formulation, is that it has $\mathcal{O}(n^2)$ variables and constraints, but its linear relaxation generally produces a weak lower bound on the integer solution, as described in Desrochers and Laporte [1991]. Bektaş and Gouveia [2014] also present a generalisation of the MTZ constraints (and their alternatives proposed by Desrochers

and Laporte) which they demonstrate offer marginal computational improvements.

2.1.2 Set Partitioning Formulation

The other main type of formulation that has been applied to the VRP is the set partitioning model, first proposed by Balinski and Quandt [1964]. Considering Ω, the set of all possible routes in the network, the aim is to establish which routes to include in the final solution so that each job is visited exactly once (thus forming a two-subset partition of the original network). This is achieved with binary variables λ_r, for $r \in \Omega$ which take value 1 if route $r \in \Omega$ is included in the solution, and 0 otherwise. The cost of each route must be predetermined, and is represented by c_r, and the coefficient a_{ir} is equal to 1 if job i is visited on route r, and 0 otherwise. Thus, the set partitioning VRP formulation can be written as follows:

$$\min \sum_{r \in \Omega} c_r \lambda_r \tag{2.7}$$

$$\text{s.t.} \sum_{r \in \Omega} a_{ir} \lambda_r = 1 \qquad \forall i \in J \tag{2.8}$$

$$\sum_{r \in \Omega} \lambda_r \leq |K| \tag{2.9}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in \Omega \tag{2.10}$$

In this formulation, (2.8) requires that each job $i \in J$ is included in exactly one route in the solution, and (2.9) limits the number of routes to $|K|$, where K is the set of available vehicles. One obvious advantage of this formulation is the fact that it implicitly deals with any complex objective functions or route feasibility constraints; any complexities within these are addressed in the definition of c and Ω, without affecting the linearity of the final formulation, or requiring the use of additional constraints. In addition, Bramel and Simchi-Levi [1997] note that set covering formulations of the VRP provide a strong lower bound on the linear relaxation, but as with the original vehicle flow formulation in equation (2.4), this comes at a cost in terms of the number of variables

in the problem. Even with additional constraints surrounding the feasibility of routes, the number of variables in this formulation grows exponentially with $|N|$, making the problem prohibitively large.

In attempt to tackle this, some methods exist to reduce the number of variables in the problem. Of the routes in the set Ω, Balinski and Quandt define the route d to be dominated if there exists a subset of routes $S \subset \Omega$ for which the following constraints are satisfied:

$$\sum_{r \in S} a_{ir} = a_{id} \qquad \forall i \in J \tag{2.11}$$

$$\sum_{r \in S} c_r \leq c_d \tag{2.12}$$

Here, (2.11) states that the routes in the subset S must include exactly those jobs which are in route d, and (2.12) states that the combined cost of the routes in subset S is less than the cost of d. When such routes exist, they clearly need not be considered in the formulation, and as such these can be removed to reduce the dimension of the problem. However, the presence of dominated routes depends on the definition of c and Ω, with no guarantee as to the size of the reduction that it will allow, so such methods should not be relied upon to reduce the above formulation to a more computationally feasible size. To tackle the infeasibly large set of routes Ω, many authors use a column generation technique to systematically add new routes to the set Ω throughout the solution of the MILP. These solution methods will be discussed further in section 2.2.1.

Both the vehicle flow and set partitioning formulations are used widely in the VRP literature; the choice of formulation does affect the solution methods which can be used, as as such most authors present only one formulation. Table 2.1 captures the formulation used by each of the authors cited in this section (if a formulation is presented). From this point, all formulations presented will be in the vehicle flow format.

2.1.3 Vehicle Routing Problem with Time Windows

One of the earliest widely studied extensions of the VRP is the Vehicle Routing Problem with Time Windows (VRPTW), sometimes referred to as the Time Constrained VRP (TCVRP). This considers the standard VRP described above, with the added constraint that each job must be visited within a predetermined time window. Some of the earliest work on integrating time windows in to the routing problem can be seen in Baker [1983], although they consider a single vehicle variation of the problem, which can be seen as the Time Constrained TSP. Solomon [1987] and Kolen et al. [1987] go on to generalise this to the multi-vehicle VRPTW, which forms the basis of much subsequent work in this area.

Given a set of jobs J, each job $i \in J$ has an associated time window $[a_i, b_i]$ within which it must be completed, and an expected duration p_i. Given a maximum route duration T, the intuitive constraints on a_i and b_i are $a_i + p_i \leq b_i \leq T$ for all $i \in J$. Defining a new set of continuous variables, t_i, which represent the start time of job i, we have the following time window constraints, based on those seen in Xu and Chiu [2001]:

$$a_i \leq t_i \leq b_i - q_i \qquad \forall i \in J \tag{2.13}$$

$$t_i + p_i + c_{ij} \leq t_j + M(1 - x_{ij}) \qquad \forall (i,j) \in A_J \tag{2.14}$$

$$c_{0j} \leq t_j + M(1 - x_{0j}) \tag{2.15}$$

$$t_i + p_i + c_{i0} \leq T + M(1 - x_{i0}) \tag{2.16}$$

with M arbitrarily large. (2.13) requires that job i is started and finished within the specified time window, and (2.14)-(2.16) ensure the correct time sequence of jobs, as well as travel to and from the depot node. The consideration of time in the above constraints removes the need for subtour elimination constraints, as it does not allow individual jobs to be revisited. As such, (2.13)-(2.16) can be used in conjunction with the vehicle flow formulation described in equation (2.4), replacing the subtour eliminiation constraints (2.4f) to provide a full formulation of the VRPTW. Such formulations of the VRPTW

have been used by Xu and Chiu [2001], Bredström and Rönnqvist [2008], Azi et al. [2010] and Kovacs et al. [2012], although time windows are not the only additional constraints considered in these cases.

Alternatively, it is possible to formulate the VRPTW as a set partition model. As highlighted previously, in such formulations additional constraints on route feasibility do not affect the final MILP, but instead the definition of Ω, the set of all possible routes. A number of different approaches to this have been used: Bramel and Simchi-Levi [1997] use a column generation technique, in combination with a discretisation of customer locations and parameters, in order to categorise customers in to one of a fixed number of groups. They note that to reduce the computation time of the column generation step, they allow subtours within routes, although this affects the tightness of the lower bound. Due to the size of their problem instance, Eveborn et al. [2006] use total enumeration of the possible routes, but they allow infeasible routes to reduce the number of constraints to consider when defining Ω. Unlike Bramel and Simchi-Levi, the routes they consider must satisfy the basic VRP constraints, but they are not required to satisfy the additional constraints imposed by the authors; instead, penalties are used in the calculation of c_r to penalise those routes which violate the constraints, with varying severity depending on the constraint(s) in question.

Further work including time windows sees the inclusion of a number of other additional extensions to the original problem, all of which deal with the multi-vehicle approach. Most notably, we see a number of pieces of work which combine time windows with the notion of heterogeneous skill levels of resources (the full definition and discussion of which can be found in 2.1.4). These works include: Begur et al. [1997] develop a decision support tool for the scheduling and routing of nurses, which also considers constraints on the allowable time between return visits to customers (although no MILP formulation is presented); Weigel and Cao [1999] develop a tool which also allows for multiple routes, and pre-specified precedence constraints between jobs, as well as resource overtime (again, no formulation is given); Xu and Chiu [2001] include job priority, using a binary definition of skill level as part of the job weighting to discourage the assignment of jobs to technicians without the required skills; Eveborn et al. [2006] consider

the problem with team building and synchronisation (discussed further in 2.1.5), which occurs when jobs require more than one resource to be completed; Kovacs et al. [2012] also include team building in their formulation, as well as the possibility of outsourcing jobs to external resources; Pillac et al. [2013] consider the multi-skill VRPTW with tools and spare part requirements at each job. Although similar to the notion of skill requirements, considering the tools and spare parts is a generalisation of this as the availability of such items fluctuates over time, subject to usage and restocking. As such, skill requirements of jobs can be seen as a special case of this in which availability is constant.

Of those works which consider the VRPTW with homogeneous skill levels across resources, we also observe a wide range of additional constraints: Bredström and Rönnqvist [2008] include precedence relationships of jobs, and synchronisation of resources in cases where jobs require multiple resources; Bostel et al. [2008] develop a multi-period formulation, which produces a solution over multiple planning horizons, with the consideration of unplanned jobs which may arrive after scheduling has been completed; in Azi et al. [2010] and Azi et al. [2014] Azi et al. allow for multiple routes, also known as multiple use of vehicles (the VRP with multiple routes is knows as VRPM), in which each resource is not restricted to performing only one route per day. Zhang et al. [2019] present the VRPTW under uncetainty, where a *riskiness index*, a function of the probability of infeasibility and the magnitude of constraint violation, is used as a criteria for decision making.

Shelbourne et al. [2017] present the VRP with Release and Due Dates (VRPRDD), which has a very similar structure to the VRPTW, the key difference being that a vehicle cannot leave the depot until all the release dates for its contents have passed; this is more relevant in cases where the VRP is considered for the purposes of delivering goods rather than providing services. In another piece of work most relevant to delivery of goods, Reil et al. [2018] consider several variations of the VRPTW with backhauls, with varying constraints on linehaul and backhaul order, and whether customers solely require linehaul or backhaul service, or both simultaneously. Bulhões et al. [2018] present a relaxation of the VRPTW in the form of the VRP with Service Levels (VRP-SL).

In this case, rather than hard time windows which must be satisfied for each job, the constraint is a service level agreement which stipulates a minimum number or proportion of on-time deliveries across all service requests.

2.1.4 Vehicle Routing Problem with Skill Levels

One of the most commonly considered extensions is the inclusion of heterogeneous resources in the VRP, often referred to as resources with varying skill levels. The extent to which they are incorporated in to the formulation varies between authors, but the basic premise is that each resource has a prespecified set of skills, and each job has a required skill level that a resource must meet in order to be assigned to the job. The vehicle flow formulation of the VRP does not explicitly assign resources to routes; the homogeneity of resources means these can be assigned post optimisation, as they have no impact on the objective function value. As such, adaptations of this formulation are required to successfully include the varying skill levels of technicians. The explicit inclusion of skill levels in the VRP formulation is first seen in Cappanera et al. [2011]; many others previous to this talk about the need to consider such skill differences, but Cappanera et al. are the first to implement it. Moving away from the VRP, work has been done to incorporate varying skill levels in to the job scheduling problem, most notably in Cordeau et al. [2010] and Fırat and Hurkens [2012], and there is potential for overlap between these two areas.

Considering the previously defined set of vehicles K, we extend the set of binary variables used in the VF formulation to x_{ijk}, which indicates that arc (i, j) is traversed by technician $k \in K$. In addition, we define $s_k \in \mathbb{R}_+$ to be the skill level of technician k, and $r_i \in \mathbb{R}_+$ the skill level requirement of job i (we note that skill levels are conventionally considered as a discrete set in $\mathbb{Z}_+$, but this is not a requirement of the formulation). For ease of comprehension, we define the set $K_{ij} = \{k | s_k \geq \max\{r_i, r_j\}\}$, the set of technicians which meet the skill requirements of both jobs i and j. In the Skill VRP presented in Cappanera et al. [2011], they use a two-index flow variable y_{ij} to represent the amount of demand remaining on the route after j (where each job is assumed to have demand 1). They also consider uncapacitated vehicles, so generalising

their formulation to a case with demand q_j at job j and vehicle capacity Q, we reach the following formulation:

$$\min \sum_{(i,j)\in A} \sum_{k\in K_{ij}} c_{ijk} x_{ijk} \tag{2.17a}$$

$$\text{s.t.} \sum_{i\in J, i\neq j} \sum_{k\in K_{ij}} x_{ijk} = 1 \qquad \forall j \in J \tag{2.17b}$$

$$\sum_{j\in J, i\neq j} \sum_{k\in K_{ij}} x_{ijk} = 1 \qquad \forall i \in J \tag{2.17c}$$

$$\sum_{j\in J} y_{0j} = \sum_{j\in J} q_j \tag{2.17d}$$

$$\sum_{i\in J} y_{ij} - \sum_{h\in J} y_{jh} = q_j \qquad \forall j \in J \tag{2.17e}$$

$$y_{ij} \leq Q \sum_{k\in K_{ij}} x_{ijk} \qquad \forall (i,j) \in A_N \tag{2.17f}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A_N, k \in K_{ij} \tag{2.17g}$$

$$y_{ij} \geq 0 \qquad \forall (i,j) \in A_N \tag{2.17h}$$

As with the previous VF formulation, (2.17b)-(2.17c) ensure that exactly one technician arrives at and departs from each job. (2.17d) requires that the flow leaving the depot is exactly equal to the total demand in the network (different from the first VF formulation due to the alternative definition of the flow variables), and (2.17e)-(2.17f) are the subtour elimination constraints.

In their formulation, Cappanera et al. account for varying skill levels among resources, but they are still restricted to a single skill domain. The ability to consider multiple skill levels within a set of skill domains has been included in job scheduling problems (see Cordeau et al. [2010] and Fırat and Hurkens [2012]), and although this is not directly linked to the VRP, there are similarities which could be carried across. Cordeau et al. define a skill matrix for each technician, and a requirement matrix for each job, which take the following form:

$$s_k^{ld} = \begin{cases} 1 & \text{if technician } k \text{ is trained to level } l \text{ in skill domain } d \\ 0 & \text{otherwise} \end{cases}$$

r_i^{ld} = the number of technicians required by job i at level l in skill domain s

These skill levels are cumulative, so $s_k^{ld} \leq s_k^{l'd}$ and $r_i^{ld} \leq r_i^{l'd}$ for all $l < l'$, meaning a technician is trained at all levels below its maximum in each domain, and similarly for job requirements. An example of such matrices for a problem with 4 skill domains and 3 skill levels can be seen below:

$$s_k = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad r_i = \begin{bmatrix} 2 & 0 & 1 & 1 \\ 2 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Here we can see that technician k is trained to level 1 in skill area 1, level 3 in area 3 and level 2 in area 4. Job i requires two technicians in area 1, one of whom is trained to level 2 and one to level 3, one technician at level 1 in area 3, and one at level 3 in area 4. Requirements across different areas may be fulfilled by one single technician, provided they reach the required level in each area. If no such technician exists, then multiple may be sent to the job, but this is only required explicitly in cases like area 1 shown above.

These requirements for multiple technicians mean consideration must be given to how the schedules of multiple resources can be matched to allow for parallel completion. This can be solved using synchronisation, where technicians' routes coincide at a particular point to allow for join job completion, or team building, where technicians are sorted in to teams and travel together for the duration of the planning period. Further discussion around the implementation of these in the VRP can be seen in section 2.1.5.

The approaches to skill levels described up to this point represent hard constraints, meaning an under-qualified technician being assigned to a job represents infeasibility. Xu and Chiu [2001] instead use soft constraints on skill levels, in which assignments

of under-qualified technicians have a zero weighting factor in the objective function, meaning they are undesirable in a maximisation problem, but still represent a feasible solution. Although Xu and Chiu use binary skill levels across multiple areas (a technician is either trained or not in a particular area), this could easily be generalised to a problem with multiple skill levels.

Although presented differently, another similar problem to this is the Truck and Trailer Routing problem (TTRP), initially presented by Chao [2002]. This is a special case of the VRP in which the available resources consist of a number of trucks t and trailers r (where $r \leq t$), and customers can either be served by a complete vehicle (a truck pulling a trailer) or just a truck. Three types of routes are considered: those completed in full by just a truck, those completed in full by a complete vehicle, or mixed routes consisting of a main tour by a complete vehicle, with subtours completed by just a truck along the way. We refer to the first two cases as *pure* routes, and the latter as a *mixed* route. Although traditionally considered for applications in which the physical access to jobs limits the ability of a complete vehicle to serve the respective customers, formulations such as these could also be adapted for the VRP with skill levels. In the case of multiple skill domains with no overlap in the skill set of resources (i.e. no resource is able to serve multiple types of jobs), the remaining problem is a special case of the TTRP described above with only pure routes, as there are no resources capable of carrying out a mixed route with both types of jobs. In cases with a single skill area with multiple levels, the TTRP resources can be viewed as two levels of the same skill. As such, the same three types of routes described for TTRP could be used for the VRP, with relxation on the order of truck and complete vehicle jobs in mixed routes.

2.1.5 Vehicle Routing Problem with Team Building and Synchronisation

In variations of the VRP, team building refers to instances in which technicians are grouped and travel together for the duration of the planning period, and synchronisation is when individuals (or different teams) meet in a specified location to carry out work together, but do not travel together. Whilst both approaches are useful in instances

where some jobs require multiple resources to be completed, the former is more suited to instances where the majority of jobs are of this type, otherwise resources in a team could spend unnecessary time being idle at jobs that do not require multiple resources. Conversely, synchronisation is most appropriate in cases where the proportion of jobs which require multiple resources is smaller, so these resources are free to follow their own routes once finished working together.

A basic formulation of the VRP with Synchronisation is as follows, based on the Vehicle-Flow fomulation seen in section 2.1.1. In order to consider sunchronised jobs in this formulation, such jobs have been split in to two dummy jobs which represent the same demand, and their start times are set to be equal.

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \tag{2.18a}$$

$$\text{s.t.} \sum_{i \in J} x_{ijk} = \sum_{h \in J} x_{jhk} \qquad \forall j \in J, k \in K \tag{2.18b}$$

$$\sum_{j \in J} x_{0jk} = 1 \qquad \forall k \in K \tag{2.18c}$$

$$\sum_{j \in J} x_{0jk} = \sum_{i \in J} x_{i0k} \qquad \forall k \in K \tag{2.18d}$$

$$\sum_{k \in K} u_{ik} = \sum_{k \in K} u_{jk} \qquad \forall (i,j) \in P^{SYNC} \tag{2.18e}$$

$$u_{ik} + (\delta_i + c_{ij}) x_{ijk} \leq u_{jk} + M(1 - x_{ijk}) \qquad \forall (i,j) \in A_J, k \in K \tag{2.18f}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A_N, k \in K \tag{2.18g}$$

$$u_{jk} \in \mathcal{R} \qquad \forall j \in J, k \in K \tag{2.18h}$$

where the binary variables x_{ijk} take value 1 if vehicle k travels along arc (i,j), and the continuous variable u_{jk} represents the start time of vehicle k at job j. c_{ij} represents the travel time for arc (i,j), δ_j the duration of job j, and P^{SYNC} the set of pairs of jobs (i,j) which need to be synchronised. Constraint (2.18b) ensure that the same resources arrive and depart each job, and (2.18c) and (2.18d) ensure this is exactly once

per technician at the base node. (2.18e) fixes the start time of synchronised jobs to be the same, and (2.18f) maintains the correct sequence of jobs completed by each vehicle k, eliminating the need for tradition subtour elimination constraints. Very little work exists on considering the VRP with Synchronisation and without any other extensions, and as such this formulation has been adapted from the VRPTW with Synchronisation presented in Afifi et al. [2016].

Most commonly synchronisation is combined with time windows, one of the most widely researched extensions of the VRP. Bredström and Rönnqvist [2008] propose a heuristic for solving the VRP with time windows, synchronisation and job precedence, with which they found that changes in the proportion of jobs requiring synchronisation does not have a notable impact on the ability to find a feasible solution to the problem, provided the number of available resources is increased accordingly. Afifi et al. [2016] consider the VRP with time windows and synchronisation, and propose a Simulated Annealing algorithm with local search which compares competitively with existing methods in the literature. Similarly, Liu et al. [2019] consider the same extensions of the VRP, and present an Adaptive Large Neighbourhood Search approach. Eveborn et al. [2006] present a VRP with time windows, synchronisation and skill levels and apply it to the problem of staff planning for home care, although synchronisation is handled by fixing the start times of such jobs before solving, eliminating the need for synchronisation constraints. For more discussion of the finer details around diferent types of synchonisation within the VRP, the reader is directed to Drexl [2012] who provides a comprehensive survey of VRPs with synchronisation.

Based on the work presented in Kovacs et al. [2012], the formulation for the VRP with Team Building is as follows:

$$\min \sum_{t \in T} \sum_{(i,j) \in A} c_{ij} x_{ijt} \tag{2.19a}$$

$$\text{s.t.} \sum_{i \in J} x_{ijt} = \sum_{h \in J} x_{jht} \qquad \forall j \in J, t \in T \tag{2.19b}$$

$$\sum_{j \in J} x_{0jt} = 1 \qquad \forall t \in T \qquad (2.19c)$$

$$\sum_{j \in J} x_{0jt} = \sum_{i \in J} x_{i0t} \qquad \forall t \in T \qquad (2.19d)$$

$$u_{it} + (\delta_i + c_{ij}) x_{ijt} \leq u_{jt} + M(1 - x_{ijt}) \qquad \forall (i,j) \in A_J, t \in T \qquad (2.19e)$$

$$\sum_{t \in T} z_{kt} \leq 1 \qquad \forall k \in K \qquad (2.19f)$$

$$q_j y_{jt} \leq \sum_{k \in K} z_{kt} \qquad \forall j \in J, t \in T \qquad (2.19g)$$

$$x_{ijt} \in \{0,1\} \qquad \forall (i,j) \in A_N, t \in T \qquad (2.19h)$$

$$y_{jt} \in \{0,1\} \qquad \forall j \in J, t \in T \qquad (2.19i)$$

$$z_{kt} \in \{0,1\} \qquad \forall k \in K, t \in T \qquad (2.19j)$$

$$u_{jt} \in \mathcal{R} \qquad \forall j \in J, t \in T \qquad (2.19k)$$

with decision variables:

$$x_{ijt} = \begin{cases} 1 & \text{team } t \text{ traverses arc } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

$$y_{jt} = \begin{cases} 1 & \text{team } t \text{ is assigned to job } j \\ 0 & \text{otherwise} \end{cases}$$

$$z_{kt} = \begin{cases} 1 & \text{technician } k \text{ is assigned to team } t \\ 0 & \text{otherwise} \end{cases}$$

u_{jt} = the start time of team t at job j

c_{ij} and δ_j are as defined in (2.18), and q_j is the minimum number of technicians required for job j. As in (2.18), (2.19b)-(2.19d) ensure the correct number of teams leaving and departing all nodes including the base, and (2.19e) maintains the correct sequence of jobs completed by each team t. (2.19f) and (2.19g) are the Team Building constraints, which ensure each technician is assigned to at most one team, and the minimum number of technicians is met for job j by the assigned team.

Like synchronisation, there is limited literature around team building, and most of the

existing work is focused on the Scheduling problem rather than the VRP. Bellenguez and Néron [2004] consier a Scheduling problem with both skill areas and skill levels, in which team building can be used to meet the skill requirement of jobs. Cordeau et al. [2010] and Fırat and Hurkens [2012] both present their solutions to the 2007 ROADEF Challenge, which are scheduling problems with skill levels, precedence, team building and outsourcing. Although these are not VRP problems, the constraints used for team building can be adapted for use with the VRP. Kovacs et al. [2012] is the only known publication which explicitly incorporates team building in to the VRP formulation; it is considered in conjunction with time windows and skill levels, and an Adaptive Large Neighbourhood Search algorithm is presented which achieves high quality solutions in short computation times.

2.2 Solution Methods

As one of the most studied problems in Operational Research, it is unsurprising that the VRP has been solved using a variety of different methods since its introduction by Dantzig and Ramser. These methods can be separated in to three distinct categories: exact algorithms, heuristics (including metaheuristics) and the group of hybrid exact and heuristic methods known as matheuristics. All three have been widely used to solve the VRP, and in this section I will provide an overview of some of the key algorithms from each area.

2.2.1 Exact Algorithms

For integer programming problems, the most commonly used exact solution methods are Branch and Bound (BB) based algorithms, first introduced by Land and Doig [1960], Cutting Planes (CP) from the work of Gomory [1960], and Column Generation (CG) from Dantzig and Wolfe [1960]. Two of these, BB and CP, were combined by Laporte et al. [1985] to form the Branch and Cut algorithm (BC), after Laporte et al. [1984] found that using Cutting Planes alone was not sufficient to solve instances above 60 nodes of the distance constrained VRP. Similarly Desrosiers et al. [1984] combined BB

and CG to create the Branch and Price algorithm (BP), and these methods now form the foundation of most of the exact algorithms studied in the literature for Integer Programming, and more specifically the VRP.

Branch and Price algorithms use the tree based structure of Branch and Bound, but use Column Generation to solve the subproblems at each node rather than the traditional LP solvers like the simplex method. The use of Column Generation requires the VRP formulation in question to be written as a set-partition problem, as described in section 2.1.2. At the root node, a restricted version of the Master Problem is solved, with only a subset of possible routes $\Omega' \subset \Omega$. At each subsequent node, the Pricing Problem is solved, which aims to find any routes $r \in \Omega \setminus \Omega'$ with negative reduced costs c_r, or to prove that no such routes exist. If a suitable route r is found, this column is added to the restricted master problem, and this is reoptimised. When no such routes can be found, branching takes place if the integrality constraints of the original problem are not satisfied, and column generation using the pricing problem is completed at the next node. The exact branching strategies used vary between authors, and are often dependent on the exact variation of the VRP being considered. Dell'Amico et al. [2006] consider the VRP with simultaneous distribution and collection and test three different branching strategies for the BP algorithm, concluding that a mixed approach works best, depending on the structure of the routes in the latest solution. Azi et al. [2010] consider the VRPTW with multiple use of vehicles, with a hierarchical branching strategy: first they branch on the number of vehicles in the solution k if k is non-integer; if this is not possible they branch on a customer whose value (the sum of all routes they included in) is non-integer; failing both of these they branch on an arc that has fractional flow. The latter branching technique is also used by Gutiérrez-Jarpa et al. [2010] for the VRPTW with deliveries and selective pickups.

In terms of performance, Branch and Price methods have been found to be capable of handling VRP problems and variations of a reasonable size. Gutiérrez-Jarpa et al. [2010] found their method consistently solved instances with 25 nodes and sometimes up to 50, with the slowest taking approximately 25 minutes and most much quicker than this. Similarly Dell'Amico et al. [2006] fount that their BP method worked well for the

given variation of the VRP, with most instances of up to 40 nodes solving in less than 20 minutes, although some took up to four hours. Azi et al. [2010] reported mixed results with problems of 25 and 40 nodes, with some taking a matter of minutes to solve, and others taking several hours, with a maximum of over 7 hours.

An extension of the Branch and Price family of algorithms is Branch Price and Cut algorithms (BPC): these follow the same steps as the BP described above, but with the addition of cutting planes at each node to strengthen the linear relaxations.

2.2.2 Heuristics and Matheuristics

In recent years, a large proportion of work around the VRP has been dedicated to the development and adaptation of heuristic methods for variants of the VRP. These represent some of the state of the art methods, and can be used on instances too large or complex to be solved exactly in a reasonable time. This section will provide a brief overview of the latest heuristic methods presented in the literature, and will then move on to the more specific area of matheuristics.

Continuous advancements in computing technology have created a rapidly changing landscape of heuristic methods, with the latest algorithms able to run on computers that would not have been possible just a number of years ago. Laporte et al. [2000] and Cordeau et al. [2002] provide a concise overview of the earlier state of the art heuristic methods, but more recent techniques will be the focus of this review as these are the most relevant to the problem studied in this work, and the computational power that is now readily available. Koç et al. [2015] present a hybrid evolutionary algorithm for the VRP with time windows and a heterogeneous fleet (similar to the concept of skill levels introduced in section 2.1.4). This particular evolutionary algorithm combines a number of different existing metaheuristic techniques, including the traditional savings algorithm by Clarke and Wright [1964] and an Adaptive Large Neighbourhood Search (ALNS) to achieve competitive solution times across benchmark problem instances. Afifi et al. [2016] and Liu et al. [2019] both propose heuristic approaches for the VRP with time windows and synchronised visits, the former using Simulated Annealing (SA) and the

latter another ALNS approach. Both report computational results which indicate the ability to outperform existing approaches on a number of benchmark instances. Another variant of the VRP which is a focus for new heuristic methods is the two-echelon VRP (2EVRP), in which a problem is considered in two levels: an upper level in which one depot serves a number of intermediate distribution facilities, and a lower level in which customers are each served from one of these intermediate facilities. Breunig et al. [2016] proppose a Large Neighbourhood Search heuristic for this problem, reporting the best known solution for 95% of benchmark 2EVRP problems tested. Belgin et al. [2018] extend the 2EVRP to include simultaneous pickup and delivery, and combine variable neighbourhood descent and local search as a proposed solution method.

A number of people have also made use of Matheuristic methods to tackle the VRP and related problems. There is no fixed structure required of such algorithms, they simply describe the broad family of algorithms which include elements of both mathematical programming techniques and heuristic methods to achieve a solution.

Villegas et al. [2013] present a matheuristic to solve the Truck and Trailer Routing Problem (TTRP), presented in section 2.1.4, a special case of the VRP which jobs can be visited either by a complete vehicle (truck and trailer) or just a truck. They present a two stage approach, beginning with a Greedy Randomised Adaptive Search Procedure (GRASP) and Iterated Local Search (ILS); the former is used to construct tours for the TTRP, and the latter to find improvements to these tours. Stage two is a set partitioning problem (as introduced in section 2.1.2), where the set Ω is taken from the routes found in the first stage. Kramer et al. [2015] also present a matheuristic which makes use of a set partitioning formulation, in this case to solve the Pollution Routing Problem. The routes of the set Ω are found using an ILS procedure, combined with a speed optimisation algorithm (SOA) to optimise speeds on the routes found. Unlike Villegas et al. however, the method presented by Kramer et al. loops between the heuristic ILS-SOA stage and the SP formulation until given stopping conditions are met, whereas the algorithm presented by Villegas et al. performs each step only once, with no return to previous stages in order to find further improvements.

Archetti et al. [2017] present a matheuristic for the Multivehicle Inventory Routing

Problem (MIRP), which comprises three stages: first they relax some routing requirements on the full formulation (the exact formulation used is dependent on the number of jobs in the problem), this is solved until a feasible solution to the MIPR is found or can be induced using a given procedure. In the event that this approach does not achieve a feasible solution, a failsafe heuristic is provided to ensure feasible solution is found as a starting point for the second phase. The second phase uses a tabu search to find improved solutions which are permitted to violate some constraints, with feasibility being recovered using one of two given procedures. Finally the last phase uses information collected in the tabu search to focus on the most promising parts of the solution space, using another MILP formulation to solve the problem for the final time. Chitsaz et al. [2019] present an approach which decomposes the full Assembly, Production and Inventory Routing Problem in to three subproblems, each being solved separately with different methods and then combining to form a solution to the full problem. The first is a special case of the lot sizing problem to determine the set up schedule; this step is performed only once at the beginning of the procedure. The second and third subproblems, used to choose node visits and shipment quantities and then to solve a series of separate VRPs respectively, are then repeated until a given stopping condition is met.

Although they do not represent all the matheuristic methods applied to the VRP and related problems, those presented here still make use of a variety of different techniques. Each matheuristic is tailored to the problem for which it is designed, and therefore they would not be easily adapted to other problems, but they do provide several examples of the broadness of matheuristic techniques, and how they can successfully be implemented to routing related problems.

2.3 Other Relevant Literature

2.3.1 Scheduling Problem

The work presented in this thesis also bears a resemblance to the family of scheduling problems, another widely studied area of Operational Research. Some example works

such as Bellenguez and Néron [2004], Cordeau et al. [2010] and Fırat and Hurkens [2012] have already been introduced in section 2.1.5, as these scheduling problems tackle the problem of team-building which is also relevant to the VRP. There are also a number of other works which include extensions to the simple scheduling problem, and these too are similar to the VRP extensions considered in this work.

The earliest work in scheduling problems, such as Bowman [1959] and Manne [1960], present a MILP formulation for the problem of scheduling consecutive jobs on multiple machines. Much like the VRP, the range of different scheduling problems and extensions tackled since these early publications is vast; a handful of these extensions will be highlighted here, although the reader is referred to the cited work for more specific details and problem formulations.

Bellenguez-Morineau and Néron [2007] and Bhulai et al. [2008] both present scheduling problems that handle skill levels and variable duration, the latter commonly used in service industries such as call centres where call duration can vary hugely. Bellenguez-Morineau and Néron also include precedence and synchronisation, although the work is theoretical so no proposed application is presented alongside the proposed Branch and Bound method. Caramia and Giordani [2009] and Avramidis et al. [2010] also tackle scheduling problem with skill levels, although in both of these cases a heuristic solution method is proposed as opposed to the exact methods used by Bellenguez-Morineau and Néron and Bhulai et al. The work presented by Cordeau et al. [2010] and Fırat and Hurkens [2012] are similar in their extensions as they were both proposed solutions to the 2007 ROADEF challenge. These works consider the scheduling problem with skill levels, precedence, team building, job priority and outsourcing, with Cordeau et al. proposing an adaptive large neighbourhood search method, and Fırat and Hurkens a two-stage exact solution method. A breakdown of all of the works mentioned here can be seen alongside their VRP counterparts in the table 2.1.

2.3.2 Knapsack Problem

Another approach considered (although not tested) in this work is a hierarchical approach to decompose large problem instances in to smaller problems which can then be solved with the main VRP formulation and matheuristic approach. This relates to the family of Knapsack Problems (KP), of which Kellerer et al. [2004] provides a comprehensive overview. Of relevance in this work are the Multidimensional Knapsack Problem (d-KP) and the Multiple Knapsack Problem (MKP). The d-KP considers multiple resource or capacity constraints on a single knapsack, with the following basic formulation:

$$\max \sum_{j=1}^{n} p_j x_j \tag{2.20}$$

$$\text{s. t.} \sum_{j=1}^{n} w_{ij} x_j \leq c_i \qquad \forall i \in \{1, \ldots, d\} \tag{2.21}$$

$$x_j \in \{0, 1\} \qquad \forall j \in \{1, \ldots, n\} \tag{2.22}$$

Where x_j is a binary variable to indicate whether item j is included in the knapsack, p_j is the objective wight of item j, w_{ij} is the weight of item j in constraint i, and c_i the capacity of constraint i. This allows for the consideration of multiple constraints, such as weight and space for a traditional knapsack problem, or any other combination of required capacity or resource constraints in an alternative application. The Multiple Knapsack Problem (MKP) generalises the original KP across multiple knapsacks m of different capacity, with the following basic formulation:

$$\max \sum_{k=1}^{m} \sum_{j=1}^{n} p_j x_{jk} \tag{2.23}$$

$$\text{s. t.} \sum_{j=1}^{n} w_j x_{jk} \leq b_k \qquad \forall k \in \{1, \ldots, m\} \tag{2.24}$$

$$\sum x_{jk} \leq 1 \qquad \forall j \in \{1, \ldots, n\} \tag{2.25}$$

$$x_{jk} \in \{0,1\} \qquad \forall j \in \{1,\ldots,n\}, \forall k \in \{1,\ldots,m\} \tag{2.26}$$

Where x_{jk} is a binary variable to indicate whether item j is included in knapsack k, p_j is the objective wight of item j, w_j is the weight of item j, and b_k the capacity of knapsack k. These two formulation can also be easily combined to create a multidimensional multiple knapsack problem if required. Puchinger et al. [2010] and Lust and Teghem [2012] provide a survey and comparison of solution methods for these two problems resprctively, including exact, heuristic and matheuristic approaches. Due to the fact that Knapsack Problems form only a small part of this work, they are not presented in more detail here, but the reader is referred to Kellerer et al. [2004] for an overview of many knapsack problem extensions and previous methods, as well as the two survey papers mentioned prevously.

2.3.3 Objective Function Weighting

One challenge of many problems in which there are several distinct objectives is to combine these in a way that they can be represented in a single objective function. One research area that can aid with this is Multiple Criteria Decision Making (MCDM), which provides methods that can help in determining the relative importance of different criteria within a decision that needs to be made. Within the context of the VRP, such methods can be used to determine objective function weightings that reflect these importances. Wallenius et al. [2008] provide an overview of a variety of different MCDM techniques, including Analytic Hierarchy Process (AHP) which is the focus of this section.

AHP, as described by Ishizaka and Nemery [2013], seeks to arrange a multiple criteria decision into a hierarchical structure, where different decision areas are broken down in to sub-criteria. Decision makers are then required to perform a pairwise comparison of all distinct pairs of sub-criteria, from which an overall importance ranking can be generated. Weber et al. [1988] state the importance of maintaining a balanced structure within the hierarchy to ensure results that are as fair as possible, as large imbalances

between the number of sub-criteria in each group can affect the overall weighting of that group. Methods such as these are by no means perfect, with possible issues such as cyclical priority order, but despite this Wallenius et al. [2008] identify AHP as one of the most widely used techniques in this area. No more MCDM methods will be presented here, and a full explanation of how this approach could be applied to the current work is provided in section 6.3.

2.4 Paper Comparison Table

Table 2.1 provides a visual overview of the works that have been mentioned throughout this literature review, and table 2.2 provides a definition of each category in the comparison table. It is worth noting here that it is not practical to include all extensions of the VRP considered in the works mentioned here, as the table would quickly become very large. Instead only those extensions relevant to this work have been included in this summary, for a more detailed description of cited works please see section 2.1.

Table 2.1: Table of VRP and Scheduling Extensions

	Routing	Scheduling	Vehicle Flow Formulation	Set Partition Formulation	Heuristic	Exact	Skill Levels	Time windows	Multiple Resources	Multiple Routes	Precedence	Team-building	Variable duration	Multi-period	Job selection	Time between return visits	Priority of jobs	Outsourcing	Synchronisation	Overtime	Unplanned work	Open end point	Same resource on return visits	Tools and spare parts
Dantzig and Ramser [1959]	x			x		x			x															
Foster and Ryan [1976]	x		x			x			x					x		x								
Baker [1983]	x		x			x		x																
Kolen et al. [1987]	x				x			x	x															
Solomon [1987]	x				x			x	x															
Taillard et al. [1996]	x				x				x	x														
Begur et al. [1997]	x				x		x	x	x					x		x								
Bramel and Simchi-Levi [1997]	x			x		x		x																
Tsang and Voudouris [1997]	x				x		x		x				x											
Weigel and Cao [1999]	x				x		x	x	x	x	x									x				
Xu and Chiu [2001]	x		x		x		x	x	x						x		x			x				
Blakeley et al. [2003]	x				x		x		x							x					x			
Bellenguez and Néron [2004]		x					x		x		x	x												
Eveborn et al. [2006]	x			x	x		x	x	x			x							x				x	

Continued on next page

Table of VRP and Scheduling Extensions

	Routing	Scheduling	Vehicle Flow Formulation	Set Partition Formulation	Heuristic	Exact	Skill Levels	Time windows	Multiple Resources	Multiple Routes	Precedence	Team-building	Variable duration	Multi-period	Job selection	Time between return visits	Priority of jobs	Outsourcing	Synchronisation	Overtime	Unplanned work	Open end point	Same resource on return visits	Tools and spare parts
Alvarenga et al. [2007]	x			x	x			x	x															
Bellenguez-Morineau and Néron [2007]		x				x	x		x		x		x						x					
Bhulai et al. [2008]		x				x	x		x				x											
Bredström and Rönnqvist [2008]	x		x		x			x	x		x								x					
Bostel et al. [2008]	x				x			x	x					x							x			
Caramia and Giordani [2009]		x			x		x		x				x											
Avramidis et al. [2010]		x			x		x		x															
Azi et al. [2010]	x		x			x		x	x	x					x									
Cordeau et al. [2010]		x			x		x		x		x	x		x			x	x						
Cappanera et al. [2011]	x		x			x	x		x															
Fırat and Hurkens [2012]		x				x	x		x		x	x					x	x						
Kovacs et al. [2012]	x		x		x		x	x	x			x						x						
Pillac et al. [2013]	x				x		x	x	x															x
Villegas et al. [2013]	x			x	x	x	x		x															

Continued on next page

Table of VRP and Scheduling Extensions

	Routing	Scheduling	Vehicle Flow Formulation	Set Partition Formulation	Heuristic	Exact	Skill Levels	Time windows	Multiple Resources	Multiple Routes	Precedence	Team-building	Variable duration	Multi-period	Job selection	Time between return visits	Priority of jobs	Outsourcing	Synchronisation	Overtime	Unplanned work	Open end point	Same resource on return visits	Tools and spare parts
Azi et al. [2014]	x				x			x	x	x					x									
Afifi et al. [2016]	x		x		x			x	x										x					
Lalla-Ruiz et al. [2016]	x		x			x	x		x		x											x		
Archetti et al. [2017]	x		x		x				x															
Xie et al. [2017]	x		x		x		x	x																
Shelbourne et al. [2017]	x			x	x			x	x															
Reil et al. [2018]	x		x		x			x	x															
Bulhões et al. [2018]	x		x		x			x	x															
Louveaux and Salazar-González [2018]	x		x			x			x	x			x											
Ticha et al. [2019]	x				x			x																
Liu et al. [2019]	x		x		x			x	x										x					
Zhang et al. [2019]	x		x	x		x		x	x				x											
Pessoa et al. [2020]	x			x		x			x															

Table Definitions

Table 2.2: Definitions of column headings seen in table 2.1

Category	Definition
Routing	Considers travel time between jobs
Scheduling	No travel time, purely job scheduling
Vehicle Flow Formulation	These are vehicle routing formulations which use the vehicle flow approach described in 2.1.1 (greyed out cells indicate those for which formulations were not given)
Set Partition Formulation	These are vehicle routing formulations which use the set partition approach described in 2.1.2 (greyed out cells indicate those for which formulations were not given)
Heuristic	Formulations which have been solved using heuristic methods
Exact	Formulations which have been solved using exact methods
Skill Levels	Formulations which have been solved using exact methods (both pre-existing solvers such as CPLEX and methods developed foe the specific problem)
Time windows	Visits to jobs must be in a certain time window
Multiple resources	Multiple resources being scheduled/routed (e.g. technicians/vehicles)
Multiple routes	Multiple trips in a single planning period (i.e. returning to base and starting a new route). Sometimes referred to as *multiple use of vehicles*.

Precedence	Jobs must be completed before/after certain other jobs
Team-building	Jobs require multiple resources, and resources are grouped and work/travel together for the planning period (e.g. day)
Variable duration	Time to complete job is dependent on resource skill level
Multi-period	Multiple planning periods output with each solution (e.g. week long planning period, solve for multiple weeks)
Job selection	Not all jobs can be completed, so which jobs to complete must be selected
Time between return visits	Time between visits is restricted (either by LB or UB)
Priority of jobs	Jobs have a priority level, and the objective function favours high priority jobs
Outsourcing	Jobs can be outsourced to external resource
Synchronisation	Resources must meet up and synchronise work in certain locations
Overtime	Possibility of resources being assigned overtime
Unplanned work	Makes allowances for the presence of unknown, last minute jobs to be included in the schedule
Open end point	Route does not have to finish at the start point
Same resource on return visits	Where possible, customers are assigned to the same resource for continuity
Tools and spare parts	Jobs have requirements for tools and parts, and each resource has an availability of each. Unlike skill levels, these availabilities can fluctuate with time (usage and restocking)

Chapter 3

Problem Description

The RNLI currently operates with a divisional structure, which can be seen in fig. 3.1. There are five divisions (although some are divided again for the purposes of scheduling technicians), and each of these has a dedicated team of System Technicians (STs), and a Divisional Maintenance Manager (DMM) who is responsible for all the maintenance work that goes on throughout their respective division. It is the work schedules of these STs that are the focus of this scheduling and routing work. Each division consists of approximately ten STs, each of whom carries out on average between one and ten jobs per week. In March 2017, I went to the RNLI DMM's quarterly meeting in Kinsale, Ireland. The aims of this meeting were twofold: to introduce the DMMs to the work that I am doing, and how this ties in with the problem of ST scheduling, and also to gain their input in to the problem. Although they do not have the technical understanding of the methods used, their experience and insight in to the business are invaluable in terms if creating a model which can be applied in this case.

Here we give an in depth explanation of the problem, including all the requirements that have been identified as a result of meetings with the RNLI. Where necessary, information is given about the specific working practices of the RNLI, and how this ties in to these requirements. All information in this chapter regarding the policies and current operations of the RNLI have been obtained through meetings with Jan Wyglendacz, Principal Maintenance Manager, and DMMs. These requirements are

Figure 3.1: Map of RNLI Divisions and Stations

outlined below.

Working hours

The expected work and travel time in each day should not exceed the length of a working day for each technician. In the case of the RNLI, STs are contracted to work 37 hours per week, which includes all travel time between jobs, including the beginning and end of the day. This is divided up as 7.5 hours per day Monday-Thursday, and 7 hours on Friday. In the case where an ST is more than 1.5 hours away from their base location at the end of the day, they are expected to stay away for that night. The exception to travel time being included in the work schedule is the use of any overnight ferries (used for some islands in Scotland).

These working hours are flexible in that they operate flexitime and TOIL policies, but the long run total should still fall within the contracted amount for that period. In reality technicians are often required to work overtime in order to complete all their assigned jobs. If overtime is to be considered explicitly in a problem formulation, it could be modelled with either linear or non-linear constraints, depending on the weighting that such overtime should bear in a final solution.

Daily driving limit

Technicians should not exceed 400 miles driving in any given day. In reality, this is very unlikely, so may not need to be excluded explicitly as a constraint.

Start location

Each technician should start and end their working week in their respective base location. Some STs are contracted from home, and start and end all their routes there, whilst others are contracted from their RNLI Divisional Base (although they may still start their routes from home instead). For the purposes of this problem, we can assume that

all STs will start their week at the contracted base, and any extra time to get there is considered commuting time.

Deadlines

Each job should be completed by its respective deadline. Planned work comes with a deadline date, with the aim to be completed within 30 days either side of this date. Exceptions to this include jobs which are restricted by external legislation, such as MOTs. Emergent or unplanned work is assigned a priority level form one of the following four priority categories: immediate (24 hours); 2-7 days; 8-30 days; next visit (twice/year).

The RNLI operates a policy that no lifeboat should be out of service for more than 48 hours (otherwise a relief lifeboat will be brought in). Any such emergent jobs which fall in to this category must be completed within 48 hours. Such jobs should be given the the priority immediate, which falls within the 48 hours specified by this policy.

Job duration

There is no guarantee that all jobs are shorter than the planning period: many jobs extend over multiple days, and some (particularly unplanned work) may require as much as 5 or more days, with 2 technicians (so > 10 technician days). Expected duration of planned work is output as part of the job list. Duration of emergent work is input by DMMs as jobs come in. Although it is not necessarily the case, expected job duration is calculated independently of the skill level of the assigned technician.

Precedence

For any job with precedence relationships, these must be completed in the correct order by the same technician. Precedence relationships in the current problem relate mostly to the requirement to pick up certain parts/equipment from another location in order to complete a job. In this instance, it is intuitive that these jobs must be completed by

the same technician, where the process of picking up equipment can be viewed as a 'job' with negligible completion time.

It is possible that precedence relationships may exist within individual stations, but these are not necessarily required to be completed by the same technician. It is highly unlikely that there will exist precedence relationships between actual jobs on different stations (as opposed to dummy jobs representing the collection of equipment), as stations operate independently.

Skill requirements

The skill requirements of each job must be met by the technician(s) assigned to it. There are jobs which require more than one technician, especially among unplanned work. There are also instances in which jobs only require one technician, but two may be assigned, in which case the expected total duration is halved (i.e. the total technician time is constant).

Team Building and Synchronisation

Team building should be carried out where required, e.g. for jobs which require multiple technicians. STs are quite often grouped to travel together: sometimes this is for the purposes of meeting multiple technician requirements, or sometimes to avoid lone working for extended periods (resulting in two technicians being sent to a location in which only one can complete the work). This happens more in locations where distances between jobs are greater, e.g. Scotland.

Synchronisation may also be used in cases where multiple STs are required for one job, but will not travel their whole routes together. In some cases, STs will discover the need for additional assistance whilst carrying out a job, and this must be covered by another ST travelling to join them, but this cannot be planned for, and is therefore difficult to capture in the formulation.

Nights away

Technicians are assumed to stay away if their time/distance to base is greater than 1.5 hours. They may spend a maximum of four nights away in a row (Monday-Friday), and they must start and end each week in their base location. In reality, some technicians choose to drive home despite being further away than the 1.5 hour threshold, but as this is against company policy, this time is not counted as part of their working hours.

Training

Time should be allowed for STs to complete necessary training, and also in some cases to spend time training others. ST training can be viewed as a job which must be completed by a particular technician, and therefore should be relatively straightforward to include in the formulation. Training others may include apprentices, but they will always travel with another ST to jobs, and therefore should not be viewed as an ST which can be scheduled independently.

Objective Function

The following were highlighted as aspects that should be included in the objective function. As the intention is to use a single objective function, these will need to be combined in to a single function. To achieve this, work will be carried out to determine weightings for each of the distinct elements; this work is described in more detail in section 6.3.

- Cost of travel: $\psi\tau_{ij}$ for all arcs (i, j) in the final solution
- Nights away: if $\tau_{jh_k} > \delta$, where $j \in J$ is the last job in the day for technician k, then the technician is assumed to stay away from home for that night, at cost ϕ
- Lateness: if the completion time of a job is greater than its deadline f_j, then some penalty is incurred
- Return visits: precedence relationships mean that one location may need to be visited multiple times in one trip, but these should be minimised. There is cur-

rently a lot of inefficiency in this area, often stemming from problems such as incorrect delivery of parts. It is very difficult to capture circumstances such as this in a model of this type; occurrences like this would be dealt with as they emerge by the STs and DMM responsible. In terms of considering return visits within planned work, it may not be necessary to include this explicitly in the objective function. If distance is being considered, then multiple visits to the same location will be unlikely due to the associated increase in travel. In the case where a solution does require a return visit to a particular place, it will be because that represents a good (and potentially optimal) solution for the given set of jobs.

- Overtime: They do operate a paid overtime policy, and this should be minimised. Flexitime and TOIL are sometimes used to balance overtime.

Chapter 4

Routing and Scheduling Formulation

In this chapter, we present a full formulation for the Routing and Scheduling problem described in chapter 3. This is something that could be used either in conjuction with the hierarchical approach, described in section 5.4, or as a stand alone formulation, provided the instance size is computationally feasible. Before introducing the formulation, details are given about the notation used, as well as a description of some preprocessing that can take place before the formulation is solved.

4.1 Definitions

4.1.1 Terminology

Table 4.1 provides definitions of all the terminology that will be used for this problem throughout the rest of this work.

Node	This refers to any node in the network, which could be a job or a depot node.
Job	This refers specifically to a job node. Note: all jobs are represented by different nodes, even if they are in the same location. The travel time within these groups of nodes will be zero.
Depot	This refers to a node which is a start/end location for one or more technicians.
Technician	These are the people performing the tasks (they replace the notion of vehicle from the VRP).
Skill Domain	These are the different areas that technicians can be trained in.
Skill Level	These represent the level to which technicians are trained. The same set of skill levels are used for all skill domains.

Table 4.1: Table of terminology in the problem

4.1.2 Sets

Below is the table of sets in the full formulation. Those marked with a $*$ are calculated from the data that is provided, and thus do not need to be present in the dataset that is provided as input. The calculation of these sets is outlined in section 4.3.

$*$	V	=	set of all nodes in the problem
	J	=	set of all job nodes
	H	=	set of all depot nodes
	K	=	set of all technicians
$*$	$\mathcal{K}$	=	set of permissible pairs (j, k) such that k can do job j (either alone or in a pair)
$*$	$\mathcal{K}'$	=	set of permissible triples (j, k, k') such that k and k' together can do job j
	S	=	set of all skill domains
	L	=	set of all skill levels
$*$	T	=	set of hours of full planning period
$*$	T_w	=	set of hours in week w
$*$	T_{wd}	=	set of hours in day d, where d is in week w
	W	=	set of all weeks in full planning period
	D	=	set of days in each week, $\{1, \ldots, 5\}$

$*$	$\mathcal{I}$	=	the set of allowable tuples (i, j, k, t) for the decision variable x^t_{ijk}

* $\bar{\mathcal{I}}$ = the set of allowable tuples (i, j, k, t) for the decision variable $\bar{x}^t_{ijk}$
* F = the set of allowable tuples (j, k, w)
* F' = the set of allowable tuples (j, k, w, d)
* G = the set of allowable tuples (j, k, t)

---

* *sets which are calculated in preprocessing*

In addition, we define the following notation for dealing with subsets of indices in sets which are made up of tuples:

Definition 4.1.1. Let A be a family of sets, $\{A_1, \ldots, A_n\}$, and let $\Delta = \prod_{i=1}^{n} A_i$, i.e., the set of tuples $(a_1, \ldots, a_n)$, where $a_i \in A_i$ for all i. let $B = \{B_1, \ldots, B_m\} \subset \{A_1, \ldots, A_n\}$. We define the set $\Delta_{(B)}$ as follows:

$$\Delta_{(B)} = \prod_{i=1}^{n-m} (A \setminus B)_i \tag{4.1}$$

In other words, $\Delta_{(B)}$ is the set of tuples in Δ, with the elements from B removed.

Example 4.1.2. Let $A = (w, x, y, z) \subset \mathbb{R}^3$, then $A_{(x,z)} = \{(w, y) \mid (w, x, y, z) \in A\}$

This notation is introduced to improve the readability and clarity of the final problem formulation.

4.1.3 Decision Variables

$$x^t_{ijk} = \begin{cases} 1 & \text{technician } k \text{ leaves job } i \text{ travelling to job } j \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$\bar{x}^t_{ijk} = \begin{cases} 1 & \text{technician } k \text{ arrives at job } j \text{ travelling from job } i \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$y^t_{jk} = \begin{cases} 1 & \text{technician } k \text{ is at job } j \text{ at time } t \text{ (working, not idle)} \\ 0 & \text{otherwise} \end{cases}$$

u_j	=	start time of job j
z_j	=	$\begin{cases} 1 & \text{job } j \text{ has two technicians assigned to it} \\ 0 & \text{otherwise} \end{cases}$
λ_{jk}^{wd}	=	$\begin{cases} 1 & \text{technician } k \text{ stays away from home at job } j \text{ in week } w, \text{ day } d \\ 0 & \text{otherwise} \end{cases}$
σ_{jk}^{wd}	=	hours of job j completed by technician k on week w, day d
ρ_{jk}^{wd}	=	$\begin{cases} 1 & \text{job } j \text{ performed by technician } k \text{ is split in week } w, \text{ from day } d \text{ to } d+1 \\ 0 & \text{otherwise} \end{cases}$
$p_{jkk'}$	=	$\begin{cases} 1 & \text{job } j \text{ is completed by } k \text{ and } k' \text{ as a pair} \\ 0 & \text{otherwise} \end{cases}$
ϵ_j	=	$\begin{cases} 1 & \text{job } j \text{ is finished past it's finish time } f_j, \text{ but within allowable lateness} \\ 0 & \text{otherwise} \end{cases}$

It is important to note that not all of these variables need to be defined for all possible combinations of their indices; they can instead just be defined for those combinations which are feasible given the problem constraints, therefore reducing the problem size. The full details of which variables can be reduced and the sets across which they are defined is outlined in section 4.3.

4.1.4 Parameters

Below is the table of parameters in the full formulation. Those marked with a $*$ are calculated from the data that is provided, and thus do not need to be present in the dataset that is provided as input. The calculation of these parameters is outlined in section 4.3.

	δ_{jn}	=	duration of job j with n technicians assigned to it, in hours, $n \in \{1, 2\}$ *n.b.* – for notational purposes, the value $\bar{\delta}_j = \max_{n\in\{1,2\}}\{\delta_{jn}\}$, and $\underline{\delta}_j = \min_{n\in\{1,2\}}\{\delta_{jn} \mid \delta_{jn} > 0\}$
$*$	η^t_{jn}	=	number of nights that job j will be split over when started at hour t in a day ($t \in \{1, \ldots, 7\}$) with n technicians *n.b.* – for notational purposes, the value $\bar{\eta}^t_j = \max_{n\in\{1,2\}}\{\eta^t_{jn}\}$, and $\underline{\eta}^t_j = \min_{n\in\{1,2\}}\{\eta^t_{jn} \mid \eta^t_{jn} > 0\}$

	$[e_j, f_j]$	=	time window of job j, $e_j < f_j$
	c_j	=	allowable lateness of a job
	$[\alpha_j, \beta_j]$	=	number of technicians required/allowed for job j $(\alpha_j, \beta_j \in \{1, 2\}, \alpha_j \leq \beta_j)$
	τ_{ij}	=	travel time from node i to j

---

	v_{ksl}	=	$\begin{cases} 1 & \text{techinican } k \text{ is qualified in skill area } s \text{ to level } l \\ 0 & \text{otherwise} \end{cases}$
	r_{jsl}	=	$\begin{cases} 1 & \text{job } j \text{ requires skill area } s \text{ to level } l \\ 0 & \text{otherwise} \end{cases}$
$*$	q_{jk}	=	$\begin{cases} 1 & \text{technician } k \text{ is qualified to do job } j \\ 0 & \text{otherwise} \end{cases}$
$*$	$\bar{q}_{jkk'}$	=	$\begin{cases} 1 & \text{technician } k \text{ and } k' \text{ combined are qualified to do job } j \\ 0 & \text{otherwise} \end{cases}$
			n.b. – these can be calculated from v_{ksl} and r_{jsl}: $q_{jk} = 1$ if $v_{ksl} \geq r_{jsl}\ \forall s \in S, l \in L$ $\bar{q}_{jkk'} = 1$ if $v_{ksl} + v_{k'sl} \geq r_{jsl}\ \forall s \in S, l \in L$

	Ω^d	=	number of hours in day d in any given week (a given day d in each week is assumed to be the same, i.e. Ω_d is not dependent on w)
	Ω	=	number of hours in working week
	ω_k	=	weekly working capacity of technician k, in hours
	ω_k^d	=	working capacity of technician k on day d in each week, in hours (a given day d in each week is assumed to be the same, i.e. ω_k^d is not dependent on w)
	h_k	=	base node of technician k
	π_{kh}	=	technician k has base node h
	γ	=	the maximum travel distance to base before someone must stay away for the night
*	μ_{jk}	=	$\begin{cases} 1 & \text{technician } k \text{ must stay away from home if they end a day at job } j \\ 0 & \text{otherwise} \end{cases}$ *n.b.* – this can be calculated from τ_{jh_k}: $\mu_{jk} = 1$ if $\tau_{jh_k} > \gamma$

---

*	$\bar{t}_{wd}$	=	the first hour in week w, day d
*	$\underline{t}_{wd}$	=	the last hour in week w, day d
*	w_t	=	the week that time t falls in
*	d_t	=	the day of the week that time t falls in, $d \in D$

	ψ	=	cost per hour of travel
	ϕ	=	cost per night away
	γ	=	equivalent cost of lateness
	θ_m	=	the weighting of the m-th term in the objective function

---

* *parameters which are calculated in preprocessing*

4.2 Assumptions

The following assumptions are used in the formulation presented below. Whilst they do not all align completely to the full RNLI problem, they are necessary for the problem

to be solved using the methods outlined in this thesis.

- Travel times and job durations are fixed.
- No jobs are split over weeks. This is not necessarily true for the RNLI problem, but it is used in the formulation as it currently stands.
- Travel time from job to home is not included in work time - if this is greater than 1.5 hours, they must stay away. If this is less than 1.5 hours, they are assumed to travel to/from home outside of work hours each day.
- Technicians always start and end their route at their base node.

4.3 Preprocessing

Before the full problem formulation is generated for any given dataset, a certain amount of calculations and preprocessing are carried out beforehand, both for ease of notation and to reduce the problem size. In sections 4.1.2 and 4.1.4, the parameters and sets marked with a $*$ are those which are calculated during this stage, and the calculation of these is outlined in tables 4.5 and 4.6 respectively.

Table 4.5: Calculation of parameters in the preprocessing stage

Parameter	**Calculation of Parameter**
η^t_{jn}	$\begin{cases} d_{t+\delta_{jn}} - d_t \in \{1, ..., \lvert D\rvert - 1\} & d_{t+\delta_{jn}} \geq d_t \\ 0 & \text{otherwise} \end{cases}$ *n.b.* This parameter takes value 0 when the start time t of a job j means it can be finished before the end of the day, so no overnight stays are required.
q_{jk}	$\begin{cases} 1 & v_{ksl} \geq r_{jsl} \ \ \forall s \in S, l \in L \\ 0 & \text{otherwise} \end{cases}$
$q_{jkk'}$	$\begin{cases} 1 & v_{ksl} + v_{k'sl} \geq r_{jsl} \ \ \forall s \in S, l \in L \\ 0 & \text{otherwise} \end{cases}$

μ_{jk}	$\begin{cases} 1 & \tau_{jh_k} \geq \gamma \\ 0 & \text{otherwise} \end{cases}$
w_t	$\left\lceil \frac{t}{\Omega} \right\rceil$
d_t	$\min\{d \mid t \mod \Omega > \sum_{i=1}^{d-1} \Omega^d\}$

Table 4.6: Calculation of sets in the preprocessing stage

Set	**Definition of Set**
V	$J \cup H$
$\mathcal{K}$	$\{(j,k) \mid v_{ksl} \geq r_{jsl}\ \forall s \in S, l \in L\} \subset J \times K$
$\mathcal{K}'$	$\{(j,k,k') \mid v_{ksl} + v_{k'sl} \geq r_{jsl}\ \forall s \in S, l \in L\} \subset J \times K \times K$
T	$\{1, \ldots, \Omega\|W\|\}$
T_w	$\{(w-1)\Omega + 1, \ldots, w\Omega\}$
T_{wd}	$\left\{(w-1)\Omega + \left(\sum_{j=1}^{d-1} \Omega^j\right) + 1, \ldots, (w-1)\Omega + \left(\sum_{j=1}^{d} \Omega^j\right)\right\}$
$\mathcal{I}$	$\{(i,j,k,t)\} \subseteq V \times V \times K \times T$ which satisfy the following conditions: ◦ $i \neq j$ ◦ $q_{ik} = 1$ **or** $\sum_{k' \in K} \bar{q}_{ikk'} \geq 1$ **or** $i = h_k$ ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ **or** $j = h_k$ ◦ $t + \tau_{ij} \in [e_j, f_j - \underline{\delta}_j]$ ◦ $t \geq e_i + \underline{\delta}_i$ ◦ $d_{t+\tau_{ij}} + \bar{\eta}_j^{t+\tau_{ij}} \leq 5$ ◦ $d_t \leq d_{t+\tau_{ij}}$ *n.b.* – This is the feasible set of indices for the variable x_{ijk}^t

$\bar{\mathcal{I}}$	$\{(i,j,k,t)\} \subseteq V \times V \times K \times T$ which satisfy the following conditions: ◦ $i \neq j$ ◦ $q_{ik} = 1$ **or** $\sum_{k' \in K} \bar{q}_{ikk'} \geq 1$ **or** $i = h_k$ ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ **or** $j = h_k$ ◦ $t \in [e_j, f_j - \underline{\delta}_j]$ ◦ $t - \tau_{ij} \geq e_i + \underline{\delta}_i$ ◦ $d_t + \bar{\eta}_j^t \leq 5$ ◦ $d_{t-\tau_{ij}} \leq d_t$ *n.b.* – This is the feasible set of indices for the variable $\bar{x}_{ijk}^t$
F	$\{(j,k,w)\} \subseteq J \times K \times W$ which satisfy the following conditions: ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ ◦ $w \in [w_{e_j}, w_{f_j}]$
F'	$\{(j,k,w,d)\} \subseteq J \times K \times W \times D$ which satisfy the following conditions: ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ ◦ $w \in [w_{e_j}, w_{f_j}]$ ◦ **if** $w = w_{e_j}$ **then** $d \geq d_{e_j}$ ◦ **if** $w = w_{f_j}$ **then** $d \leq d_{f_j}$
G	$\{(j,k,t)\} \subseteq J \times K \times T$ which satisfy the following conditions: ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ ◦ $t \in [e_j, f_j]$

Definition of Decision Variables

In order to reduce the number of decision variables in the full formulation, it is possible to only define each set of variables for feasible combinations of their indices. These are outlined in table 4.7. As a result of this, we must ensure that summations and constraints are only performed or defined across allowable combinations of indices, which means the sets across which variables are defined must also be preprocessed. They are outlined in table 4.7.

Table 4.7: Definition of decision variables

Variable	**Indices for which variable is defined**
x^t_{ijk}	defined for all tuples (i, j, k, t) which satisfy the following conditions: ◦ $i \neq j$ ◦ $q_{ik} = 1$ **or** $\sum_{k' \in K} \bar{q}_{ikk'} \geq 1$ **or** $i = h_k$ ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ **or** $j = h_k$ ◦ $t + \tau_{ij} \in [e_j, f_j - \underline{\delta}_j]$ ◦ $t \geq e_i + \underline{\delta}_j$ ◦ $d_{t+\tau_{ij}} + \bar{\eta}_j^{t+\tau_{ij}} \leq 5$ ◦ $d_t \leq d_{t+\tau_{ij}}$
$\bar{x}^t_{ijk}$	defined for all tuples (i, j, k, t) which satisfy the following conditions: ◦ $i \neq j$ ◦ $q_{ik} = 1$ **or** $\sum_{k' \in K} \bar{q}_{ikk'} \geq 1$ **or** $i = h_k$ ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ **or** $j = h_k$ ◦ $t \in [e_j, f_j - \underline{\delta}_j]$ ◦ $t - \tau_{ij} \geq e_i + \underline{\delta}_j$ ◦ $d_t + \bar{\eta}_j^t \leq 5$ ◦ $d_{t-\tau_{ij}} \leq d_t$
y^t_{jk}	defined for all triples (j, k, t) which satisfy the following conditions: ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ ◦ $t \in [e_j, f_j]$
u_j	defined for all $j \in J$
z_j	defined for all $j \in J$
λ^{wd}_{jk}	defined for all tuples (j, k, w, d) which satisfy the following conditions: ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ ◦ $w \in [w_{e_j}, w_{f_j}]$ ◦ **if** $w = w_{e_j}$ **then** $d \geq d_{e_j}$ ◦ **if** $w = w_{f_j}$ **then** $d \leq d_{f_j}$ ◦ $\mu_{jk} = 1$

σ_{jk}^{wd}	defined for all tuples (j, k, w, d) which satisfy the following conditions: ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ ◦ $w \in [w_{e_j}, w_{f_j}]$ ◦ **if** $w = w_{e_j}$ **then** $d \geq d_{e_j}$ ◦ **if** $w = w_{f_j}$ **then** $d \leq d_{f_j}$
ρ_{jk}^{wd}	defined for all tuples (j, k, w, d) which satisfy the following conditions: ◦ $q_{jk} = 1$ **or** $\sum_{k' \in K} \bar{q}_{jkk'} \geq 1$ ◦ $w \in [w_{e_j}, w_{f_j}]$ ◦ **if** $w = w_{e_j}$ **then** $d \geq d_{e_j}$ ◦ **if** $w = w_{f_j}$ **then** $d \leq d_{f_j}$ ◦ $d \neq 5$
$p_{jkk'}$	defined for all triples (j, k, k') which satisfy the following condition: ◦ $q_{jkk'} = 1$
ϵ_j	defined for all $j \in J$

The use of this level of preprocessing will also determine whether certain constraints need to be included in the formulation, as they are included implicitly in the definition of sets and decision variables. Such constraints will be identified as such in the following section.

4.3.1 Reference for Notation

Table 4.8 provides a reference for meaning of any single letter or character used in this formulations described in this section. It does not identify which subscripts are present for particular decision variables, but is instead intended as a quick reference. Detailed descriptions of all the sets, decision variables and parameters in the problem can be found in sections 4.1.2 to 4.1.4 respectively.

Table 4.8: Reference table for notation

Letter	Use	Letter	Use	Letter	Use
A		a		α	no. of techs LB
B		b		β	no. of techs UB
C		c	allowable lateness parameter	γ	night away distance
D	days in each week	d	day index	δ	duration
E		e	time window LB	ϵ	lateness d.v.
F	feasible tuples (j,k) by week/day	f	time window UB	ζ	
G	feasible tuples (j,k) by hour	g		η	no. of nights for job
H	depot/base nodes	h	depot/base node parameter	θ	objective function weightings
I		i	job index	ι	
J	job nodes	j	job index	κ	
K	technicians	k	technician index	λ	nights away d.v.
L	skill levels	l	skill levels index	μ	night away indicator
M	big M	m	index for weighting of o.f.	ν	
N		n	number of technicians index	ξ	
O		o		o	–
P		p	pair d.v.	π	base node binary parameter
Q		q	technician qualification	ρ	split job d.v.
R		r	job skill requirements	σ	split job duration
S	skill domains	s	skill domain index	τ	travel time
T	hours in planning period	t	hours index	υ	

U		u	start time d.v.	ϕ	cost of overnight stay
V	all nodes in problem	v	technician skill levels	χ	
W	weeks in planning period	w	week index	ψ	cost of travel
X		x	leave/arrive d.v.	ω	work capacity
Y		y	jobs d.v.	Ω	hours/week
Z		z	two technicians d.v.		

4.4 Problem Formulation

4.4.1 Objective Function

There are several aspects of this problem that the RNLI would like to be included in an objective function, which can be seen in chapter 3. The objective function in its full state is as follows: Minimise the following Objective Function:

$$\theta_1 \psi \left(\sum_{(i,j,k,t) \in \mathcal{I}} \tau_{ij} x_{ijk}^t \right) + \tag{4.2a}$$

$$\theta_2 \phi \left(\sum_{(j,k,w,d) \in F'} \lambda_{jk}^{wd} \right) + \tag{4.2b}$$

$$\theta_3 \gamma \left(\sum_{j \in J} \epsilon_j \left(u_j + (1 - z_j)\delta_{j1} + z_j \delta_{j2} - f_j \right) \right) \tag{4.2c}$$

(4.2a) represents the cost of travel, by multiplying the variable for departure on arcs by its respective duration, and then by a scalar to calculate the associated cost. (4.2b) is the cost of nights away, which are simply summed and multiplied by the cost of a single night away. (4.2c) introduces the 'cost' of lateness, which is a lot harder to define. This term is non-linear due to the fact that we want to be able to penalise a job not

just by whether it is late, but by how much. To maintain a linear formulation we have replaced 4.2c with the simpler term $\theta_3\gamma\left(\sum_{j\in J}\epsilon_j\right)$, which simply counts the number of jobs that are late, regardless of the time. It would also be possible to add a job-based weighting here to prioritise the on time completion of particular jobs if required. The purpose of having two scalar values in each term of the objective function (e.g. $\Theta_1\psi$) is to allow these values to be set at different levels within the stakeholder use process. For example, the baseline cost of travel (ψ) would be set centrally by the RNLI, and the weighting apportioned to this part of the objective (Θ_1) would be set at a divisional level by end users depending on the specific requirements of that division. The weighting of the objective function is discussed further in section 6.3.

The other elements discussed as needing to be incorporated in to the objective are return visits and overtime. These have not been included at present because the formulation assumes that no overtime is used (so in reality overtime would occur when jobs overrun), and return visits, if they represent an inefficient use of resources, will be minimised implicitly through the formulation anyway.

4.4.2 Constraints

The following are the constraints for the full formulation of the Vehicle Routing Problem with Synchronisation, Time Windows ad Skill Levels. We will introduce and explain each constraint (or group of constraints, where relevant) individually. The full formulation of the problem can be found in section 4.4.3.

Number of technicians per job

$$\sum_{(i,k,t)\in\mathcal{I}_{(j)}} x^t_{ijk} \geq \alpha_j \qquad \forall j \in J \tag{4.3}$$

$$\sum_{(i,k,t)\in\mathcal{I}_{(j)}} x^t_{ijk} \leq \beta_j \qquad \forall j \in J \tag{4.4}$$

Constraints **??** ensure that, for each job, the number of technicians arriving to the job

is within the allowable range for that job, $[\alpha_j, \beta_j]$, where $\alpha_j, \beta_j \in [1, 2]$, therefore in the case where $\alpha_j = \beta_j$, these constraints will become binding on $\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk}$. These constraints are only defined for job nodes, J, as opposed to all nodes in the problem, $V = J \cup H$, because the parameters α_j, β_j are not defined for nodes in H. The matter of ensuring that the same number of technicians also leave each job is dealt with in the conservation of flow constraints, equation (4.5).

Conservation of flow

$$\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} = \sum_{\substack{i\in V, t\in T \mid \\ (j,i,k,t)\in\mathcal{I}}} x^t_{jik} \qquad \forall j \in V, k \in K \tag{4.5}$$

Equation (4.5) ensures the conservation of flow in and out of nodes. We note that, unlike equations (4.3) and (4.4) this is defined for all nodes in the network, rather than just job nodes, as we need to ensure that the number of technicians that arrive and depart each node is balanced at all nodes, including bases. The constraint to limit the number of technicians arriving and leaving each base is equation (4.6).

Returning to base

$$\sum_{\substack{j\in J, t\in T \mid \\ (j,k,t)\in G}} x^t_{h_k jk} = 1 \qquad \forall k \in K \tag{4.6}$$

Equation (4.6) ensures that each technician leaves their base node exactly once per week, and by extension, they also only return once (as equation (4.5) conserves flow in and out of all nodes). This is necessary because the constraints that ensure the correct timing for consecutive are only defined for nodes in J. As a result of this, placing no constraint on the flow in and out of base nodes could result in any given technician k being assigned to two jobs i and j at the same time, as returning to base between jobs would deactivate the constraint which ensures that travel to job j is started only once

technician k has completed job i.

If preprocessing (as described in section 4.3) is not used, an additional constraint is required:

$$\sum_{j \in J, t \in T} x^t_{h_k jk} = 0 \qquad \forall (h,k) \in H \times K \mid h_k \neq h \tag{4.7}$$

Equation (4.7) ensures that technicians can only depart from their own base node, as it is implied implicitly in preprocessing so removal of this requires this additional constraint.

Completion of arcs

$$x^t_{ijk} = \bar{x}^{t+\tau_{ij}}_{ijk} \qquad \forall (i,j,k,t) \in \mathcal{I} \tag{4.8}$$

Equation (4.8) ensures that, for a departure on an arc (i,j) at time t, denoted by variable x^t_{ijk}, the corresponding arrival at j happens at time $t + \tau_{ij}$ (represented by variable $x^{t+\tau_{ij}}_{ijk}$).

Jobs with two technicians

$$\sum_{(i,k,t) \in \mathcal{I}_{(j)}} x^t_{ijk} = z_j + 1 \qquad \forall j \in J \tag{4.9}$$

Equation (4.9) ensures that the total number of technicians arriving at a given job j is equal to the number of technicians assigned to that job. This constraint only works for cases in which each job may be performed by one or two technicians, as with the problem presented in this work. In other cases it would be necessary to adjust this constraint, as well as others, to allow for jobs to be completed by a larger number of technicians.

$$\sum_{(k,k')\in K_{(j)}} p_{jkk'} = 2z_j \qquad \forall j \in J \tag{4.10}$$

$$p_{jkk'} = p_{jk'k} \qquad \forall j \in J, k, k' \in K | k < k' \tag{4.11}$$

Equation (4.10) ensures that, if a job j is performed by a pair of technicians, there are exactly two corresponding $p_{jkk'}$ variables which take value one. Furthermore, equation (4.11) ensures the symmetry of this variable; the combination of these two constraints means that the pair assigned to this job is made up of only two technicians. It is necessary for $p_{jkk'}$ to be defined for both permutations of k and k' because of how it is used in other constraints; these constraints will always refer to the first k index, and as such it needs to be defined for all other feasible k' indices.

$$\sum_{k'\in K_{(j,k)}} p_{jkk'} \leq \sum_{(i,t)\in \mathcal{I}_{(j,k)}} x^t_{ijk} \qquad \forall j \in J, k \in K \tag{4.12}$$

Equation (4.12) means that a technician can only be assigned to a job as part of a pair if they travel to that job, at any time, and from any other node in the network.

Skill levels

$$\sum_{i\in J, t\in T} x^t_{ijk} \leq q_{jk}(1 - z_j) + \left(\sum_{k'\in K} q_{jkk'}\right) z_j \qquad \forall (j,k) \in \mathcal{K} \tag{4.13}$$

Equation (4.13) ensures that only technicians with the appropriate skills for a job can be assigned to that job. This constraint only needs to be included if preprocessing is **not** used in the definition of sets and decision variables (see section 4.3 for more information).

Time windows

$$u_j \geq e_j \qquad \forall j \in J \tag{4.14}$$

$$u_j + \delta_{j2} z_j + \delta_{j1}(1 - z_j) \leq f_j + c_j \epsilon_j \qquad \forall j \in J \tag{4.15}$$

Equation (4.14) requires the start time of each job to be at least the beginning of the time window for that job. Similarly, equation (4.15) means the start time, plus the duration of the job dependent on how many technicians complete the job, is at most the end of the time window. The term $c_j \epsilon_j$ transforms this constraint in to a soft one, in that time windows may be exceeded, but they will be subject to some penalty in the objective function. To make the time window binding for any (or all) jobs, it is simply required to set the relevant $c_j = 0$.

Start time

The equations below represent three different forms of the same constraint

$$u_j \geq t \bar{x}^t_{ijk} \qquad \forall (i,j,k,t) \in \bar{\mathcal{I}} \tag{4.16a}$$

$$u_j \geq t \left(\sum_{(i,k) \in \bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk} \right) \qquad \forall j \in J, t \in T \tag{4.16b}$$

$$u_j \geq t \left(\sum_{(i,k) \in \bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk} - z_j \right) \qquad \forall j \in J, t \in T \tag{4.16c}$$

In particular, (4.16a) is the correct constraint to ensure that the start time of a job is after the arrival of the technician (or technicians) that are completing the job, and equations (4.16b) and (4.16c) represent two alternative forms of this. The difference between them is how they handle the case in which two technicians are working on the same job (i.e. $z_j = 1$): if we had a guarantee that both technicians would arrive at exactly the same time, we could use equation (4.16c), as $z_j = 1$ would counter the fact

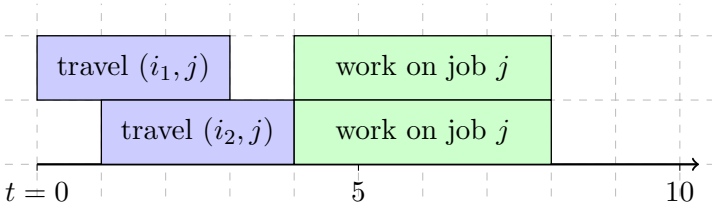
that $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} x^t_{ijk} = 2$, therefore maintaining a lower bound of t on u_j. However, this is not the case, which could lead to this equation becoming an arbitrary lower bound $u_j \geq 0$ (if $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} x^t_{ijk} = z_j = 1$). The alternative forms of equation (4.16a) are considered here because of the computational differences associated with such formulations; Wolsey [1998] shows that any formulation will always be stronger than its relaxation(s), and the comparative computational performance of these alternative forms against the original formulation is of interest, particularly in cases where the size of the problem is prohibitively large.

The possible cases of (4.16b) are outlined below:

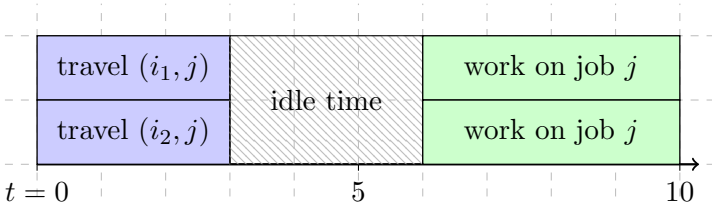
1. $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk} = 0$ (nobody arrives at job j at time t): $u_j \geq 0$
2. $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk} = 1$ (exactly one technician arrives at job j at time t): $u_j \geq t$
3. $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk} = 2$ (exactly two technicians arrive at job j at time t)
   (a) $t > \frac{|T|}{2}$: infeasible due to (4.15)
   (b) $t \leq \frac{|T|}{2}$: : $u_j \geq 2t$ - **this is higher than the required lower bound, which should be $u_j \geq t$. In this case, this constraint imposes too high a bound when compared to the original constraint.**

The possible forms of (4.16c) are outlined in the following cases:

1. $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk} = 0$ (nobody arrives at job j at time t)
   (a) $z_j = 0$: $u_j \geq 0$
   (b) $z_j = 1$: $u_j \geq -t \Rightarrow$ superseded by non-negativity of variables
2. $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk} = 1$ (exactly one technician arrives at job j at time t)
   (a) $z_j = 0$: $u_j \geq t$
   (b) $z_j = 1$: $u_j \geq 0$ - **not a sufficient lower bound, it should be $u_j \geq t$. In this case this constraint is weaker than the original constraint.**
3. $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk} = 2$ (exactly two technicians arrive at job j at time t)
   (a) $z_j = 0$: infeasible due to (4.9)



(a) Equation (4.16b), case 2: example of offset arrival times ($u_j \geq t$, $t = 3, 4$)



(b) Equation (4.16b), case 3(b): example of equal arrival times ($u_j \geq 2t$, $t = 3$)

Figure 4.1: Examples for cases of equation (4.16b)

(b) $z_j = 1$: $u_j \geq t$

Here we see that (4.16b) does not behave as desired for cases where $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk} = 2$ (case 3(b)), with some values of t resulting in an infeasible constraint, and others placing an unnecessarily high lower bound on u_j of $2t$, double the arrival time of the two technicians. From this, we see that, by the definition of Valid Inequalities given in Nemhauser and Wolsey [1999], (4.16b) is not a valid inequality for all solutions of this problem. However, if we consider a solution in which the arrival of multiple technicians is always offset by an hour (at least), this will never fall in to case 3, hence for this subset of equations, the inequality would therefore be valid. See fig. 4.1 for a diagram of cases 2 and 3(b) of this equation. If this constraint were used, an additional constraint to prohibit the simultaneous arrival of technicians in any location could be included to ensure the validity of (4.16b). Conversely, (4.16c) imposes to loose a lower bound on $\sum_{(i,k)\in\bar{\mathcal{I}}_{(j,t)}} \bar{x}^t_{ijk}$ in case 2(b), and hence we see that neither alternative form is correct in all cases. These alternative forms should be considered in cases where the full formulation is prohibitively large, especially in instances where reoptimisation happens regularly and as such computation times need to be kept to a minimum. In the case of the RNLI example, it was not necessary to use these alternative forms, but they have been included for the reader's reference.

Finish time

Similarly to the contraints in start time seen above, here we see three different forms of the constraint of job finish time

$$u_j + \delta_{j2} z_j + \delta_{j1}(1 - z_j) \leq M_1 + (t - M_1) x^t_{jhk} \qquad \forall (j, h, k, t) \in \mathcal{I} \tag{4.17a}$$

$$u_j + \delta_{j2} z_j + \delta_{j1}(1 - z_j) \leq M_1 + (t - M_1) \left(\sum_{\substack{i \in V, k \in K \mid \\ (j,i,k,t) \in \mathcal{I}}} x^t_{jik} \right) \qquad \forall j \in J, t \in T \tag{4.17b}$$

$$u_j + \delta_{j2} z_j + \delta_{j1}(1 - z_j) \leq M_1 + (t - M_1) \left(\sum_{\substack{i \in V, k \in K \mid \\ (j,i,k,t) \in \mathcal{I}}} x^t_{jik} - z_j \right) \qquad \forall j \in J, t \in T \tag{4.17c}$$

Where $M_1 = |T|$ for all combinations of (j, h, k, t). Equations (4.16a), (4.17b) and (4.17c) all ensure that a job is completed before the relevant technician(s) depart, with equations (4.17b) and (4.17c) representing alternative forms of (4.16a). Again, the difference between these alternative forms lies in how they handle jobs which are performed by two technicians.

The possible forms of (4.17b) are outlined in the following cases:

1. $\sum_{\substack{i \in V, k \in K \mid \\ (j,i,k,t) \in \mathcal{I}}} x^t_{jik} = 0$ (nobody leaves job j at time t):
   (a) $z_j = 0$: $u_j + \delta_{j1} \leq |T|$
   (b) $z_j = 1$: $u_j + \delta_{j2} \leq |T|$
2. $\sum_{\substack{i \in V, k \in K \mid \\ (j,i,k,t) \in \mathcal{I}}} x^t_{jik} = 1$ (exactly one technician leaves job j at time t):
   (a) $z_j = 0$: $u_j + \delta_{j1} \leq t$
   (b) $z_j = 1$: $u_j + \delta_{j2} \leq t$

3. $\sum_{\substack{i \in V, k \in K \mid \\ (j,i,k,t) \in \mathcal{I}}} x_{jik}^t = 2$ (exactly two technicians leave job j at time t):

   (a) $z_j = 0$: infeasible due to (4.9)

   (b) $z_j = 1$:

      i. $t < \frac{|T|+e_j+\delta_{j2}}{2}$: infeasible due to (4.14)

      ii. $t \geq \frac{|T|+e_j+\delta_{j2}}{2}$: $u_j + \delta_{j2} \leq 2t - |T|$

      $t \leq |T|$ (by definition) $\Rightarrow 2t - |T| \leq t \Rightarrow$ idle time detween job completion and technician departure of at least $|T| - t$ - **this places a tighter upper bound than required, it should be** $u_j + \delta_{j2} \leq t$

The possible forms of (4.17c) are outlined in the following cases:

1. $\sum_{\substack{i \in V, k \in K \mid \\ (j,i,k,t) \in \mathcal{I}}} x_{jik}^t = 0$ (nobody leaves job j at time t):

   (a) $z_j = 0$: $u_j + \delta_{j1} \leq |T|$

   (b) $z_j = 1$: $u_j + \delta_{j2} \leq 2|T| - t$

2. $\sum_{\substack{i \in V, k \in K \mid \\ (j,i,k,t) \in \mathcal{I}}} x_{jik}^t = 1$ (exactly one technician leaves job j at time t):

   (a) $z_j = 0$: $u_j + \delta_{j1} \leq t$

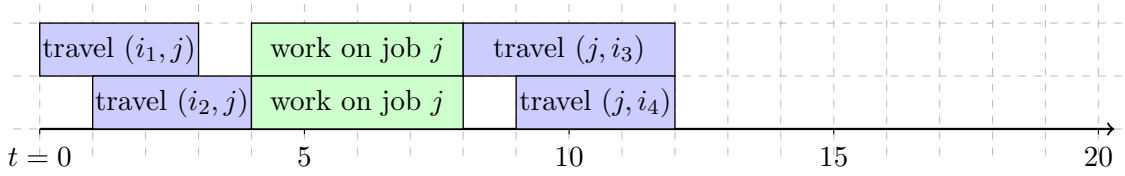
   (b) $z_j = 1$: $u_j + \delta_{j2} \leq |T|$ - **not a sufficient upper bound, should be** $u_j + \delta_{j2} \leq t$

3. $\sum_{\substack{i \in V, k \in K \mid \\ (j,i,k,t) \in \mathcal{I}}} x_{jik}^t = 2$ (exactly two technicians leave job j at time t):

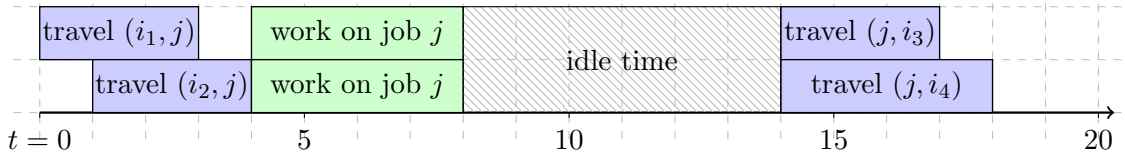
   (a) $z_j = 0$: infeasible due to (4.15)

   (b) $z_j = 1$: $u_j + \delta_{j2} \leq t$

Similarly to (4.16), we see that (4.17b) is not correct for cases in which $\sum_{\substack{i \in V, k \in K \mid \\ (j,i,k,t) \in \mathcal{I}}} x_{jik}^t = 2$, by either causing infeasibility or a tighter than required upper bound on the left hand side, from which we can conclude that this is not a valid constraint for the solution space. Similarly to the start time constraints, further constraining the feasible set to not include those solutions which fall in case 3(b)ii of (4.17b) would guarantee its validity. Two of the cases for equation (4.17b) can be seen in fig. 4.2, which illustrates the difference in



(a) Equation (4.17b), case 2: example of offset departure times ($u_j + \delta_{j2} \leq t$, $t = 8, 9$)



(b) Equation (4.17b), case 3(b)ii: example of equal departure times ($u_j + \delta_{j2} \leq 2t - |T|$, $t = 14$)

Figure 4.2: Examples for cases of equation (4.17b)

idle time for equal and offset departure times. As with the previous constraints on start time, these alternative forms should be considered if savings in computation time are required.

Tracking variables y sum to duration

$$\sum_{(k,t) \in G_{(j)}} y_{jk}^t = \delta_{j1}(1 - z_j) + 2\delta_{j2} z_j \qquad \forall j \in J \tag{4.18a}$$

$$\sum_{t \in G_{(j,k)}} y_{jk}^t = \delta_{j1} \left(\sum_{(i,t) \in \mathcal{I}_{(j,k)}} x_{ijk}^t - \sum_{k' \in K_{(j,k)}} p_{jkk'} \right) + \delta_{j2} \sum_{k' \in K_{(j,k)}} p_{jkk'} \qquad \forall j \in J, k \in K \tag{4.18b}$$

As the variable y_{jk}^t indicates that technician k is working on job j at a given time t, equation (4.18a) ensures that, for each job, the total number of hours worked is equal to the duration of the job (given that idle time in a location is not identified by this variable). Similarly, equation (4.18b) places the same constraint on the total of the y_{jk}^t variables, but in this case for individual technicians, rather than in total.

Summing equation (4.18b) across $k \in K$, we obtain the following equation:

$$\sum_{(k,t)\in G_{(j)}} y^t_{jk} = \delta_{j1}\left(\sum_{(i,k,t)\in\mathcal{I}_{(j)}} x^t_{ijk} - \sum_{(k,k')\in K_{(j)}} p_{jkk'}\right) + \delta_{j2}\sum_{(k,k')\in K_{(j)}} p_{jkk'} \quad \forall j \in J \tag{4.19}$$

Substituting equations (4.9) and (4.10) in to (4.19), the resulting equation is identical to (4.18a). From this, we know that (4.18a) is an alternative form of (4.18b), so it is not necessary to include both constraints in the same formulation.

Hours per technician variables σ sum to duration

$$\sum_{(k,w,d)\in F'_{(j)}} \sigma^{wd}_{jk} = \delta_{j1}(1 - z_j) + 2\delta_{j2}z_j \quad \forall j \in J \tag{4.20a}$$

$$\sum_{(w,d)\in F'_{(j,k)}} \sigma^{wd}_{jk} = \delta_{j1}\left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} - \sum_{k'\in K_{(j,k)}} p_{jkk'}\right) + \delta_{j2}\sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j \in J, k \in K \tag{4.20b}$$

Equations (4.20a) and (4.20b) perform the same task as equations (4.18a) and (4.18b), but in this case for the variable σ^{wd}_{jk}. As with the previous constraints, we can easily show that equation (4.20a) is an alternative form of (4.20b), so once again, it is not necessary to include both constraints in any one formulation.

It is also worth noting that, as the RHS of (4.18b) and (4.20b) (and (4.18a) and (4.20a)) are identical, and it would therefore be possible to replace either (4.18b) or (4.20b) with the following constraint:

$$\sum_{t\in G_{(j,k)}} y^t_{jk} = \sum_{(w,d)\in F'_{(j,k)}} \sigma^{wd}_{jk} \quad \forall j \in J, k \in K \tag{4.21}$$

or if the relaxed constraints (4.18a) and (4.20a) are being used, then either can be replaced with:

$$\sum_{(k,t)\in G_{(j)}} y_{jk}^{t} = \sum_{(k,w,d)\in F'_{(j)}} \sigma_{jk}^{wd} \qquad \forall j \in J \tag{4.22}$$

Hours per technician with two technicians

$$\sigma_{jk}^{wd} - \sigma_{jk'}^{wd} \leq \max_{i=k,k'}\{\omega_i^d\}(1 - p_{jkk'}) \quad \forall (j,k,w,d),(j,k',w,d) \in F' \mid \bar{q}_{j,k,k'} = 1 \tag{4.23}$$

$$\sigma_{jk'}^{wd} - \sigma_{jk}^{wd} \leq \max_{i=k,k'}\{\omega_i^d\}(1 - p_{jkk'}) \quad \forall (j,k,w,d),(j,k',w,d) \in F' \mid \bar{q}_{j,k,k'} = 1 \tag{4.24}$$

Equations (4.23) and (4.24) ensure that, for any pair of technicians that are working on the same job, they must work the same number of hours each day. For combinations of (j,k,k') for which $p_{jkk'} = 0$ (i.e. the pair of technicians k and k' are *not* working together on job j), these constraints simply limit the difference between the variables σ_{jk}^{wd} and $\sigma_{jk'}^{wd}$ to at most the length of that working day. Although there is some similarity between these constraints and (4.20b), these are still required, as (4.20b) does not consider the daily working amount of each technician, but instead considers their total working time spent on any given job.

Order of jobs

As with equations (4.16) and (4.17), here we consider two different forms of the following constraint, with differences on the right hand side of the inequality, and the handling of jobs that are performed by two technicians.

$$u_i + \delta_{i1}(1 - z_i) + \delta_{i2} z_i + \tau_{ij} \leq u_j + M_2\left(1 - x_{ijk}^{t}\right) \qquad \forall (i,j,k,t) \in \mathcal{I} \mid i,j \in J \tag{4.25a}$$

$$u_i + \delta_{i1}(1 - z_i) + \delta_{i2} z_i + \tau_{ij} \leq u_j + M_2 \left(1 - \left(\sum_{(k,t) \in \mathcal{I}_{(i,j)}} x_{ijk}^t\right)\right) \qquad \forall i, j \in J \tag{4.25b}$$

$$u_i + \delta_{i1}(1 - z_i) + \delta_{i2} z_i + \tau_{ij} \leq u_j + M_2 \left(1 - \left(\sum_{(k,t) \in \mathcal{I}_{(i,j)}} x_{ijk}^t - z_j\right)\right) \qquad \forall i, j \in J \tag{4.25c}$$

Where $M_2 = |T|$. Equations (4.25a) to (4.25c) require that the start time of a given job j is at least greater than the start time of its predecessor i, plus the time taken to perform job i, and the time to travel from i to j. As with previous constraints, we can see that equations (4.25b) and (4.25c) are alternative forms of (4.25a). To assess which one is most suitable for the given problem, each constraint is broken in to cases as follows:

For equation (4.25b):

1. $\sum_{(k,t) \in \mathcal{I}_{(i,j)}} x_{ijk}^t = 0$ (nobody travels from i to j, at any time)
   (a) $z_j = 0$: $u_i + \delta_{j1} + \tau_{ij} \leq u_j + |T| \Rightarrow$ weaker than non-negativity of variables
   (b) $z_j = 1$: $u_i + \delta_{j2} + \tau_{ij} \leq u_j + |T| \Rightarrow$ weaker than non-negativity of variables
2. $\sum_{(k,t) \in \mathcal{I}_{(i,j)}} x_{ijk}^t = 1$ (exactly one technician travels form i to j during the full planning period)
   (a) $z_j = 0$: $u_i + \delta_{j1} + \tau_{ij} \leq u_j$
   (b) $z_j = 1$: $u_i + \delta_{j2} + \tau_{ij} \leq u_j$
3. $\sum_{(k,t) \in \mathcal{I}_{(i,j)}} x_{ijk}^t = 2$ (exactly two technicians travel form i to j during the full planning period)
   (a) $z_j = 0$: infeasible due to (4.9)
   (b) $z_j = 1$: $u_i + \delta_{j2} + \tau_{ij} \leq u_j - |T|$ - infeasible due to (4.17) (as $u_j \leq |T|$)

and for (4.25c):

1. $\sum_{(k,t)\in\mathcal{I}_{(i,j)}} x^t_{ijk} = 0$ (nobody travels from i to j, at any time)

   (a) $z_j = 0$: $u_i + \delta_{j1} + \tau_{ij} \leq u_j + |T| \Rightarrow$ superseded by non-negativity of variables

   (b) $z_j = 1$: $u_i + \delta_{j2} + \tau_{ij} \leq u_j + 2|T| \Rightarrow$ superseded by non-negativity of variables

2. $\sum_{(k,t)\in\mathcal{I}_{(i,j)}} x^t_{ijk} = 1$ (exactly one technician travels form i to j during the full planning period)

   (a) $z_j = 0$: $u_i + \delta_{j1} + \tau_{ij} \leq u_j$

   (b) $z_j = 1$: $u_i + \delta_{j2} + \tau_{ij} \leq u_j + |T|$
   places a lower bound on u_j that is less than or equal to zero, as $u_i + \delta_{j1}(1 - z_j) + \delta_{j2} z_j + \tau_{ij} \leq |T|$. **Not as tight as the required constraint,** $u_i + \delta_{j1} + \tau_{ij} \leq u_j$

3. $\sum_{(k,t)\in\mathcal{I}_{(i,j)}} x^t_{ijk} = 2$ (exactly two technicians travel form i to j during the full planning period)

   (a) $z_j = 0$: infeasible due to (4.9)

   (b) $z_j = 1$: $u_i + \delta_{j1}(1 - z_j) + \delta_{j2} z_j + \tau_{ij} \leq u_j$

From this we see that (4.25b) is infeasible for all all cases where $\sum_{(k,t)\in\mathcal{I}_{(i,j)}} x^t_{ijk} = 2$, which means solutions in which two technicians travel together form one node to another would not be feasible. On the other hand, we see that (4.25c) does not impose as tight a constraint as desired for this problem when $\sum_{(k,t)\in\mathcal{I}_{(i,j)}} x^t_{ijk} = z_j = 1$. As the latter does not pose unnecessary infeasibility on some cases of the problem, we conclude that (4.25c) is the preferred alternative form in this case, if one is to be used at all. As with the constraints on start and finish time, these alternative forms should be considered in instances where computation time needs to be reduced. Due to the solution method and size of the examples solved in this thesis, these alternative forms were not required in this case but are included for the reader's reference.

Daily working capacity

$$\sum_{\substack{(i,j,t)\in\mathcal{I}_{(k)}\\ t\in T_{wd}}} \tau_{ij}x^t_{ijk} + \sum_{(j\in F'_{(k,w,d)}} \sigma^{wd}_{jk} + \sum_{\substack{j\in J, t\in T_{wd}\mid\\ (j,k,t)\in G}} \tau_{jh_k}x^t_{jh_kk} \leq \omega^d_k \quad \forall k\in K, w\in W, d\in D \tag{4.26}$$

Equation (4.26) constrains the daily working and travelling time of each technician to be at most their working capacity for that day. At present, we assume that any travelling to and from the base not at the beginning or end of the week is not included in this time (i.e., for technicians not staying away from home, their time to travel home each night is not included in their daily working hours). If it were to be included, the duration of a job would also be dependant on the technician, as well as the number of technicians performing the job. As such, complications arise in dealing with a job that is performed by two technicians, one of whom is required to stay away from home, and the other is not, as they will have different completion times for the same job. It is for this reason that travel time to or from their base node has been excluded from the daily working hours.

Nights away bounds

$$\sum_{(w,d)\in F'_{(j,k)}} \lambda^{wd}_{jk} \geq \eta^1_{j1}\left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} - \sum_{k'\in K_{(j,k)}} p_{jkk'}\right) + \eta^1_{j2}\sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j\in J, k\in K|\mu_{jk}=1 \tag{4.27}$$

$$\sum_{(w,d)\in F'_{(j,k)}} \lambda^{wd}_{jk} \leq \eta^7_{j1}\left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} - \sum_{k'\in K_{(j,k)}} p_{jkk'}\right) + \eta^7_{j2}\sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j\in J, k\in K|\mu_{jk}=1 \tag{4.28}$$

Equations (4.27) and (4.28) constrain the total number of nights spent away at a particular job to be between the minimum and maximum number of nights over which the job can be split. The cases for these constraints are as follows:

1. $\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} = 0$ (technician k does not perform job j)
   (a) $\sum_{k'\in K_{(j,k)}} p_{jkk'} = 0$: $\sum_{(w,d)\in F'_{(j,k)}} \lambda^{wd}_{jk} \geq 0$, $\sum_{(w,d)\in F'_{(j,k)}} \lambda^{wd}_{jk} \leq 0$
   (b) $\sum_{k'\in K_{(j,k)}} p_{jkk'} = 1$: infeasible due to (4.12)
2. $\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} = 1$ (technician k performs job j)
   (a) $\sum_{k'\in K_{(j,k)}} p_{jkk'} = 0$: $\sum_{(w,d)\in F'_{(j,k)}} \lambda^{wd}_{jk} \geq \eta^1_{j1}$, $\sum_{(w,d)\in F'_{(j,k)}} \lambda^{wd}_{jk} \leq \eta^7_{j1}$
   (b) $\sum_{k'\in K_{(j,k)}} p_{jkk'} = 1$: $\sum_{(w,d)\in F'_{(j,k)}} \lambda^{wd}_{jk} \geq \eta^1_{j2}$, $\sum_{(w,d)\in F'_{(j,k)}} \lambda^{wd}_{jk} \leq \eta^7_{j2}$

As these constraints (and also the variables λ^{wd}_{jk}) are only defined for j, k such that $\mu_{jk} = 1$ (i.e. technician k must stay away from home if working on job j), no consideration of this is included in the constraints themselves.

Job split bounds

$$\sum_{(w,d)\in F'_{(j,k)}} \rho^{wd}_{jk} \geq \eta^1_{j1} \left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} - \sum_{k'\in K_{(j,k)}} p_{jkk'} \right) + \eta^1_{j2} \sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j \in J, k \in K \tag{4.29}$$

$$\sum_{(w,d)\in F'_{(j,k)}} \rho^{wd}_{jk} \leq \eta^7_{j1} \left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} - \sum_{k'\in K_{(j,k)}} p_{jkk'} \right) + \eta^7_{j2} \sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j \in J, k \in K \tag{4.30}$$

Similarly to equations (4.27) and (4.28), these constraints ensure that the number of nights over which a job is split (regardless of whether the technician stays away from home) is within the range $(\eta^1_{jn}, \eta^7_{jn})$. These constraints are only required if the variables ρ^{wd}_{jk} are included, and in such a case can be removed without affecting any other constraints in the formulation. The cases of these constraints are as follows:

1. $\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} = 0$ (technician k does not perform job j)
   (a) $\sum_{k'\in K_{(j,k)}} p_{jkk'} = 0$: $\sum_{(w,d)\in F'_{(j,k)}} \rho^{wd}_{jk} \geq 0$, $\sum_{(w,d)\in F'_{(j,k)}} \rho^{wd}_{jk} \leq 0$
   (b) $\sum_{k'\in K_{(j,k)}} p_{jkk'} = 1$: infeasible due to (4.12)

2. $\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x_{ijk}^t = 1$ (technician k performs job j)

   (a) $\sum_{k'\in K_{(j,k)}} p_{jkk'} = 0$: $\sum_{(w,d)\in F'_{(j,k)}} \rho_{jk}^{wd} \geq \eta_{j1}^1$, $\sum_{(w,d)\in F'_{(j,k)}} \rho_{jk}^{wd} \leq \eta_{j1}^7$

   (b) $\sum_{k'\in K_{(j,k)}} p_{jkk'} = 1$: $\sum_{(w,d)\in F'_{(j,k)}} \rho_{jk}^{wd} \geq \eta_{j2}^1$, $\sum_{(w,d)\in F'_{(j,k)}} \rho_{jk}^{wd} \leq \eta_{j2}^7$

equation (4.29)-equation (4.30): the total number of nights that each job is split over is equal to the number of nights given by η_{jn}^t

Consecutive variables

$$\sum_{t'=t}^{t+\bar{\delta}_j} y_{jk}^{t'} \geq \bar{\delta}_j \sum_{i\in\bar{\mathcal{I}}_{(j,k,t)}} \bar{x}_{ijk}^t - \left(\bar{\delta}_j - \delta_{j2}\right) z_j \qquad \forall (j,k,t) \in G \tag{4.31}$$

$$\sum_{d=d_t}^{d_t+\bar{\eta}_j^t} \sigma_{jk}^{w_t d} \geq \bar{\delta}_j \sum_{i\in\bar{\mathcal{I}}_{(j,k,t)}} \bar{x}_{ijk}^t - \left(\bar{\delta}_j - \delta_{j2}\right) z_j \qquad \forall (j,k,t) \in G \tag{4.32}$$

$$\sum_{d=d_t}^{d_t+\bar{\eta}_j^t} \rho_{jk}^{w_t d} \geq \bar{\eta}_j^t \sum_{i\in\bar{\mathcal{I}}_{(j,k,t)}} \bar{x}_{ijk}^t - \left(\bar{\eta}_j^t - \eta_{j2}^t\right) z_j \qquad \forall (j,k,t) \in G \tag{4.33}$$

Equations (4.31) to (4.33) ensure that, if a technician k arrives at job j at time t, then the subsequent y_{jk}^t, σ_{jk}^{wd} and ρ_{jk}^{wd} variables are equal to one for the duration of the job. To check the accurancy of these constraints, we again consider the following cases:

1. $\sum_{i\in\bar{\mathcal{I}}_{(j,k,t)}} \bar{x}_{ijk}^t = 0$ (technician k does not arrive at job j at time t)

   (a) $z_j = 0$: $\sum_{t'=t}^{t+\bar{\delta}_j} y_{jk}^{t'} \geq 0$
   $\sum_{d=d_t}^{d_t+\bar{\eta}_j^t} \sigma_{jk}^{w_t d} \geq 0$
   $\sum_{d=d_t}^{d_t+\bar{\eta}_j^t} \rho_{jk}^{w_t d} \geq 0$

   (b) $z_j = 1$: $\sum_{t'=t}^{t+\bar{\delta}_j} y_{jk}^{t'} \geq \bar{\delta}_j - \delta_{j2}$
   $\sum_{d=d_t}^{d_t+\bar{\eta}_j^t} \sigma_{jk}^{w_t d} \geq \bar{\delta}_j - \delta_{j2}$
   $\sum_{d=d_t}^{d_t+\bar{\eta}_j^t} \rho_{jk}^{w_t d} \geq \bar{\eta}_j^t - \eta_{j2}^t \Rightarrow$ as we know that $\delta_{j2} \leq \bar{\delta}_j$ and $\eta_{j2}^t \leq \bar{\eta}_j^t$, these are either a negative or zero lower bound on the LHS

2. $\sum_{i\in\bar{\mathcal{I}}_{(j,k,t)}} \bar{x}_{ijk}^t = 1$ (technician k arrives at job j at time t)

(a) $z_j = 0$: $\sum_{t'=t}^{t+\bar{\delta}_j} y_{jk}^{t'} \geq \bar{\delta}_j = \delta_{j1}$
$\sum_{d=d_t}^{d_t+\bar{\eta}_j^t} \sigma_{jk}^{w_t d} \geq \bar{\delta}_j = \delta_{j1}$
$\sum_{d=d_t}^{d_t+\bar{\eta}_j^t} \rho_{jk}^{w_t d} \geq \bar{\delta}_j = \delta_{j1}$

(b) $z_j = 1$:

i. $\alpha_j = 1$: $\sum_{t'=t}^{t+\delta_{j1}} y_{jk}^{t'} \geq \delta_{j2}$
$\sum_{d=d_t}^{d_t+\eta_{j1}^t} \sigma_{jk}^{w_t d} \geq \delta_{j2}$
$\sum_{d=d_t}^{d_t+\eta_{j1}^t} \rho_{jk}^{w_t d} \geq \delta_{j2} \Rightarrow$ these are weaker than the required constraints

ii. $\alpha_j = 2$: $\sum_{t'=t}^{t+\delta_{j2}} y_{jk}^{t'} \geq \delta_{j2}$
$\sum_{d=d_t}^{d_t+\eta_{j2}^t} \sigma_{jk}^{w_t d} \geq \delta_{j2}$
$\sum_{d=d_t}^{d_t+\eta_{j2}^t} \rho_{jk}^{w_t d} \geq \delta_{j2}$

Here we further consider case 2(b)i: if $\alpha_j = 1$ (i.e. job j has a minimum number of technicians 1), then $\bar{\delta}_j = \delta_{j1} \geq \delta_{j2}$ and $\bar{\eta}_j^t = \eta_{j1}^t \geq \eta_{j2}^t$, as the job duration (and therefore the number of nights over which it is split) is assumed to go down when it is performed by two instead of one technicians. In this case, these constraints will provide a weaker that required bound, as they are summing the variables across more days than the job takes to complete. Ideally we need to develop a linear way of making the sum across technician hours dependent on the number of technicians completing the job.

4.4.3 Full Formulation

As discussed previously, there are some decision variables which are not essential to the formulation, but can instead be included or excluded based on the requirements of the objective function or any additional constraints which are added in to the problem. Those which must be included in any basic formulation are x_{ijk}^t, $\bar{x}_{ijk}^t$, z_j, $p_{jkk'}$, u_j, ϵ_j and σ_{jk}^{wd}. The full formulation presented directly below includes only these variables, and following that we present the constraints that should be added for any of the additional optional variables. Where alternative forms have been proposed for some constraints, the original and correct constraints are presented in the full formulation, and those which can be swapped for alternative forms will be highlighted after the full formulation has been presented.

Minimise the Objective Function:

$$\theta_1 \psi \left(\sum_{(i,j,k,t) \in \mathcal{I}} \tau_{ij} x_{ijk}^t \right) + \tag{4.34a}$$

$$\theta_2 \phi \left(\sum_{(j,k,w,d) \in F'} \lambda_{jk}^{wd} \right) + \tag{4.34b}$$

$$\theta_3 \gamma \left(\sum_{j \in J} \epsilon_j \left(u_j + (1 - z_j)\delta_{j1} + z_j \delta_{j2} - f_j \right) \right) \tag{4.34c}$$

Subject to the following constraints:

$$\sum_{(i,k,t) \in \mathcal{I}_{(j)}} x_{ijk}^t \geq \alpha_j \qquad \forall j \in J \tag{4.35}$$

$$\sum_{(i,k,t) \in \mathcal{I}_{(j)}} x_{ijk}^t \leq \beta_j \qquad \forall j \in J \tag{4.36}$$

$$\sum_{(i,t) \in \mathcal{I}_{(j,k)}} x_{ijk}^t = \sum_{\substack{h \in V, t \in T \mid \\ (j,i,k,t) \in \mathcal{I}}} x_{jik}^t \qquad \forall j \in V, k \in K \tag{4.37}$$

$$\sum_{\substack{j \in J, t \in T_w \mid \\ (j,k,t) \in G}} x_{h_k jk}^t = 1 \qquad \forall k \in K, w \in W \tag{4.38}$$

$$x_{ijk}^t = \bar{x}_{ijk}^{t+\tau_{ij}} \qquad \forall (i,j,k,t) \in \mathcal{I} \tag{4.39}$$

$$\sum_{(i,k,t) \in \mathcal{I}_{(j)}} x_{ijk}^t = z_j + 1 \qquad \forall j \in J \tag{4.40}$$

$$\sum_{(k,k') \in K_{(j)}} p_{jkk'} = 2 z_j \qquad \forall j \in J \tag{4.41}$$

$$p_{jkk'} = p_{jk'k} \qquad \forall j \in J, k, k' \in K | k < k' \tag{4.42}$$

$$\sum_{k' \in K_{(j,k)}} p_{jkk'} \leq \sum_{(i,t) \in \mathcal{I}_{(j,k)}} x_{ijk}^t \qquad \forall j \in J, k \in K \tag{4.43}$$

$$u_j \geq e_j \qquad \forall j \in J \tag{4.44}$$

$$u_j + \delta_{j2} z_j + \delta_{j1}(1 - z_j) \leq f_j + c_j \epsilon_j \qquad \forall j \in J \tag{4.45}$$

$$u_j \geq t\bar{x}^t_{ijk} \qquad \forall (i,j,k,t) \in \bar{\mathcal{I}} \qquad (4.46)$$

$$u_j + \delta_{j2} z_j + \delta_{j1}(1 - z_j) \leq M_1 + (t - M_1) x^t_{jik} \qquad \forall (i,j,k,t) \in \mathcal{I} \qquad (4.47)$$

$$\sum_{(w,d) \in F'_{(j,k)}} \sigma^{wd}_{jk} = \delta_{j1} \left(\sum_{(i,t) \in \mathcal{I}_{(j,k)}} x^t_{ijk} - \sum_{k' \in K_{(j,k)}} p_{jkk'} \right) + \delta_{j2} \sum_{k' \in K_{(j,k)}} p_{jkk'} \quad \forall j \in J, k \in K \qquad (4.48)$$

$$\sigma^{wd}_{jk} - \sigma^{wd}_{jk'} \leq \max_{i=k,k'} \{\omega^d_i\}(1 - p_{jkk'}) \qquad \forall (j,k,w,d), (j,k',w,d) \in F' \mid \bar{q}_{j,k,k'} = 1 \qquad (4.49)$$

$$\sigma^{wd}_{jk'} - \sigma^{wd}_{jk} \leq \max_{i=k,k'} \{\omega^d_i\}(1 - p_{jkk'}) \qquad \forall (j,k,w,d), (j,k',w,d) \in F' \mid \bar{q}_{j,k,k'} = 1 \qquad (4.50)$$

$$u_i + \delta_{j1}(1 - z_j) + \delta_{j2} z_j + \tau_{ij} \leq u_j + M_2 \left(1 - x^t_{ijk}\right) \qquad \forall (i,j,k,t) \in \mathcal{I} \qquad (4.51)$$

$$\sum_{\substack{(i,j,t) \in \mathcal{I}_{(k)} \\ t \in T_{wd}}} \tau_{ij} x^t_{ijk} + \sum_{(j \in F'_{(k,w,d)}} \sigma^{wd}_{jk} + \sum_{\substack{j \in J, t \in T_{wd} \mid \\ (j,k,t) \in G}} \tau_{jh_k} x^t_{jh_k k} \leq \omega^d_k \quad \forall k \in K, w \in W, d \in D \qquad (4.52)$$

$$\sum_{d=d_t}^{d_t + \bar{\eta}^t_j} \sigma^{w_t d}_{jk} \geq \bar{\delta}_j \sum_{i \in \bar{\mathcal{I}}_{(j,k,t)}} \bar{x}^t_{ijk} - \left(\bar{\delta}_j - \delta_{j2}\right) z_j \qquad \forall (j,k,t) \in G \qquad (4.53)$$

If the variable y^t_{jk} is to be used, the following constraints should be included:

$$\sum_{t\in G_{(j,k)}} y_{jk}^{t} = \delta_{j1}\left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x_{ijk}^{t} - \sum_{k'\in K_{(j,k)}} p_{jkk'}\right) + \delta_{j2}\sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j\in J, k\in K \tag{4.54}$$

$$\sum_{t'=t}^{t+\bar{\delta}_j} y_{jk}^{t'} \geq \bar{\delta}_j \sum_{i\in\bar{\mathcal{I}}_{(j,k,t)}} \bar{x}_{ijk}^{t} - \left(\bar{\delta}_j - \delta_{j2}\right) z_j \quad \forall (j,k,t)\in G \tag{4.55}$$

If the variable λ_{jk}^{wd} is to be used, the following constraints should be included:

$$\sum_{(w,d)\in F'_{(j,k)}} \lambda_{jk}^{wd} \geq \eta_{j1}^{1}\left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x_{ijk}^{t} - \sum_{k'\in K_{(j,k)}} p_{jkk'}\right) + \eta_{j2}^{1}\sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j\in J, k\in K \,|\mu_{jk}=1 \tag{4.56}$$

$$\sum_{(w,d)\in F'_{(j,k)}} \lambda_{jk}^{wd} \leq \eta_{j1}^{7}\left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x_{ijk}^{t} - \sum_{k'\in K_{(j,k)}} p_{jkk'}\right) + \eta_{j2}^{7}\sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j\in J, k\in K \,|\mu_{jk}=1 \tag{4.57}$$

If the variable ρ_{jk}^{wd} is to be used, the following constraints should be included:

$$\sum_{(w,d)\in F'_{(j,k)}} \rho_{jk}^{wd} \geq \eta_{j1}^{1}\left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x_{ijk}^{t} - \sum_{k'\in K_{(j,k)}} p_{jkk'}\right) + \eta_{j2}^{1}\sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j\in J, k\in K \tag{4.58}$$

$$\sum_{(w,d)\in F'_{(j,k)}} \rho_{jk}^{wd} \leq \eta_{j1}^{7}\left(\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x_{ijk}^{t} - \sum_{k'\in K_{(j,k)}} p_{jkk'}\right) + \eta_{j2}^{7}\sum_{k'\in K_{(j,k)}} p_{jkk'} \quad \forall j\in J, k\in K \tag{4.59}$$

$$\sum_{d=d_t}^{d_t+\bar{\eta}_j^t} \rho_{jk}^{w_t d} \geq \bar{\eta}_j^t \sum_{i \in \bar{\mathcal{I}}_{(j,k,t)}} \bar{x}_{ijk}^t - \left(\bar{\eta}_j^t - \eta_{j2}^t\right) z_j \qquad \forall (j,k,t) \in G \tag{4.60}$$

If preprocessing described in section 4.3 is **not** used, the following constraints should be included:

$$\sum_{i \in J, t \in T} x_{ijk}^t \leq q_{jk}(1 - z_j) + \left(\sum_{k' \in K} q_{jkk'} \right) z_j \qquad \forall (j,k) \in \mathcal{K} \tag{4.61}$$

$$\sum_{j \in J, t \in T} x_{h_k jk}^t = 0 \qquad \forall (h,k) \in H \times K \mid h_k \neq h \tag{4.62}$$

In addition, the alternative forms given in table 4.9 can be used if required.

Table 4.9: Alternative Constraint Forms

Constraint	**Alternative Form(s)**
(4.46)	(4.16b), (4.16c)
(4.47)	(4.17b), (4.17c)
(4.48)	(4.20a)
(4.51)	(4.25c)
(4.54)	(4.18a)

In the case of the RNLI problem, the variables y_{jk}^t and ρ_{jk}^{wd} are not required as we do not need to consider the location of technicians at a given time t (indicated by y_{jk}^t), or the number of nights over which jobs are split (ρ_{jk}^{wd}). However, the variable λ_{jk}^{wd} is necessary to satisfy the requirement that the number of nights spent away from home is included as part of the objective function. Therefore, the variables λ_{jk}^{wd} and equations (4.56) and (4.57) are included in the final formulation. The preprocessing of decision variables and sets described in section 4.3 was not used in the experiments presented in this thesis, as it was decided removing these restrictions would allow CPLEX to make use of it's own preprocessing techniques without additional restrictions imposed from the beginning. As such constraints (4.61) and (4.62) are required in the final formulation. In a case where

computation time needs to be further reduced or state of the art solvers such as CPLEX are not available, it is advisable to carry out testing of this preprocessing, but that was not needed in this case.

Unfortunately, not all of the requirements outlined in chapter 3 have been included in this formulation. Including them all explicitly would have created an even more complex formulation, so the decision was taken to exclude some in order to create a formulation which could still be solved in a reasonable time using appropriate techniques. The requirements stated in chapter 3 that are not included explicitly in this formulation are daily driving limit, training and precedence. The omission of the first two of these does not have to great an impact on the usability of the formulation within the RNLI: as stated previously, the daily driving limit is so high that it is very rare that anyone would reach this limit, as it would not leave any significant time to do any maintenance work so it is only in rare cases that this might be an issue. With regards to training, these can be included as jobs for which only those needing the training have the required skills, and as such no additional constraints are needed to leave empty time in a schedule in the event that a technician requires training. The final omitted requirement is precedence, which was chosen to be excluded as it was deemed less important than the other requirements discussed with the problem stakeholders, and not something that occurs frequently.

4.5 Formulation Reduction

Through the initial stages of computational testing with this formulation it became apparent that even with some requirements removed the formulation still could not be solved in reasonable time, or in some cases could not be solved at all. Through testing it was discovered that the removal of (4.53) from the formulation resulted in a problem that could be solved with much greater success: when testing ten instances of a small problem, feasibility was restored to those instances which had previously been infeasible, and some which had previously reached the time limit imposed on the solvers (one hour) were solved. Removal of this constraint would create infeasibility in the σ_{jk}^{wd} variables, so the decision was taken to remove these variables from the formulation, and all associated

constraints. As a result, (4.48), (4.49), (4.50), (4.52), (4.53) were all removed from the formulation along with the σ_{jk}^{wd} variables. In addition constraint (4.39), which is a direct equality between variables x_{ijk}^{t} and $\bar{x}_{ijk}^{t+\tau_{ij}}$ can be removed by substitution. Based on the fact that the $\bar{x}$ variables are used less in the full formulation, these were chosen to be removed, although this is just a matter of convenience and the same result would be achieved in either case. Therefore, altering the indices of $\bar{x}$ gives us the following equivalence for substitution:

$$\bar{x}_{ijk}^{t} = x_{ijk}^{t-\tau_{ij}} \tag{4.63}$$

This can be substituted in to equation (4.46) to achieve the following new constraint:

$$u_j \geq t x_{ijk}^{t-\tau_{ij}} \qquad \forall (i,j,k,t) \in \mathcal{I} \tag{4.64}$$

$$\tag{4.65}$$

These represent all the changes that were made to the formulation before the final stages of testing and implementation.

Chapter 5

Methodology

In this chapter we present the full methodology used to solve the problem formulated in chapter 4. This begins with Lagrangian relaxation and the subgradient method, including testing of which constraints to dualise and algorithm performance. This is followed by the local search heuristic, for which the subgradient method provides the initial solution, and which forms the second part of the overall matheuristic approach. We also present an approach which can be used to create a hierarchical problem structure, but this is purely theoretical and does not form part of the computational testing carried out. This approach would be useful in cases where the full problem is too large to be solved in reasonable time by the matheuristic method presented in this work, and therefore a hierarchical structure could break the full problem down in to subproblems of a suitable size for the matheuristic method which can then be solved individually, for example splitting the planning horizon or creating further geographical divisions.

5.1 Lagrangian Relaxation

Lagrangian relaxation is commonly used method in Operational Research for obtaining a lower bound of a combinatorial optimisation problem. In the case of linear or integer programming, this can be achieved by assigning multipliers to a selection of the problem constraints and dualising them in the objective function, as outlined by Held et al. [1974].

We consider an integer programming problem in standard form:

$$\begin{aligned} \min \ & c^T x && (5.1a)\\ \text{s.t.} \ & Ax \geq b && (5.1b)\\ & Dx \geq e && (5.1c)\\ & x \in \mathbb{Z} && (5.1d) \end{aligned}$$

where $Dx \geq e$ are the constraints to be dualised. Taking a vector of multipliers λ, the problem is reformulated in to the Lagrangian relaxation

$$\begin{aligned} \min \ & c^T x + \lambda(e - Dx) && (5.2a)\\ \text{s.t.} \ & Ax \geq b && (5.2b)\\ & x \in \mathbb{Z} && (5.2c) \end{aligned}$$

The associated Lagrangian Dual is therefore

$$\max_{\lambda \geq 0} \left\{ \min_{x \in \mathbb{Z}} c^T x + \lambda(e - Dx) \mid Ax \geq b \right\} \tag{5.3}$$

It is easy to prove that a solution to this problem, known as the Lagrangian Lower Bound Problem (LLBP), provides a lower bound to the original problem: let $\bar{x}$ be a feasible solution to the full problem (5.1). We know that $e - D\bar{x} < 0$ due to (5.1c), so it follows that:

$$c^T \bar{x} + \lambda(e - Dx) \leq c^T \bar{x}$$

and hence for any solution $\bar{x}$, the objective value of the LLBP is a lower bound to the objective value of the original problem.

The performance of any algorithm to solve this LLBP is hugely dependent on the selection of constraints to be dualised, as the removal of each set of constraints has a different impact of the problem structure, and therefore how quickly it can be solved with traditional solvers. In order to determine which constraints to dualise for the given problem, each set of constraints were removed in turn, and the problem solved on the same eight instances in order to compare solution time. The full results of these tests and introduction of the instances used can be found in section 7.1.1. Even though these were only tested on a small instance of the problem, there were two sets of constraints which stood out as having the greatest reduction on computation time:

$$u_j \geq t x_{ijk}^{t-\tau_{ij}} \qquad \forall (i,j,k,t) \in \bar{\mathcal{I}} \tag{5.4}$$

$$u_j + \delta_{j2} z_j + \delta_{j1}(1 - z_j) \leq M_1 + (t - M_1) x_{jhk}^{t} \qquad \forall (j,h,k,t) \in \mathcal{I} \tag{5.5}$$

These ensure that the arrival and departure times of a job respectively happen before and after the job has been completed. Further testing of these revealed that the best solution times on larger instances of the problem result from both constraints being dualised simultaneously, the full results of which can be seen in section 7.1.2.

Combining these two, the new objective function of the Lagrangian dual problem is

$$Z + \lambda(t x_{ijk}^{t-\tau_{ij}} - u_j) + \mu(u_j + \delta_{j2} z_j + \delta_{j1}(1 - z_j) - (M_1 + (t - M_1) x_{jhk}^{t})) \tag{5.6}$$

Where Z is the original objective function. The formulation is as presented in chapter 4, but without the constraints (5.4) and (5.5); the full formulation will not be presented again here due to its size.

5.2 Subgradient Method

The subgradient method is one frequently used for the solution of the Lagrangian Lower Bound Problem (LLBP), outlined in section 5.1. Below is the subgradient method as stated by Beasley [1996]:

1. Let π be a user defined 'step-size' parameter. Recommended value $0 \leq \pi \leq 2$.
2. Initialise Z_{UB}, for example using some heuristic for the original problem.
3. Initialise $Z_{LB} = -\infty$
4. Set λ_i, the initial values for the Lagrange multipliers
5. Solve LLBP with the current set of multipliers λ_i, to achieve solution X_j, with objective value Z^*
6. Calculate subgradients for the current solution:

$$G_i = b_i - \sum_{j=1}^{n} a_{ij} X_j, \quad i = 1, \ldots m$$

7. Calculate step size δ for the current solution:

$$\delta = \frac{\pi(Z_{UB} - Z_{LB})}{\sum_{i=1}^{m} G_i^2}$$

8. Update λ_i for the current solution:

$$\lambda_i = \max(0, \lambda_i + \delta G_i), \quad i = 1, \ldots, m$$

9. Update $Z_{LB} = \max(Z_{LB}, Z^*)$
10. Repeat from step 5 until termination condition(s) are met

Termination conditions:

- $Z_{UB} = Z_{LB}$
- $\pi \leq 0.005$

Beasley [1996] also made a recommendation based on their observations: to avoid getting stuck in a local minima, it is recommended to update $\pi = \frac{\pi}{2}$ if Z_{LB} has not improved in a fixed number of iterations N ($N = 30$ recommended).

One drawback to this particular variation of the subgradient method is the need for an upper bound on Z when calculating the lagrangian multipliers: such a method is likely being used because of the complexity of solving the problem in full, and as such even finding an upper bound to the objective value may be computationally demanding. For the problem described in this work, the upper bound will be determined by setting the CPLEX MIP solution limit to 1 when solving the full problem, in other words taking the first integer solution found. The performance of this method will then be compared with other step sizes in order to determine which is best suited to the given problem. The full list of step sizes to be considered will be outlined in the next section.

5.2.1 Step Size Selection

In the literature relating to subgradient methods, there are a number of different step sizes that are used in order to achieve the best results for a given problem. In this section I will outline the necessary conditions of a step size calculation to guarantee convergence, as well as the the different step sizes that were considered for this particular problem, and evaluate their suitability from a theoretical standpoint. The full results produced by the different step sizes, and an in depth comparison between them, is presented in section 7.2.

Brännlund [1995] outline two conditions on the calculation of the step size h_k, either of which is sufficient to guarantee convergence of the subgradient method. These conditions are:

1. $h_k = \frac{\gamma_k(f(x_k)-f^*)}{||g_k||^2}$, where $0 < \delta \leq \gamma_k \leq 2 - \delta$ and f^* is the known optimal value
2. $h_k > 0, \lim_{k\to\infty} h_k = 0$ and $\sum_{h=0}^{\infty} h_k = \infty$

This does not guarantee that step sizes which do not satisfy either of these conditions will not converge when used with the subgradient method on a particular problem, just

that it is not guaranteed to converge in all cases.

The first step size considered is that outlined by Beasley [1996], and introduced in section 5.2. The step size δ is calculated as follows:

$$\delta = \frac{\pi(Z_{UB} - Z_{LB})}{\sum_{i=1}^{m} G_i^2} \tag{5.7}$$

where π is a fixed step size parameter such that $0 < \pi < 2$, Z_{UB} is a previously calculated upper bound to the lagrangian dual, and G_i are the subgradient values. This stepsize satisfies the first of the conditions stated by Brännlund, so it is guaranteed to converge with an appropriate selection of π.One drawback of this step size which can be observed immediately however is the need to calculate Z_{UB}. This can be calculated in a number of different ways, for example through the use of a simple heuristic, or terminating after the first integer solution from a traditional sover such as CPLEX, but for large problems this could become a very costly procedure in terms of time. Furthermore, the challenge at hand is already to find a suitable solution method for the given problem, and as such requiring another method for finding an upper bound within this further complicates the matter. Nevertheless, computational experiments for this step size were carried out in order to establish whether the time taken to find an upper bound is offset by the performance of this step size within the subgradient method.

Further to this, a number of other calculations of step size were considered and tested for comparative purposes. In order to assess whether it is necessary to use a step size which requires the calculation of an upper bound, these step sizes are calculated independently of the objective function value, and instead depend solely on the iteration number in the subgradient method. The step sizes tested were:

$$\delta = \frac{1}{i+1} \tag{5.8}$$

$$\delta = \frac{1}{\sqrt{i+1}} \tag{5.9}$$

$$\delta = \rho^i \tag{5.10}$$

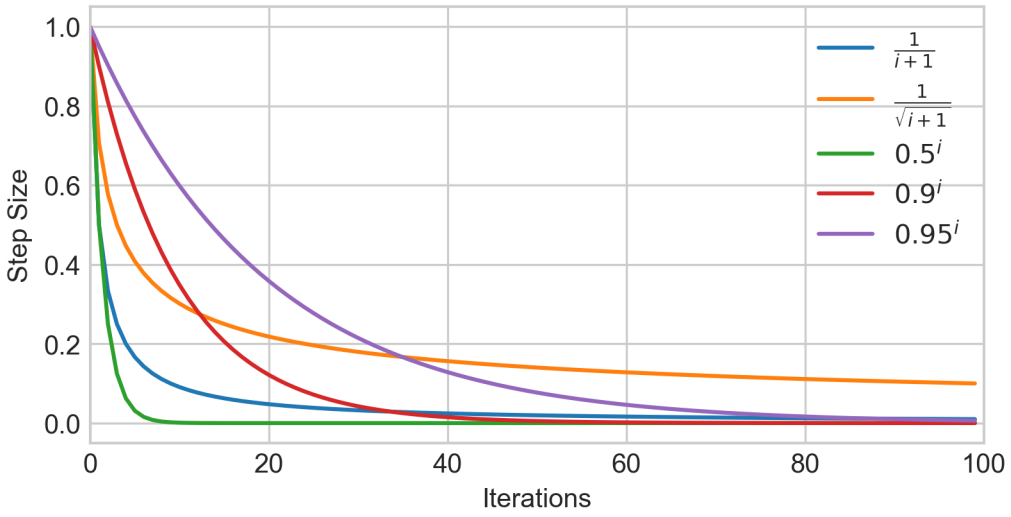
where i is the iteration number (beginning at 0), and $0 < \rho < 1$. Of these, the first two satisfy the second condition stated by Brännlund for guaranteed convergence, but the third one does not. As previously stated however, this does not mean that it will not work in this case, just that it is not guaranteed to converge, so it will still be tested alongside the other step sizes. These all have the characteristic of decreasing in an exponential-type curve, but even the variations between these can have a significant impact on the performance and behaviour of the subgradient method. Whilst the behaviour of these functions is familiar to us, fig. 5.1 helps to illustrate the differences between them, and the resulting impact on subgradient convergence times will be presented in section 7.2.2. In addition to these step size calculations being used on their own, two normalisations were also tested with each variation in order to reduce the impact of subgradient vectors with very large magnitude, the effects of which were observed in initial testing of these step sizes. The normalisations tested were $\frac{G_j}{N}$, where N takes the values values:

$$||G||_1 = \sum_i g_i$$
$$||G||_2 = \sqrt{\sum_i g_i^2}$$

There are also a number of other methods which can be used to improve convergence speed of the subgradient method. Deflection techniques, as presented by Camerini et al. [1975], uses a linear combination of the current subgradient and the previous direction in order to counteract the 'zig-zagging' behaviour that is observed in some instances. Frangioni et al. [2017] provide a comprehensive overview and testing of this and other methods such as projection to show the performance capability of the sub-gradient method when tuned correctly. Due to the behaviour described in section 5.2.2, no further tuning of the subgradient was carried out in this case as the decision was taken to proceed with a matheuristic approach, but these methods are invaluable in cases where the subgradient is being used and needs to be correctly tuned.

The full results of the performance of these step sizes, both standard and normalised, is

Figure 5.1: Comparison of step size values



presented in section 7.2.2.

5.2.2 Behaviour of Subgradient

During testing of the subgradient algorithm, some unusual behaviour was observed. When using initial multipliers $\lambda = \underline{0}$, the objective function value obtained at the first iteration is positive and integer as is expected, but from the second iteration where some multipliers have non-zero value, the objective function takes a very negative value (see section 7.2 for the full results). Over the subsequent iterations it begins to climb back towards the solution found in the first iteration, but the speed at which this happens (and whether it reaches it before convergence) is dependent on the step size used. The reason for this is explained in the following steps, followed by a more in depth proof.

1. Iteration 0: problem is solved with $\lambda = \underline{0}$, and achieves a positive integer valued objective function z_0, with solution x_0. This is to be expected due to the constraints and objective function of the problem.

2. A small number of dualised constraints are violated given solution x_0. The respective multipliers λ of these constraints are increased to penalise this violation.

3. Due to the structure of the problem (i.e. wide time windows) it is very easy for jobs

to be shifted in time. As a result, the inner problem now benefits from introducing a lot of slack in the previously violated constraints, leading to an overall negative objective value.

4. The multipliers that were increased in value in iteration 0 (and now benefit the inner problem) are reduced back to (or towards) 0 to remove this benefit. The multipliers of newly violated constraints are increased to penalise these violations.

5. The behaviour described in 3-4 repeats between iterations. Due to the fact that a very small proportion of constraints are violated at each iteration, there are always easy shifts that can be made to counter this.

6. As with any subgradient method, the step size for updating λ decreases over time. As such, the increases made to λ at each iteration become smaller, before being removed again at the next iteration. With the passage of sufficient number of iterations, the increases to λ become so small that the vector of multipliers tends towards $\underline{0}$. As a result of this, some step sizes cause the subgradient method to converge prematurely as the multipliers become too small to make any meaningful steps in the solution space.

This behaviour is also observed in instances where the initial multipliers are all given some positive value. Initially, the subgradient will return an objective with a negative value several orders of magnitude less than the expected solution, as the inner problem is able to benefit from a lot of slack in feasible constraints, and only a small number are violated in each solution. Figure 5.2 shows the number of non-zero multipliers at each iteration in the subgradient method for an instance of the problem with initial multipliers taking a random value between 0 and 0.1. Here we observe that the number of non zero multipliers decreases very rapidly at the beginning of the algorithm, and very soon stabilises to a number not far above the number of violated constraints at each iteration, confirming the idea that a number of multipliers are being decreased back to or towards zero at each iteration. Additionally, we see in graph fig. 5.2 that the average value of non zero-multipliers is converging towards zero, as described in point 6 of the list above.

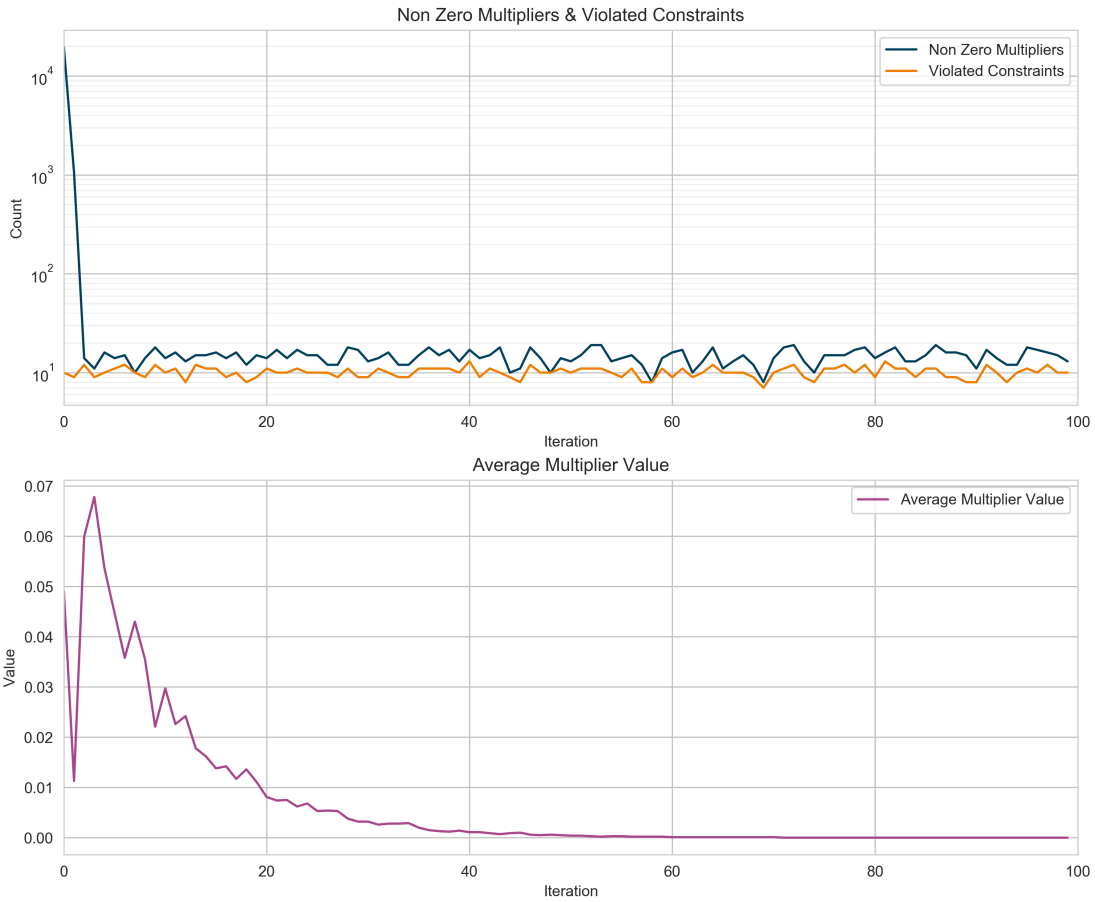


Figure 5.2: Count and Value of Non Zero Multipliers in the Subgradient Method - $\lambda_0 = \frac{\mathcal{U}(0,1)}{10}$

To formalise this explanation, we now present a proof of the behaviour of the subgradient method when the lagrangian multiplers take value $\lambda = \underline{1}$. Whilst this is only one case of the subgradient method, the same logic can be applied to all multipliers with positive value to explain the behaviour observed in the computation experiments carried out.

Proof of objective value at $\lambda = \underline{1}$

Let $(i^*, k^*, t^*) \in J \times K \times T$ be the combination of indices for which $x^{t^*}_{i^* j k^*} = 1$ for a given value of j for which $z_j = 0$. Then from the constraint

$$\sum_{(i,k,t) \in \mathcal{I}_{(j)}} x^t_{ijk} = z_j + 1$$

we can deduce that $x^t_{ijk} = 0$ for all $(i, k, t) \in (J \times K \times T) \setminus (i^*, k^*, t^*)$, or in other words all other x variables associated with j will take value 0. We then consider two different cases of the following dualised constraint:

$$u_j \geq t x^{t-\tau_{ij}}_{ijk} \qquad \forall (i, j, k, t) \in \mathcal{I} \tag{5.11}$$

Case 1: $(i, j, k, t) = (i^*, j, k^*, t^*) \;\Rightarrow\; u_j \geq t^*$

The maximum possible value of $t^* = |T|$, so for a given value of j the maximum violation that can be achieved by this constraint is $|T| - u_j$.

Case 2: $(i, j, k, t) \in (J \times J \times K \times T) \setminus (i^*, j, k^*, t^*) \;\Rightarrow\; u_j \geq 0$

This constraint holds for all valid combinations of (i, j, k, t), meaning there are $(|J||K||T|) - 1$ constraints for each value of j which each have a feasibility value of u_j. For any j for which $u_j = 0$, there will be no feasibility benefit from the constraints identified in this case, but for any value of $u_j > 0$, there will be a potential negative effect on the objective function of the Lagrangian dual, and this would be the case for up to $(|J||K||T|) - 1$ constraints if all their associated λ multipliers were positive. Conversely, there will only

ever be a maximum of one constraint which has the potential to increase the objective function value, as identified in case 1, if its respective λ multiplier is positive. Considering the start point where $\lambda = \underline{1}$, we define LR_j as the total contribution to the objective function by the constraint (5.11) for a given value j.

$$\begin{aligned} LR_j &= t^* - u_j - ((|J||K||T|) - 1)u_j \\ &= t^* - |J||K||T|u_j \\ &\leq |T| - |J||K||T|u_j \\ &= |T|(1 - |J||K|u_j) \\ &\leq 0 \iff u_j > 0 \end{aligned}$$

We can see here that for $u_j = 0$, LR_j can take a positive value, but it is bounded above by $|T|$. On the other hand, for positive values of u_j, the overall contribution to the objective function will be negative. Furthermore, as u_j increases this negative contribution grows significantly: if we consider the small example where $|J| = 8, |K| = 4, |T| = 40$, then an increase of 1 in u_j will result in a -1280 change in LR_j.

It is not unreasonable for us to disregard instances in which $|J| \leq |K|$, i.e. the number of jobs is not more than the number of technicians, as these could be considered trivial. As a result, amongst the remaining instances it is guaranteed that at least one job will have a start time $u_j > 0$, and in the current case of $\lambda = \underline{1}$, the feasibility benefit achieved by such jobs will far outweigh the penalty incurred by jobs with a start time $u_j = 0$. Even if we consider initial values of λ such that $0 < \lambda < 1$, the magnitude of the benefit of having many feasible constraints will decrease but the overall effect will be the same.

Similarly, we consider two different cases of the other dualised constraint:

$$u_j + \delta_{j2}z_j + \delta_{j1}(1 - z_j) \leq M_1 + (t - M_1)x^t_{jik} \qquad \forall (j, i, k, t) \in \mathcal{I} \tag{5.12}$$

Case 1: $(j, i, k, t) = (j, i^*, k^*, t^*) \Rightarrow u_j + \delta_{j1} \leq t^*$

For any given value of j, the maximum violation value for this constraint is $u_j + \delta_{j1}$, as

t^* has a minimum value of 0.

Case 2: $(j,i,k,t) \in (J \times J \times K \times T) \setminus (j,i,k^*,t^*) \;\Rightarrow\; u_j + \delta_{j1} \leq M_1$

Constraints of valid combinations of (j,i,k,t) will yield a feasibility value of $u_j+\delta_{j1}-M_1$. The minimum value of M_1 is $|T|$, as this will ensure that, for non binding constraints, (5.12) simply states that job j must be completed within the planning horizon of the stated problem. For any jobs which finish at the latest allowable time $u_j + \delta_{j1} = |T|$, all constraints from (5.12) which relate to this job will have a feasibility value of 0. For all other values of j however, it is guaranteed that $u_j + \delta_{j1} - M_1 < 0$, meaning there will be an overall negative contribution to the objective function for constraints with multiplier $\lambda > 0$. As with the previous constraint, if we let initial multipliers $\lambda = \underline{1}$, we define $\overline{LR}_j$ as the total contribution of the dualised constraint (5.12) to the objective function.

$$\begin{aligned}
\overline{LR}_j &= u_j + \delta_{j1} - t^* - ((|J||K||T|) - 1)(u_j + \delta_{j1} - |T|) \\
&= |T| - t^* + |J||K||T|(u_j + \delta_{j1} - |T|) \\
&\leq |T| + |J||K||T|(u_j + \delta_{j1} - |T|) \\
&= |T|(1 + |J||K|(u_j + \delta_{j1} - |T|)) \\
&\leq 0 \iff u_j + \delta_{j1} - |T| < 0
\end{aligned}$$

Similarly to the previous dualised constraint, in cases where $u_j + \delta_{j1} = |T|$ the overall contribution to the objective function can take positive value, but it is bounded above by $|T|$. However for all other jobs, the overall contribution will be negative, and in most cases with a much larger absolute value. Considering the same small problem as before, i.e. $|J| = 8, |K| = 4, |T| = 40$, a decrease in $u_j + \delta_{j1} - |T|$ by a value of 1 will change $\overline{LR}_j$ by -1280. Furthermore, disregarding the cases in which $|J| \leq |K|$ as with the previous constraint, there is a guarantee that at least one job will have finish time $u_j + \delta_{j1} < |T|$, i.e. it will finish before the end of the planning period, and as such its feasibility will take negative value of at most $|T|(1 - |J||K|)$, which will far outweigh the maximum penalties of $|T|$ from any other jobs.

The above examples do not provide definitive proof that positive Lagrangian multipliers will always yield a negative objective function for the Lagrangian dual problem, but it does explain why in many cases, included those considered in this work, the objective function takes a negative value at the beginning of the subgradient method.

In all the instances and variations of the subgradient method that were tested, no instances were observed in which the final objective value achieved after a fixed number of iterations (the highest number tested was 1000) was higher than that found in iteration 0 with $\lambda = \underline{0}$. In many cases it converged back to this value, but improvements were never found. It is possible that adjusting the step size to eliminate early convergence and running for a higher number of iterations would yield an improvement to the initial solution found, but in this case this was deemed unnecessary. Given the need for problems such as this to be solved in a reasonably quick time, carrying out a very high number of iterations (with a non-negligible computation time at each iteration) could lead to very high computation times to achieve very little gain on the initial solution found. For this reason it was decided to use the solution found when $\lambda = \underline{0}$ and use this as the starting point for a heuristic method to improve the solution.

5.3 Heuristic

As described in section 5.2.2, we know that the objective value obtained at the first subgradient iteration with zero valued multipliers provides a good lower bound on the optimal objective, but the associated solution is not guaranteed to be feasible for the full problem as it is obtained after removing two constraints. Figure 5.3 shows an example of what this infeasible schedule may look like, as taken from one of the instances used for testing. It is clear that this solution is not feasible due to the overlapping of many jobs: technician 0 for example has six separate jobs scheduled for the same time. This overlapping of many jobs occurs as a result of the constraints which were removed for the subgradient method. As such, the variables relating to travel (x^t_{ijk}) do not align with the order and start times of jobs in the initial solution as they are not constrained to do so. This and all other schedules in this section are used for illustrative purposes

only at this stage, the full results will be discussed in depth in section chapter 7.

The process for making this solution feasible and subsequently finding improvements is divided into multiple stages which are outlined in this section, beginning with reaching feasibility.

Algorithm 1 Make Schedule Feasible

1: **for** each techncian $k \in K$ **do**
2: Fix tour order for technician k from original solution
3: **if** Tour of technician k contains disjoint tours **then**
4: Swap nodes to join tours
5: **end if**
6: **end for**
7: **for** Each job $j \in J$ with $z_j = 1$ (two technicians) **do**
8: **if** j overlaps with job i for some $i \in J$ with $z_i = 1$ **then**
9: Find earliest start time $u_j = t$ such that j does not clash with $i, \forall i \in J, z_i = 1$
10: Add job j to schedule for technicians k and k'
11: **end if**
12: **end for**
13: **for** Each technician $k \in K$ **do**
14: **for** each job j in tour of technician k, $z_j = 0$ **do**
15: Add job to schedule for technician k at earliest possible time
16: **if** Schedule does not align with tour for technician k **then**
17: Correct tour
18: **end if**
19: Add travel between jobs to schedule
20: **end for**
21: **end for**

Algorithm 1 uses the solution output from iteration 0 of the subgradient method to determine a feasible solution to the problem. An example of this initial solution can be seen in fig. 5.3. It starts by constructing tours from the initial solution based on the arcs that technicians traverse, regardless of the time at which they do this (lines 1-6). This approach is used because many jobs overlap or have start times at zero, so attempting to construct a tour based on the initial start times of jobs would require a lot of alteration, whereas the remaining constraints in the problem ensure all jobs are visited, so there must exist tours (sometimes disjoint) which travel to all jobs when time is ignored. Disjoint tours are eliminated by swapping two jobs between the tours until each technician has a single complete tour. The next step is fixing the start times of those jobs which are completed by two technicians, as this is where the biggest difficulty

lies. Moving a job within the schedule of one technician can have knock on effects for many other technicians and jobs, as it may trigger a chain reaction of changes in start time. Each of these jobs is fixed at the start time given in the initial solution, or moved to the earliest possible time if this conflicts with other two technican jobs that have already been fixed (lines 7-12). The remaining solo technician jobs are then inserted around two technician jobs, attempting where possible to maintain the order of a technician's tour, but moving where necessary to avoid changing start times of jobs with two technicians (lines 13-18). Time for travel is then inserted between all jobs, shifting their start times later as necessary (line 19). From the initial solution, no z_j variables are changed, meaning the number of technicians assigned to each job is fixed. At this stage, no jobs move between technicians. More testing could be done at this stage to further improve the initial tour construction algorithm in lines 1-7, for example using a scheduling type approach based on existing algorithms such as the one presented by Nawaz et al. [1983].

Once the initial solution has been made feasible with algorithm 1, we are left with a solution like that seen in fig. 5.4 (some jobs extend beyond the planning horizon and therefore do not appear on the schedule). Here we see that whilst there are no longer any overlapping jobs, there are still improvements to be made: there are technicians who make multiple trips to the same location (indicated by colour), and there are jobs which finish beyond the end of the planning period. Improvements to this solution are achieved using the local search heuristic described in algorithm 2. This heuristic is divided in to two distinct stages: lines 3-28 deal with improving the schedule without moving any jobs between technicians, and lines 29-47 move jobs between technicians in order to reduce lateness. In only considering movement of jobs within their tour initially, we greatly reduce the number of solutions to be searched, whilst still observing a significant improvement in the objective. Once an iteration is reached where both stages are completed without any improvement being found then the heuristic can be terminated, or if this does not occur then the heuristic will terminate once the maximum number of iterations is reached. In the case of this work, the recommended value of both m and n is 10, the reason for which will be discussed in chapter 7. A more detailed

Algorithm 2 Local Search Heuristic

```
1: Initialise best objective (z*), max. completion time (f*), tours (r*) and start times
   (s*)
2: for m iterations do
3:     for n iterations do
4:         for Each technician k ∈ K do
5:             for Each job j in r_k, the tour of k do
6:                 for Each job h in r_k, the tour of k s.t. j ≠ h do
7:                     Move job j to the position of job h in r_k
8:                     Calculate the earliest possible start times for jobs in r_k
9:                     if z_j = 1 then
10:                        Find k', the other tech working on job j
11:                        Move job j in r_k' based on start time of j and jobs in r_k'
12:                    end if
13:                    Calculate all new start times s (see algorithm 3)
14:                    Calculate new objective z and max. completion time f
15:                    if z < z* then
16:                        Update z*, f*, r* and s*
17:                    else if z = z* then
18:                        if f < f* then
19:                            Update z*, f*, r* and s*
20:                        end if
21:                    end if
22:                end for
23:            end for
24:        end for
25:        if there is no change in z*, f*, r* and s* then
26:            break
27:        end if
28:    end for
29:    if Any jobs are late f* > T then
30:        Find L, list of all jobs which finish late
31:        for each job j ∈ L do
32:            for each tech k or pair of techs (k, k') qualified to do job j do
33:                Move job j to last position of r_k (and r_k' if z_j = 1)
34:                Calculate all new start times s (see algorithm 3)
35:                Calculate new objective z and max. completion time f
36:                if z < z* then
37:                    Update z*, f*, r* and s*
38:                else if z = z* then
39:                    if f < f* then
40:                        Update z*, f*, r* and s*
41:                    end if
42:                end if
43:            end for
44:        end for
45:    else
46:        Return z*, f*, r* and s*
47:    end if
48: end for
```

description of both stages will now be given.

Stage one considers all possible movements of each job without moving it to a different technician's tour. Within each technician's tour, every job is moved to the position of each other job in the tour in terms of job order (line 7). It then calculates the earliest possible start times for all jobs in this newly ordered tour, assuming the technician will travel directly from job to job with no idle time (line 8). If the moved job is also being completed by another technician, it is moved to the appropriate place in their tour such that start times are non-decreasing (lines 9-12). These changes may have created infeasibility with job start times, so the new earliest start times of all jobs are recalculated using algorithm 3 (line 13), described in more detail below. With these new tours and start times, the new objective function value is calculated and compared to the current best, with ties being broken by earliest maximum completion time, and then job index (lines 15-21). Repeating this for all jobs in all tours will give us the best improvement for that iteration. The number of comparisons to be performed at this stage is $\sum_{k \in K} |r_k|(|r_k| - 1)$, where r_k is the tour of technician k. This is at most $J(J-1)$ (in the case where all jobs are being completed by one technician), and is much smaller than this in practice as jobs are distributed across a number of technicians. This stage is repeated for a fixed number of iterations, or until no improvement in the solution is observed (lines 25-27). An example of what a solution may look like after this stage can be seen in fig. 5.5: we can see easily that the number of return visits to the same location has been reduced as jobs of the same colour are much more grouped in the schedule, and this is reflected in the objective function which has been reduced from 128 to 63, and the maximum completion from 54 to 44. Nevertheless, there may still be jobs which are completed outside their time window, one example of which is job 11 in fig. 5.5.

Given that lateness is one of the three elements in the objective function, moving those jobs which finish late to other technicians provides another possible area for improving the objective value, which is the purpose of stage two. Considering each job which finishes outside its time window, we consider all other technicians or pairs of technicians who are qualified to complete this job alone or together respectively. For each alternative

technician assignment, we move the late job to the last place in the tour of those one or two technicians. As in the previous stage, the earliest start times of all jobs are now calculated, and the new technician assignment which offers the greatest improvement is selected, breaking ties again by maximum completion time and then job index (lines 29-40). Once this has been completed for all late jobs, one full iteration of the heuristic is complete. If at least one late job was moved to a different technician, then the process can be started again from stage one, as there may be improvements to be made to the new tours.

Algorithm 3 Calculating Earliest Start Times

1: **for** each job j in tour r_k that has been changed **do**
2: Set start time of j to earliest possible time, $s_j = s_{j-1} + \delta_{j-1} + \tau_{j-1,j}$
3: **end for**
4: **for** n sufficiently large **do**
5: set schedule feasibility to **True**
6: **for** each tech $k' \in K$ **do**
7: **if** s_j in k' are not feasible, i.e. $s_i + \delta_i + \tau_{ij} \not\leq s_j$ for all adjacent jobs **then**
8: set schedule feasibility to **False**
9: **for** each job j in tour $r_{k'}$ **do**
10: Set start time of j to earliest possible time, $s_j = \max(s_i + \delta_i + \tau_{ij}, s_j)$
11: **end for**
12: **end if**
13: **end for**
14: **if** job feasibility is **True then**
15: break
16: **end if**
17: **end for**
18: **for** each tech $k \in K$ **do**
19: **for** each job $j \in r_k$ **do**
20: **for** each hour $t \in T$ **do**
21: set temp start $s'_j = s_j - 1$
22: **if** s'_j is feasible in r_k (and $r_{k'}$ if $z_j = 1$) **then**
23: $s_j = s'_j$
24: **end if**
25: **end for**
26: **end for**
27: **end for**

Algorithm 3 for calculating earliest possible start times of all jobs is also divided in to two parts: lines 1-17 ensure the feasibility of all start times given any recent changes to tours, and lines 18-27 remove any idle time that is present between jobs, allowing for

time spent waiting for another technician. Initially, the earliest possible start times for the new tour are calculated, assuming the technician will travel directly from job to job with no idle time (line 1-3). For each technician, we check the feasibility of their tour based on the existing start times of all jobs, and updated start times for those jobs which appear in the changed tour: if there is idle time then this is not changed as bringing the start time of a job forward may create new infeasibility elsewhere, but if there is overlap then all subsequent start times are pushed back to ensure they allow for job completion and travel between each pair of jobs (lines 7-12). This is repeated, looping through all technicians as many times as necessary until a feasible schedule has been reached. This process of pushing back start times may introduce unnecessary idle time between jobs, so stage two removes these gaps. Starting with the earliest jobs by start time, each job's start time is reduced by one hour at a time until it can no longer be made any earlier, allowing for completion of the previous job and travel between them (lines 18-23). This way we ensure that the jobs all have their earliest possible start time for the given arrangement of tours. This algorithm is used as a function in both stages of the local search heuristic 2.

Overall the full solution process for the proposed VRP formulation combines mathematical programming and heuristic methods, resulting in a matheuristic algorithm.

5.4 Hierarchical Approach

Given the possible size of a problem like this, it is also important to consider possible reduction methods to decompose the full instance into smaller, more manageable subproblems. One approach for the given problem is to use a hierarchical structure, solving first for the assignment of jobs to weeks for completion, and then solving the routing problem within each week (and possibly geographical division) separately.

Given a set of jobs to be scheduled, each of which has an assigned division, and a set of weeks during which all work must be completed, the task is to assign jobs to weeks so that the weekly duration of jobs in each division does not exceed the capacity for that division. At this stage, skill levels are ignored; we assume that the skill levels available

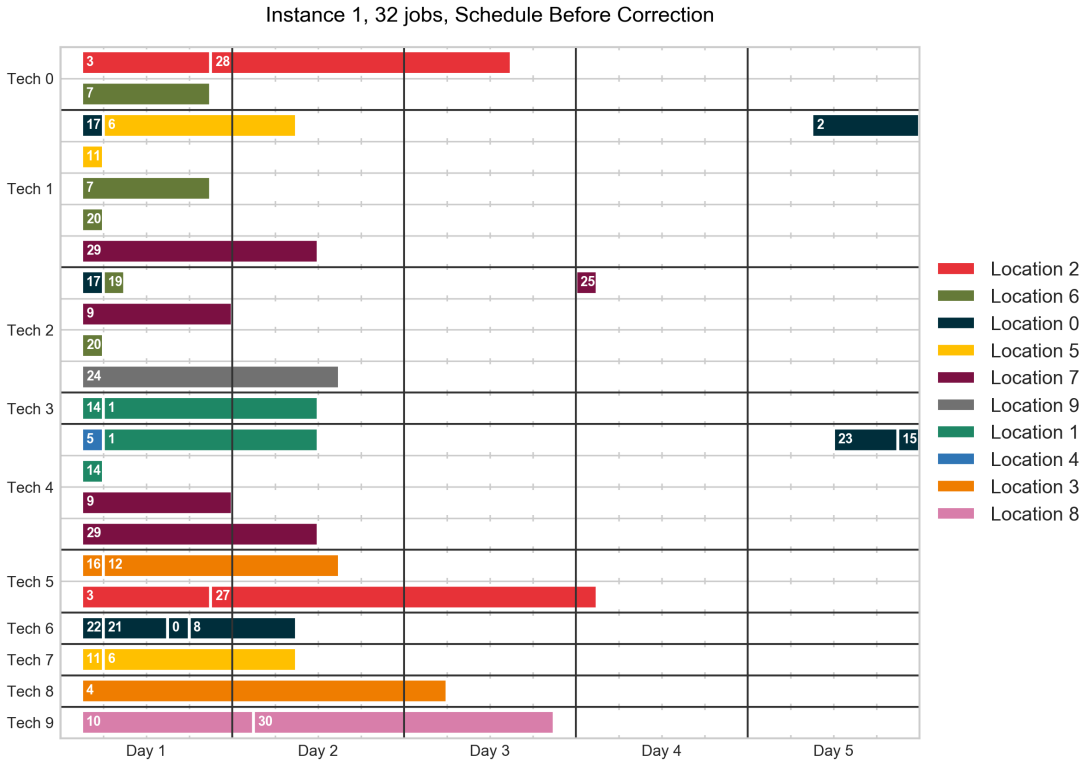
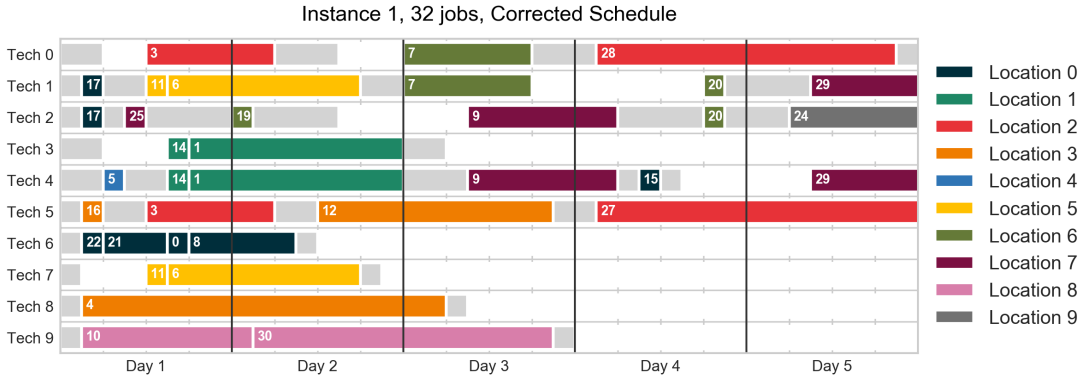


Figure 5.3: Schedule result directly from Subgradient method



Objective Value: 184, Maximum Completion Time: 52

Figure 5.4: Schedule solution directly after applying algorithm 1 to achieve feasibility (some jobs not visible at the end of tours)

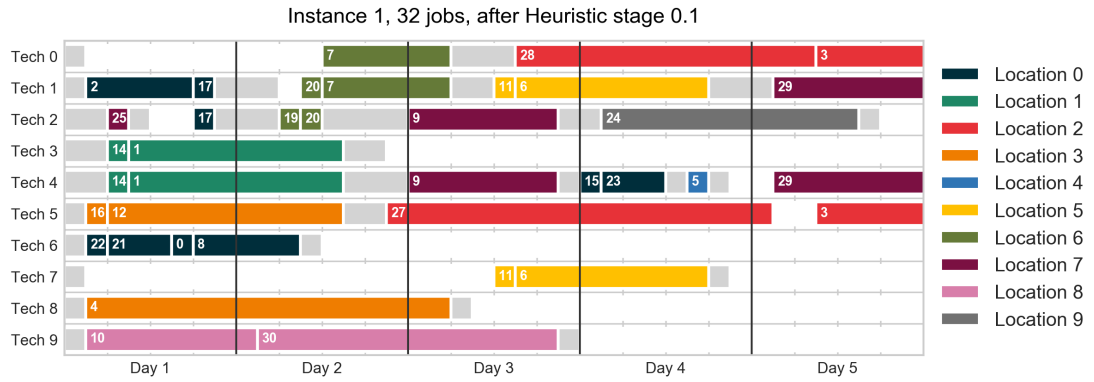


Figure 5.5: Schedule solution directly after first stage of algorithm 2, keeping all jobs with their original technician

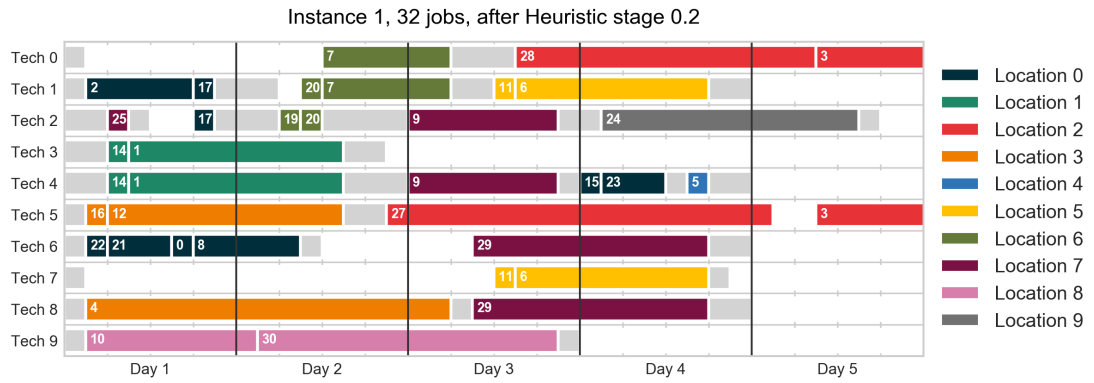


Figure 5.6: Schedule solution directly after second stage of algorithm 2, moving late jobs to different technicians

will be the same in each week (as we are dealing with the same set of technicians), and therefore any issues surrounding job infeasibility due to skill level will be an overarching problem across the whole planning horizon, not just in a given week. In other words, if no technician k possesses the correct skills to perform a particular job j in week w, then assuming homogeneity of technicians across weeks means it will be infeasible for all weeks in the planning period.

This can be viewed as a Generalised Assignment Problem (GAP), which can be stated as follows: given a set M of m agents, and a set N of n jobs, each agent $j \in M$ has a capacity C_j, and for each item and agent pair (i, j), we are given a resource required to complete the job r_{ij} and a profit p_{ij}. The objective is to find an assignment of items to agents that provides maximum profit. For the given problem, the costs and resource of each job are equal across all agents $j \in M$, a special case known as the Multiple Knapsack Problem (MKP). In addition, the presence of multiple divisions means this problem should also be treated as a Multidimensional Knapsack Problem (d-KP). Full explanations of both the MKP and the d-KP can be found in Kellerer et al. [2004].

Combining the MKP and the d-KP, the current task of decomposing the full problem can be formulated in two different ways: either we assign jobs in J to weeks in W (with a parameter to determine which division a job is in), or we assign jobs in J to division-week pairs in $D \times W$. The formulations for each of these approaches can be seen in (5.13) and (5.14) respectively.

$$\min \sum_{(j,w)\in A} w x_{jw} \tag{5.13a}$$

$$\text{s.t.} \sum_{w|(j,w)\in A} x_{jw} = 1 \qquad \forall j \in J \tag{5.13b}$$

$$\sum_{j)|(j,w)\in A} \delta_j r_{jd} x_{jw} \leq H_d \qquad \forall d \in D, w \in W \tag{5.13c}$$

$$x_{jw} \in \{0, 1\} \qquad \forall j \in J, w \in W \tag{5.13d}$$

$$\min z = \sum_{(j,d,w)\in\hat{A}} w x_{j(d,w)} \tag{5.14a}$$

$$\text{s.t.} \sum_{(d,w)|(j,d,w)\in\hat{A}} x_{j(d,w)} = 1 \qquad \forall j \in J \tag{5.14b}$$

$$\sum_{j|(j,d,w)\in\hat{A}} \delta_j x_{j(d,w)} \leq H_d \qquad \forall (d,w) \in D \times W \tag{5.14c}$$

$$x_{j(d,w)} \in \{0,1\} \qquad \forall j \in J, (d,w) \in D \times W \tag{5.14d}$$

Where

Parameters

J = set of jobs to be completed

D = set of working divisions

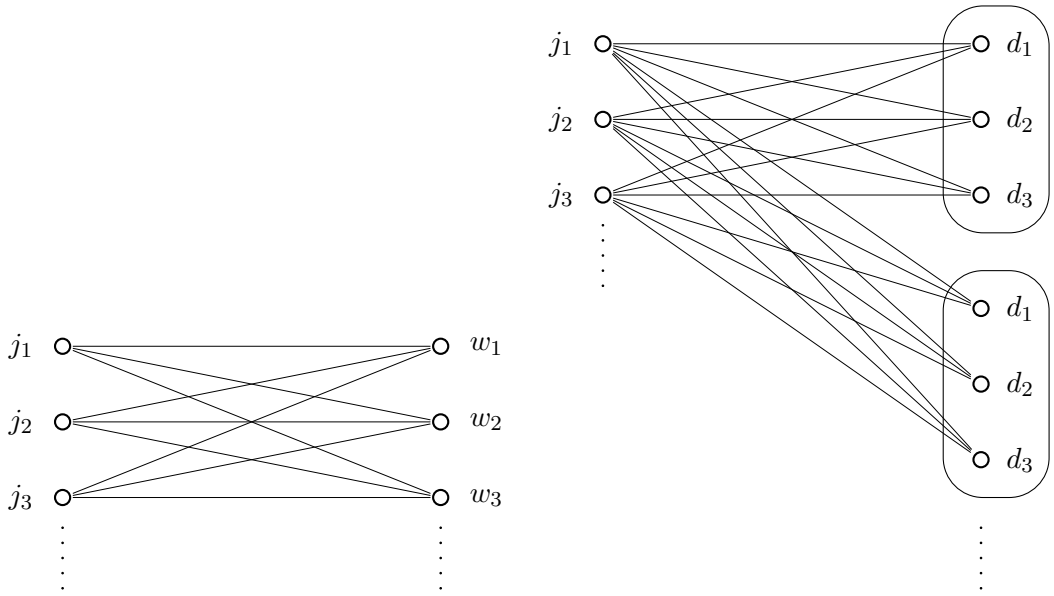
W = set of weeks

$$r_{jd} = \begin{cases} 1 & \text{job } j \in J \text{ is in division } d \in D \\ 0 & \text{otherwise} \end{cases}$$

δ_j = duration of job $j \in J$

$[a_j, b_j]$ = time window of job $j \in J$ $(a_j \leq b_j, a_j, b_j \in W)$

H_d = the total work hours available per week in division d



(a) Bipartite graph of 2D formulation (b) Bipartite Graph of 3D formulation

Figure 5.7: Example bipartite graphs

Decision Variables

$$x_{jw} = \begin{cases} 1 & \text{job } j \in J \text{ is assigned to week } w \in W \\ 0 & \text{otherwise} \end{cases}$$

$$x_{j(d,w)} = \begin{cases} 1 & \text{job } j \in J \text{ is assigned to the division-week pair } (d, w) \in D \times W \\ 0 & \text{otherwise} \end{cases}$$

The associated bipartite graphs of these special cases of the MKP can be seen in fig. 5.7.

If the decision variables x_{jw} and $x_{j(d,w)}$ are only defined for feasible combinations of j, d and w, it is easy to show that these formulations are identical with the presence of parameter r_{jd} to determine the division that each job is in.

Unplanned work can be handled as part of this approach in a number of different ways. Taking a robust optimisation type approach, it would be possible to reduce the weekly job capacity and in doing so ensure each week has available capacity for additional jobs. The exact reduction in the capacity constraint would need to be determined through analysis of the urgency and frequency of unplanned jobs. Such a method would not

guarantee enough time to complete all unplanned jobs that emerge, and it also may result in idle time if no unplanned jobs are reported to fill the time. In the event that the robust approach does not allow sufficient capacity for corrective jobs in a given week, or if the robust approach is not being used, then the small size and simplicity of this problem means it could be resolved at short notice to reassign jobs to weeks for completion. This would in turn also require the affected subproblems to be re-optimised at short notice, which reinforces the need to find a method for the problem which can achieve a solution in a short space of time. The issue of time windows can be handled by ensuring that the variables for a job j are only declared for weeks w which overlap with that jobs time window; all other extensions to the VRP which are included in this problem will be considered in the individual subproblems which are the main focus of this work, and therefore not included at this level.

Further investigation in to this technique for problem decomposition will not be presented here; there is a wealth of existing research on extensions of the Knapsack Problem and associated solution methods, and the reader is directed to Kellerer et al. [2004] for a comprehensive overview of these. The focus of this work will be solving the problem at a divisional level over short planning periods, which are small enough to be solved independently. Such subproblems (i.e. the allocation of jobs to weeks) may be created manually as is currently done by the RNLI, or they could be determined by solving either of the formulations presented above. The formulation presented in chapter 4 is independent of this hierarchical method: splitting the problem would simply aim to reduce the size of each VRP to be solved and not the constraints being considered, and as such the formulation can be used with or without the preceding work to divide the problem instance.

5.5 Implementation

Initial testing with regards to which constraints to dualise was carried out in AMPL. Due to the fact that this software requires a license, the decision was taken to transfer all subsequent computational work to Python so it can be implemented without the

need for a license. Whilst testing was carried out using CPLEX to solve the initial Lagrangian Relaxation problem, the Python package PuLP allows for formulations to be solved with any solver, enabling the use of free solvers such as COIN-OR if required in application.

Chapter 6

Data

We were only able to obtain a limited amount of data from the RNLI, an overview of which is given in the first part of this chapter. This is followed by details of the methods used to generate data for the purposes of testing, as not enough was provided from the RNLI to carry out extensive testing.

6.1 Job Data

In total, information on 14,043 jobs over one year (excluding December) was provided; of interest to us at this stage is the job priority (defined below), type (corrective or preventative) and expected duration. Of these, ∼3,400 jobs are listed as having an expected duration on zero. As this data could be viewed as incomplete, all summaries and statistics presented in this chapter have been carried out both including and excluding these jobs.

One thing that we define here is the notion of job priority. This does not explicitly impact on the problem formulation in terms of prioritising jobs; instead it categorises jobs based on the time within which they need to be completed, giving us the time window (e_j, f_j) described in section 4.1.4. The categories are as follows:

'Next visit' refers to the job being completed on the next time the station is visited by

Priority 1:	Immediate – 24 hours
Priority 2:	2 – 7 days
Priority 3:	8 – 30 days
Priority 4:	Next visit

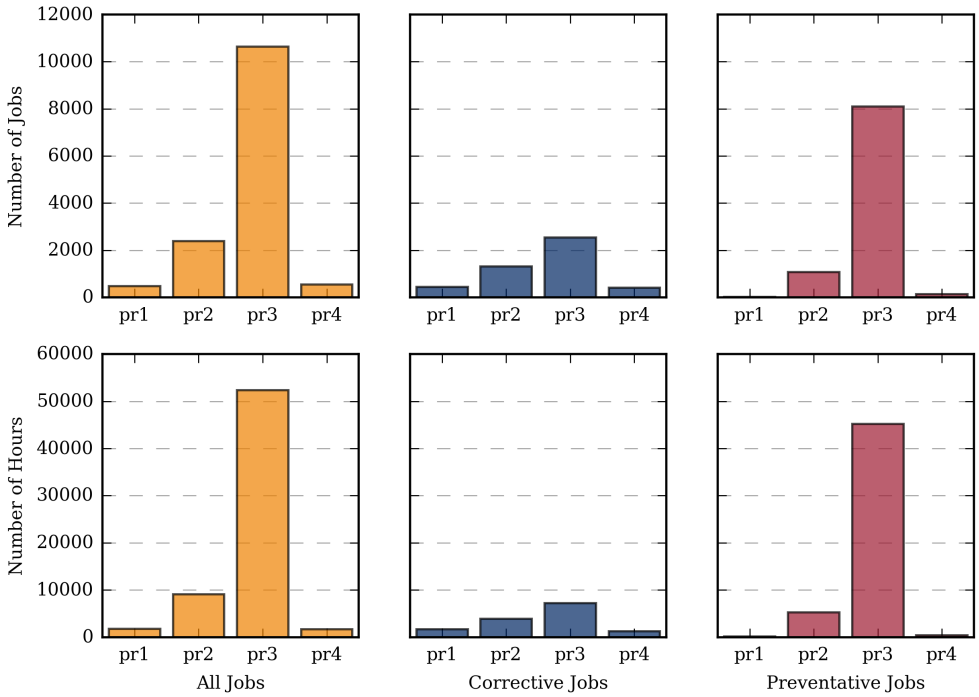


Figure 6.1: Number of Jobs and Job Hours by Type

an ST, on the basis that stations are visited roughly twice per year. This is equivalent to saying it has a time window of 'Immediate – 6 months', and can thus be scheduled any time during the planning period.

Additionally, all jobs are categorised as either corrective or preventative: the former are jobs which are generated at the start of the planning period (every three months), whereas the latter are jobs which emerge as problems develop. All of these jobs are assigned a priority level from those described above. The preventative jobs are those which will be used as the initial input in to the problem formulation (as these are the jobs that are known at the beginning of the planning period). The corrective jobs will need to be incorporated in to ST routes as they emerge, or for those with lower priority they can be added to the set of jobs to be completed over the subsequent weeks.

The graphs in fig. 6.1 show the number of jobs and job hours in each priority, for each job type. From this, we can clearly see that, for all of these categories, the most common priority is 3, that is, those jobs which must be completed within 8-30 days. More specifically, these jobs made up 75.7% of the total number of jobs, and 80.67% of the number of hours. In comparison, jobs with immediate priority made up only 3.37% of all jobs, and 2.77% of hours. A full breakdown of this data can be found in appendix A.1. In addition, we consider whether there is any seasonal behaviour in the pattern of when jobs are created: fig. 6.2 shows a plot of the number of corrective and preventative jobs over the same year. The most notable feature of this graph is the large number of preventative jobs generated in the first month of the year, at least three times that of any other month. This is likely to be due to the timing of decisions for large scale work and upgrades, meaning a large number of jobs for the coming year are created around the same time. Of more interest for the purposes of this problem is whether there is any seasonality in the creation of corrective jobs. With this we see an increase in the months June to September; this is perhaps unsurprising, as the number of calls being responded to by the RNLI is likely to be higher over the summer months due to an increased number of people spending time on, in or near the water, and as such is possible that more frequent use of vessels could lead to a higher rate of repair work needing doing.

The full results when broken down by month can be seen in table A.5.

Whilst these results do not directly impact the formulation presented in chapter 4, they will help to shape the approaches used in considering emergent jobs. This may be implemented in very different ways for instances with high and low numbers of corrective jobs, and having an idea of how many corrective jobs will emerge during any given period will help to influence these decisions.

6.2 Instance Generation

In order to be able to carry out robust testing in the absence of real data, randomly generated instances were used. The python script allows the user to specify the size

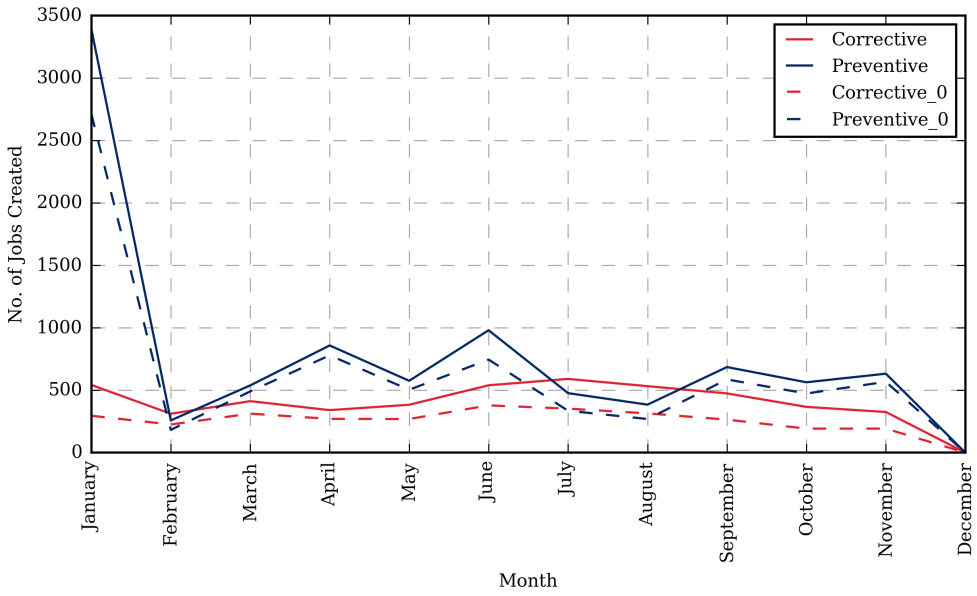
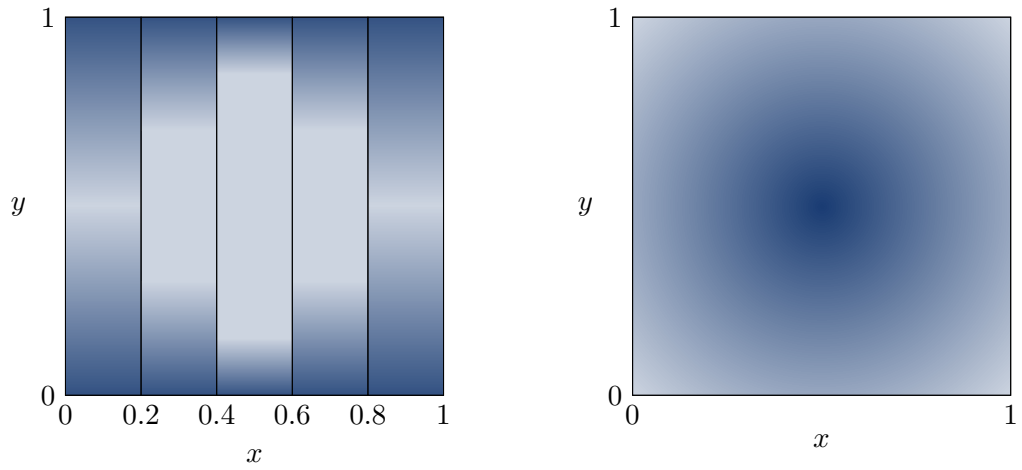


Figure 6.2: Number of jobs created per month ('_0' indicates that jobs with zero duration have been ignored)

of the problem and the random seed. In this way, there is no limit to the number of files that can be generated, and they can be easily replicated if necessary by using the same seed. The inputs of the generation function and their descriptions can be seen in table 6.1. The only constant values between different instances are the number of hours per week, 40, the number of days per week, 5, and the cutoff beyond which technicians must stay away from home, 1.5 hours. The rest of the data is generated as per the process outlined below.

Table 6.1: Table showing arguments of Data Generation function

Argument	**Filename Code**	**Description**
`locs`	`C`	Number of job locations, i.e. lifeboat stations in this case
`dpts`	`H`	Number of depots or bases of technicians
`jobs`	`J`	Number of jobs to be completed
`techs`	`K`	Number of technicians available
`weeks`	`W`	Number of weeks in the planning period
`skls`	`S`	Number of distinct skill areas
`lvls`	`L`	Number of skill levels within each skill area
`initial_seed`	`SD`	Initial seed for the problem randomisation



Distribution of job nodes Distribution of base nodes

Figure 6.3: Approximate representation of node distributions

Each instance will be saved with a generated filename: `data_SD0_C4_H2_J8_K4_W2_S2_L2.csv`. At several stages throughout the instance generation, the random seed is reset to ensure similarity between instances of different sizes. This way, increasing the size on an instance with a fixed seed will have the effect of adding in additional jobs and/or technicians, whilst the existing ones and their respective information remain the same.

6.2.1 Node Locations

Job Node Locations

Job locations are randomly generated on a unit square. In order to emulate the nature of the RNLI problem, job locations should be more likely to occur around the edge of the square. The x-coordinate is taken form a uniform distribution, and categorised according to its value. The y-coordinate is taken from a reflected normal distribution whose parameters are dependent on the category of the x-coordinate, meaning an x coordinate in one of the outermost strips of the unit square will have an almost equal distribution across the y-axis, but the strips nearer the centre will be much more likely to generate a y-coordinate near the edge of the square. A visual approximation of this distribution can be seen in fig. 6.3.

Base Node Locations

Similarly to job locations, base locations are also generated randomly on a unit square, but in this case they should be more likely to occur nearer the centre of the square, again to emulate the typical RNLI problem. Both the x- and y-coordinates are generated with a normal distribution, with a visual approximation also shown in fig. 6.3.

Inter-node Distances

The distances between all nodes are calculated based on their cartesian distance in the unit square. In order to be as similar to the RNLI problem as possible, these cartesian distances were multiplied by 3 and rounded to the nearest integer, as the problem considers discrete units of time, in this case hours.

6.2.2 Job Information

Job Locations

Each job in the instance is randomly assigned one of the existing job nodes using a uniform distribution, from which it takes its own location coordinates. If there are more jobs than job locations (which is very likely), there is guaranteed to be at least two jobs which share a location. These jobs will have the same coordinates, and therefore a travel time of 0 between them, but they will still be considered separate nodes for the purposes of this problem.

Time Windows

The length of the time window for each job, given in weeks, is taken from the distribution $\lfloor \mathcal{U}(1, \min\{|W| + 1, 5\}) \rfloor$. The maximum time window for any job is four weeks, but for planning periods shorter than this, a lower cap on the length of time windows is imposed. From this, the start of the time window is taken from the floor of a uniform distribution, ensuring that it will start early enough to allow the whole job time window to fall within

the planning period. The finish time is simply calculated from the start time and time window.

Number of Technicians Required

in the RNLI case, the majority of jobs can be completed by one technician, and for many they can *only* be completed by one technician. Some jobs however can be completed by two technicians, with the assumption that this will half the job duration, and some require two to be completed. There are assumed to be no jobs which require more than two technicians, as this is not included in the problem formulation. In order to reflect this, the minimum number of technicians for a job takes value 1 with probability 0.8, and 2 otherwise. If the minimum number of technicians is 2 then the maximum will also be 2, otherwise the maximum number of technicians is 1 with probability 0.8, or 2 otherwise.

Skill Requirements

Skill requirements for a job are simply calculated using a uniform distribution. In each skill domain, the skill requirement takes the value $\lfloor \mathcal{U}(1, |L| + 1) \rfloor$, an integer between 1 and the number of skill levels in the problem.

Job Duration

Job duration is calculated as $\delta = \min\{\max\{\lfloor \mathcal{N}(6, 6) \rfloor, 1\}, \tau^*\}$, where $\tau^* = \Omega - 2 \max_{i,j}\{\tau_{ij}\}$, the maximum duration of a job assuming a technician can travel there, complete the job and travel back to their base within one week. This value is the duration if the job when completed by its minimum number of technicians; if this minimum is 1, and the maximum is 2, then the duration of the job for 2 technicians is $\lceil \frac{\delta}{2} \rceil$.

Lateness

The allowable lateness of jobs is stated in days, and in these instances has a maximum value of 5 days. By generating a number $p = \mathcal{U}(0, 1)$, the allowable lateness value is determined according to the value of p according to the table below:

	Lateness (days)
$0 \leq p < 0.5$	0
$0.5 \leq p < 0.7$	1
$0.7 \leq p < 0.8$	2
$0.8 \leq p < 0.9$	3
$0.9 \leq p < 0.95$	4
$0.95 \leq p$	5

If a a job has a time window which extends to the end of the planning period, then it will be constrained to finish on time by the formulation. Such jobs will still be assigned a lateness, in case they are used in another longer instance, but otherwise the lateness will be dominated by other constraints in the formulation.

6.2.3 Technician Information

Technician Base

Each technician is randomly assigned one of the available base nodes by way of a uniform distribution, from which it takes its base coordinates.

Skills

Skill levels for technicians are generated from a normal distribution. This is then restricted to be between the values of 0 and $|L|$, giving the following calculation for skill level:

$$\min\left\{|L|, \max\left\{0, \left\lfloor \mathcal{N}\left(|L|+1, \left(\frac{|L|}{2}\right)^2\right)\right\rfloor\right\}\right\}$$

This means technicians are most likely to be fully trained in each skill domain, and their probability of being any given level decreases as the level decreases.

Work Hours

In these instances, technicians are all assumed to work full time, i.e. 40 hours per week.

6.3 Objective Function Weighting

In order for the formulation and method presented in this thesis to be applied to the RNLI, weightings for the objective function would need to be determined. One reason behind the decision to optimise all divisions separately is that each division will be able to provide its own objective function weightings to capture the subtle differences between each area. An example of this would be nights away: in areas where jobs are more spread out such as Scotland, having to stay away for a night may be almost inevitable in some cases, so less importance might be placed on reducing these as it is accepted that they will be necessarily. Conversely, for technicians covering a relatively small area with short distances between stations there is a very real possibility that route selection may determine whether or not they have to stay away, and as such reducing these may be a higher priority.

In the absence of real data, the weightings used in the examples tested in this work were all equal. In a case where this work is being applied and these weightings need to be determined, there are existing methods which can be used to aid this decision, such as the Analytic Hierarchy Process (AHP) described in Ishizaka and Nemery [2013]. Using the AHP, the problem is arranged in to a hierarchical structure (see fig. 6.4), where the root node (level 1) represents the decision to be made, the next level the criteria to be considered (level 2), and below that each criteria is broken down in to two or more

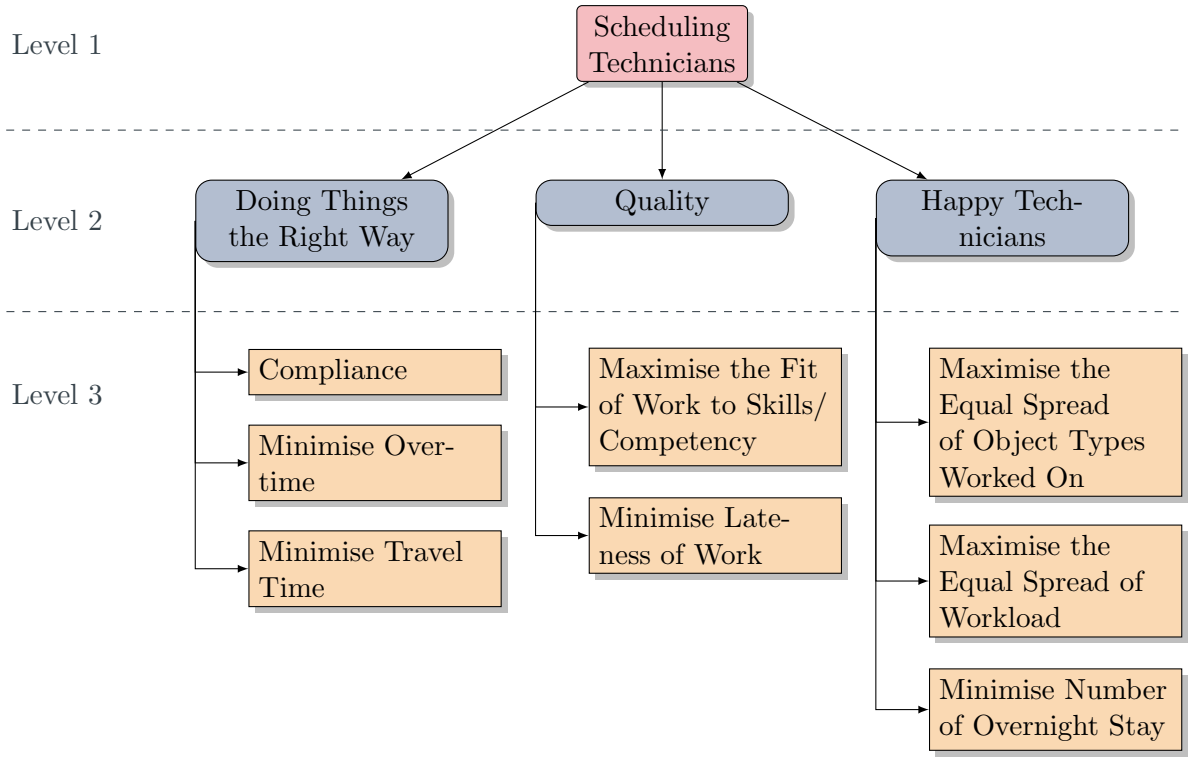


Figure 6.4: Tree of Decision Criteria

subdivisions (level 3). Rather that generating an overall ranking of these criteria and sub-criteria manually, pairwise comparisons are used, as Psychologists such as Yokoyama [1921] state that it is easier and more accurate to express preference between two options, rather that ranking more than two options together. Collating all of these comparisons enables an overall preference ranking to be calculated.

The structure of the decision tree is important, as Weber et al. [1988] note that a criteria with a higher number of sub-criteria is more likely to receive a higher overall ranking. In addition, if there are n criteria, the number of pairwise comparisons that must be made is $\frac{n(n-1)}{2}$, so a large number of criteria will result in a process that is time-consuming, and may therefore be discouraging for participants. As such, effort should be directed towards ensuring that the tree is of a reasonably small and balanced nature, to increase the overall effectiveness of the task. AHP is traditionally used to compare a discrete (and easily counted) set of possible outcomes according to the given criteria; whilst the problem we are dealing with is technically discrete in the sense that it uses integer decision variables, we are not able to consider all possible solutions as is suggested for

AHP. Nonetheless, this approach can still be applied to the given problem, by utilising the process to generate criteria priorities, without then applying these to individual decision options.

The criteria in fig. 6.4 are for the purposes of demonstration; whilst they do not necessarily represent the final criteria that will be used with this process going forward, they give an idea as to the types of criteria that are being considered in this problem. To perform this task, Transparent Choice's AHP software would be used [Choice], which automates the creation of the 'quiz' to allow participants to compare all possible pairs, and the calculation of criteria priorities from this. These would then be used to determine appropriate weightings to be used in the objective function. This method however is not without flaws. It is possible that someone completing the comparison quiz could inadvertently create a cyclic order of priority, especially in cases with a large number of comparisons as it becomes harder to keep track of previous answers. Despite this, in a survey of multiple criteria decision making processes, Wallenius et al. [2008] highlight that AHP is the fastest growing method of this type in terms of number of publications. No further methods will be presented here, but the reader is referred to the work by Wallenius et al. for further information on alternative methods.

Chapter 7

Results

In this chapter we present the findings of this work, from initial testing through to final computational results. First, we will discuss the results used to identify which constraints should be dualised to form the Lagrangian dual problem, followed by the results obtained from implementing the Subgradient algorithm with this dualisation. The output from the Subgradient method was then used as the basis of a heuristic algorithm, the results of which will be the final part of this chapter. Initial testing was carried out with small problem instances in order to determine which solver should be used for any instances of the MILP or its relaxations. CPLEX and Gurobi provided the quickest solution times compared to other solvers, but performed equally when compared to each other, so CPLEX was chosen for all of the work presented in this chapter.

7.1 Identifying Constraints to Dualise

7.1.1 Initial Results

An important factor in the successful implementation of the subgradient algorithm is the selection of constraints to be dualised, as this will impact the solution time of the resulting problem. In order to determine the best set or sets of constraints to dualise for this problem, experiments were run with each individual set of constraints being

Table 7.1: Computational details for section 7.1.1

Software	AMPL (20181005)
Solver	CPLEX (12.5.0.0)
Computer	Iridis 4
Processors	6 x 2.6GHz

Table 7.2: Average time to reach optimality with individual constraints removed

Constraint	**Average Time to Optimality**
(4.35)	00:07:03
(4.36)	00:31:51
(4.37)	00:04:16
(4.40)	00:21:48
(4.12)	00:24:21
(4.41)	00:24:16
(4.42)	00:31:42
(4.61)	00:22:02
(4.44)	00:25:03
(4.45)	00:36:31
(4.46)	00:00:08
(4.47)	00:00:10
(4.38)	00:49:41
(4.62)	00:31:22
(4.51)	00:13:40
(4.56)	00:20:40
(4.57)	00:11:54

removed from the formulation to establish the effect on solution time. As these were carried out in the initial phases of this work, they were calculated using AMPL. The full computational details are listed in table 7.1. These were tested on eight instances of size $|J| = 8, |H| = 2, |K| = 4, |S| = 2, |L| = 2, |W| = 2$; each of the listed constraints was removed from each of the eight instances, and the problem solved to optimality to record the solution time. A summary of these results can be seen in table 7.2.

We observe a wide range of solution times across the different problem formulations, ranging between 8 seconds and almost 50 minutes. As these tests were carried out on a

test size dataset, we expect to observe an increase in solution times when it is used to solve instances of a more realistic size, however the impact of removing each constraint on the solution time is still clearly visible here. The two constraints which lead to the greatest reduction in solution time are

$$u_j \geq t x_{ijk}^{t-\tau_{ij}} \qquad \forall (i,j,k,t) \in \bar{\mathcal{I}} \qquad (7.1)$$

$$u_j + \delta_{j2} z_j + \delta_{j1}(1 - z_j) \leq M_1 + (t - M_1) x_{jhk}^{t} \qquad \forall (j,h,k,t) \in \mathcal{I} \qquad (7.2)$$

Constraint (7.1) ensures the start time of a job is not earlier than the the time at which its technician(s) arrive, and similarly (7.2) ensures technician(s) do not leave until a job has been completed. Whilst the removal of either of these constraints represents a significant improvement from solving the problem in full, further testing is required to establish whether this reduction will be sufficient when applied to problems of a bigger size, which will be presented in the next section.

7.1.2 Full Problem Size

Having established which constraints lead to the greatest reduction in solution time when removed, we now continue testing for these two constraints in order to determine which should be used in the final lagrangian dual formulation. Similar to the testing described in section 7.1.1, we tested each of the constraints separately, but this time implementing the subgradient method over a maximum of ten iterations. In order to test the performance of these dual problems on larger instances, the problem size was increased to $|J| = 20, |H| = 2, |K| = 10, |S| = 2, |L| = 2, |W| = 1$. This size was chosen based on the average number of technicians per division in the RNLI. The computational details are given in table 7.3 and the results in table 7.4 show the average solution time per iteration across six different instances.

As expected, the average solution time at this stage has inreased significantly when compared to the previous results from a test size problem. We also observe a much

Table 7.3: Computational details for section 7.1.2

Software	Python (3.6.8)
Package	PuLP (1.6.9)
Solver	CPLEX (12.5.0.0)
Computer	Iridis 4
Processors	6 x 2.6GHz

Table 7.4: Average time for subgradient iterations by dualised constraint

Constraint	**Average Time per Subgradient Iteration**
c_tw_arrive	0:58:18
c_tw_leave	0:03:40

bigger difference between the two constraints, but the times for both are higher than is really suitable for a subgradient method: even when considering a relatively small number of iterations, the total time to implement the full subgradient method could become prohibitively large, especially when considering a problem which may need to be resolved regularly as new jobs emerge. In order to further reduce the average time required to complete one subgradient iteration, both constraints were dualised together.

Combining these two, the new objective function of the Lagrangian dual problem is

$$Z + \lambda(t\bar{x}^t_{ijk} - u_j) + \mu(u_j + \delta_{j2}z_j + \delta_{j1}(1 - z_j) - (M_1 + (t - M_1)x^t_{jhk})) \tag{7.3}$$

where Z is the objective function of the original problem. We then implemented the subgradient algorithm on the new Lagranigan dual problem, across the same instances and for a maximum of 1000 iterations, the results of which can be seen in table 7.5. These results show a significant improvement in the average iteration time for each instance when compared to the solution times for each constraint being dualised separately (table 7.4), with an overall average of 16.12s.

Another advantage of the selection of these particular constraints for dualisation is the impact it has on the objective function when these constraints are removed from the

Table 7.5: Average iteration time of subgradient algorithm with double dualised constraints

Instance Seed	**Average Iteration Time (s)**
0	17.81
1	15.70
2	11.99
3	15.21
4	16.53
5	15.22
6	13.78
7	14.29
8	28.39
9	12.23

full formulation. Given an optimal solution from the subproblem with (7.1) and (7.2) removed, the remaining constraints on the x^t_{ijk} variables will ensure that all jobs still have a technician arrive to and depart from them. In other words, if we consider the network that represents this problem, the correct number of arcs will be traversed to provide a complete solution, the absence of the dualised constraints simply means the timings of travel along these arcs will not necessarily align with the start times of jobs. When we consider the full objective function without dualised penalty terms, the only variables present are x^t_{ijk}, λ^{wd}_{jk} and σ_j, which refer to travel between nodes, overnight stays and lateness respectively. As such, a solution with the correct number of arcs will provide a strong lower bound to the travel element from the objective function of the full problem, even if this solution violates constraints (7.1) and (7.2), as all jobs are still being visited and the travel between them contributes to the objective value, regardless of whether this travel aligns correctly with the start times of the jobs. On the other hand, we do not observe the same behaviour for all constraints. We consider the constraint which provides the next quickest average subgradient iteration when dualised in section 7.1.1:

Table 7.6: Objective values of problem instances based on dualised constraint

Seed	(7.4)	(7.1)	(7.2)
0	10	20	20
1	7	5	5
2	11	20	20
3	11	20	20
4	12	20	20
5	10	20	20
6	4	2	2
7	11	20	20
8	5	5	5
9	11	20	20

$$\sum_{(i,t)\in\mathcal{I}_{(j,k)}} x^t_{ijk} = \sum_{\substack{i\in V, t\in T \mid \\ (j,i,k,t)\in\mathcal{I}}} x^t_{jik} \qquad \forall j \in V, k \in K \tag{7.4}$$

This constraint ensures that, for each technician that arrives at a given node, they must also depart that node. By removing (7.4) from the full formulation instead of the two constraints chosen previously, the resulting optimal solution will not necessarily traverse the correct number of arcs to provide complete tours, possibly resulting in a worse lower bound. This is reflected in the maximum objective values achieved by the subgradient method when testing the dualisation of each of these constraints, as shown in table 7.6. Here we see that for most instances the best lower bound is achieved when either (7.1) or (7.2) are dualised.

In both of these cases the effect of the dualised constraints in the objective has been ignored; of course in cases with non zero multipliers the objective values produced by the lagrangian dual would be much lower than the optimal objective value due to the penalty terms, as described in section 5.2.2, but the solution they produce will still be relevant. In a case where no optimal solution can be found using the subgradient method, using the solution from one of the iterations will still provide a lower bound to the optimal solution, and this bound is more often stronger when either or both of (7.1) and (7.2) are

dualised as opposed to (7.4). As such, there is not only a computational speed reason to select these constraints for dualisation, but also the fact that they will provide a good lower bound to the full problem. The solution itself would not be optimal, but it could provide a starting point for a heuristic to find a feasible solution.

7.2 Subgradient Results

We now present the results obtained through the testing of a number of variations of the subgradient method, based around changes to the step size used in the calculation of the lagrangian multipliers. As it has been shown that the best option for dualising constraints is to use both (7.1) and (7.2), all results from this point are only for this particular case.

7.2.1 Step Size $\delta = \frac{\pi(Z_{UB}-Z_{LB})}{\sum_{i=1}^{m} G_i^2}$

First we consider the step size proposed by Beasley [1996] from the original subgradient method discussed in this work, which is:

$$\delta = \frac{\pi(Z_{UB} - Z_{LB})}{\sum_{i=1}^{m} G_i^2} \tag{7.5}$$

where $G_i = b_i - \sum_{j=1}^{n} a_{ij} X_j, \quad i = 1, \ldots m$, in other words the violation or slack in each constraint, and z_{UB} is a known upper bound on the objective function of the full problem. In this case, this upper bound was found by solving the full problem with the CPLEX parameter 'mip solution limit' set to one, so CPLEX returns the first integer solution found. The subgradient method described in section 5.2 was implemented with this stepsize on instances of size $|J| = 8, |H| = 2, |K| = 4, |S| = 2, |L| = 2, |W| = 2$. The first ten iterations based on the instance with seed 0 produced results which can be seen in table 7.7.

We observe from the first iteration that the step sizes take a very small value, which is down to the very high value of the denominator, $\sum_{i=1}^{m} G_i^2$, the sum of squares of the

Table 7.7: Initial results for Beasley step size

Iteration	**Status**	**Time (s)**	**Violation Count**	**Step Size**	**Obj. Value**	**Max. Obj.**	$\sum_{i=1}^{m} G_i^2$
0	Optimal	2.281	8	$< 10^{-3}$	8	8	75413351
1	Optimal	2.565	7	$< 10^{-3}$	7.999	8	80744304
2	Optimal	2.611	7	$< 10^{-3}$	7.999	8	86614714
3	Optimal	2.403	7	$< 10^{-3}$	7.999	8	83350692
4	Optimal	2.478	6	$< 10^{-3}$	7.999	8	86621698
5	Optimal	2.308	8	$< 10^{-3}$	7.999	8	80596839
6	Optimal	2.471	8	$< 10^{-3}$	7.999	8	80743693
7	Optimal	2.271	8	$< 10^{-3}$	7.999	8	87009880
8	Optimal	2.4	8	$< 10^{-3}$	7.999	8	84973208
9	Optimal	2.338	7	$< 10^{-3}$	7.999	8	84588183

subgradient vector. This inhibits the subgradient method from moving very far from the initial multipliers $\lambda = \underline{0}$, and as a result there is very little change in the objective function value between iterations. In order to verify that the behaviour observed is not unique to this instance, table 7.8 shows the average magnitude of the subgradient vector across 100 iterations from 10 different instances of the problem. For each instance this value is extremely large, and in the same order of magnitude as the instance described above.

The values in the column 'Violation Count' of table 7.7 indicate the number of violated constraints at each iteration, and in all cases this is very small, between six and eight. The total number of constraints is of the order $|J|^2|K||T|$ which is greater than 10,000 even in this small example, and as shown in the proof in section 5.2.2 some of these are guaranteed to have positive slack. Given the very small number of violated constraints, there are a large number which are guaranteed to be feasible, each by a value of up to $2|T|$. As such, summing across the squares of these values leads to a very large denominator $||G||^2$. These results suggest that the step size provided by Beasley [1996] is not suitable for the given problem, and as such alternative step sizes were also considered.

Table 7.8: Magnitude of violation for multiple instances across 100 iterations

Seed	**Average** $\sum_{i=1}^{m} G_i^2$
0	83,065,656.2
1	95,236,147.6
2	83,910,959.8
3	115,587,754.8
4	105,399,882.0
5	86,549,877.1
6	97,944,731.3
7	87,190,024.0
8	57,564,737.0
9	72,813,594.9

7.2.2 Iteration Based Step Sizes

Whilst the step size considered in section 7.2.1 depends on both the objective value and the subgradient vector, sometimes a step size based solely on the iteration is used, as described in section 5.2.1. One benefit of these is their simplicity to calculate; the previous step size requires an upper bound value of the Lagrangian dual, which must therefore be calculated in advance. Whilst this is possible, the performance of iteration based step sizes may be sufficient to remove the need for calculating an upper bound.

Step Size $\delta = \rho^i$

Section 5.2.1 introduces the iteration based step sizes included in this work, one of which was $\delta = \rho^i$, where $0 < \rho < 1$ and i is the iteration number. In this case, the selection of the value of ρ can have a dramatic impact on the performance of the subgradient algorithm, and even its ability to converge to the correct value. In fig. 7.1 we see the performance of the subgradient algorithm with this step size and four different values of ρ.

These tests were carried out with $\lambda_0 = \underline{0}$, so we observe the same behaviour in iteration 0 as described in section 5.2.2. As with the other step sizes, we see that the objective value drops after the first iteration, but this drop is smaller in magnitude for smaller

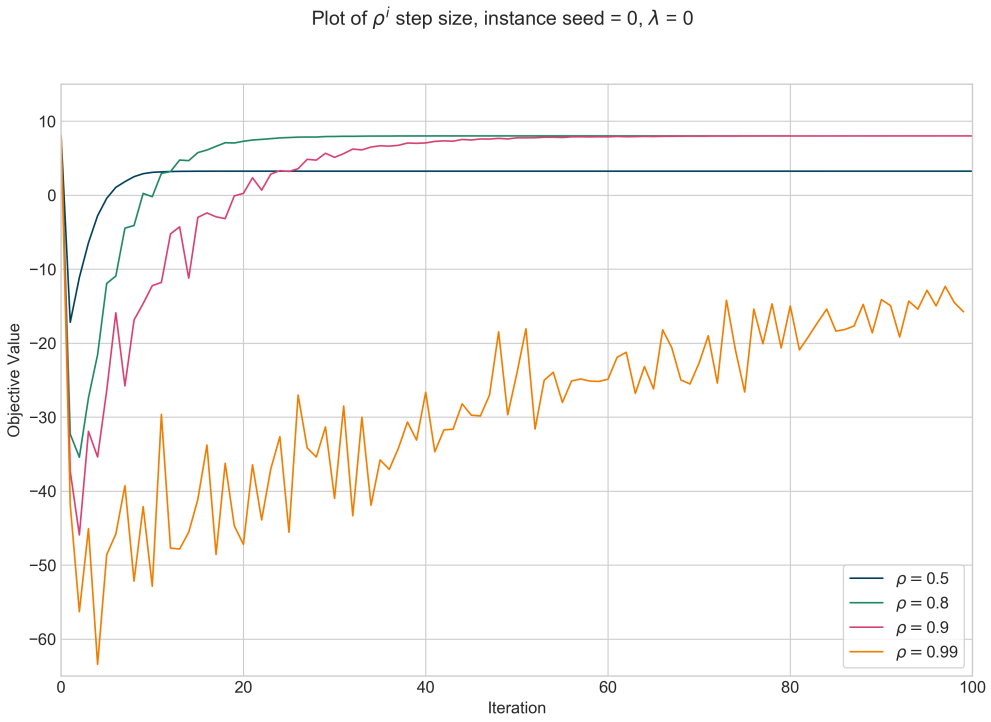


Figure 7.1: Plot for instance size $|C| = 4, |H| = 2, |J| = 8, |K| = 4, |W| = 2, |S| = 2, |L| = 2$

values of ρ. Whilst a smaller drop in the objective is desirable as it is less far for the subgradient method to climb back towards its convergence value, decreasing ρ does have a negative effect on the performance of the algorithm.

Looking at the line relating to $\rho = 0.5$, we see that it converges prematurely to 3.232, instead of 8 as happens for $\rho = 0.8$ and $\rho = 0.9$. This is not surprising when we consider the numerical implications of selecting a relatively small ρ; in such cases the step size will converge to 0 very quickly, meaning the subgradient can no longer find any other solution that is a meaningful distance away, and it will thus converge before it has a chance of reaching its optimal value. Conversely, considering very large values of ρ, like 0.99, we observe a line which is much more erratic, as the step size is very slow to decrease in value and the resulting subgradient method can make very large 'steps' in the solution space. The key here is to achieve a balance between finding a value of ρ which will allow quick convergence, without causing premature convergence; such values for this case include 0.8 and 0.9, as can be seen in the graph.

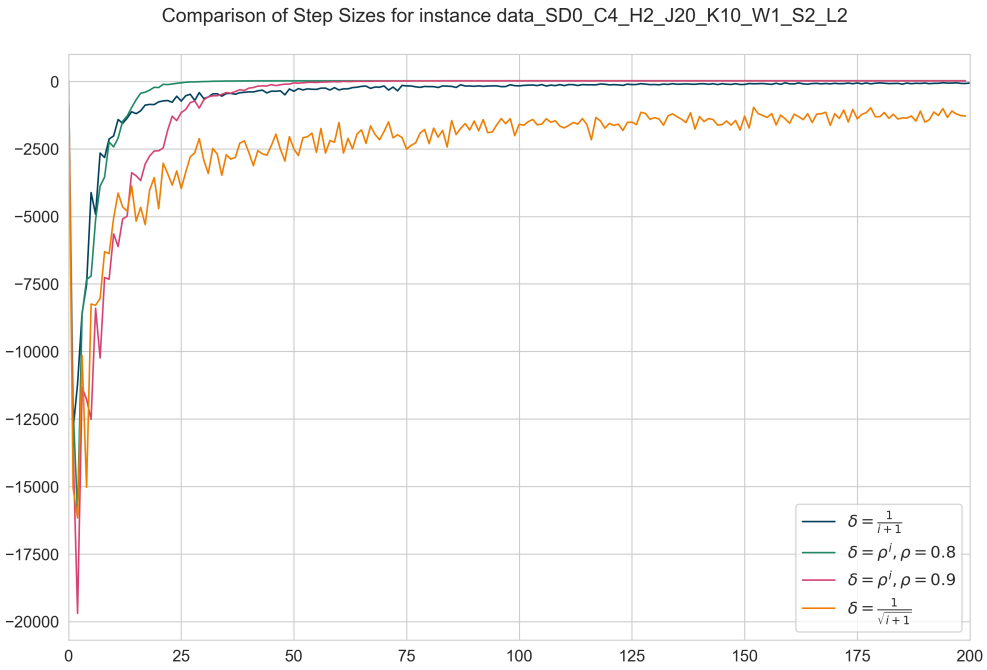


Figure 7.2: Comparison of subgradient step size

Comparison of Step Sizes

In total, three iteration based step sizes were tested: $\delta = \rho^i$, as seen in section 7.2.2, with $\rho = 0.8$ and 0.9; $\delta = \frac{1}{i+1}$ and $\delta = \frac{1}{\sqrt{i+1}}$, where i is counted from zero. Figure 7.2 shows the change in objective function value for each of these step sizes over 200 subgradient iterations, when tested on a problem of size $|J| = 20, |H| = 2, |K| = 10, |S| = 2, |L| = 2, |W| = 1$.

Here we see that all three step sizes follow a similar pattern, but with variation in the speed at which they converge and their erraticism. $\delta = \rho^i, \rho = 0.8$ converges to 20 in around 40 iterations, increasing from an objective value of less than -12,000 in the second iteration, whereas the two fractional step sizes experience the same dramatic drop at the beginning and both tend upwards from there, but neither reach convergence in the first 200 iterations shown in the graph. From this we see clearly that the best performing step size for this problem so far is $\delta = \rho^i, \rho = 0.8$, but further work is required to try and find greater improvements in its performance. The behaviour we observe in the second iteration, as explained in section 5.2.2, means unnecessary time is being spent in trying

Table 7.9: Number of iterations required to reach objective within 1 of first iteration

	$\delta = \frac{1}{i+1}$	$\delta = \frac{1}{\sqrt{i+1}}$	$\delta = \rho^i, \rho = 0.9$	$\delta = \rho^i, \rho = 0.8$
No normalisation	-	-	92	43
$\|\|X\|\|_2$	148	-	47	23
$\|\|X\|\|_1$	35	-	33	16

to return the objective value back towards positive numbers, as we know the optimal solution must be positive due to nonnegativity of decision variables and associated costs in the objective function.

7.2.3 Subgradient Normalisation

In order to try and reduce the magnitude of the decrease in the objective observed in the subgradient algorithm, we tested normalisations of each of the three step sizes compared in the previous section. As described in section 5.2.2, the first iteration of the subgradient method (or second if we start with zero value multipliers) has a very negative objective value. This is due to the high number of dualised constraints, and the guarantee that at least some of these will be feasible, and as such their penalties in the objective are cumulatively much larger than the original terms combined. As a result of this, the number of iterations required for the subgradient method to return to positive objective values, and therefore near to its final convergence value, is unnecessarily high.

We tested two different normalisations of the subgradient vector in combination with each step size. The normalisations tested were $\frac{G_j}{N}$, where N takes two different values:

$$||X||_1 = \sum_i x_i$$
$$||X||_2 = \sqrt{\sum_i x_i^2}$$

Figure 7.3 shows the objective values for each of the step sizes and their respective normalisations across 200 iterations. It is important to note that the objective value axes in fig. 7.3 is a log scale in order to see the behaviour of the normalised step sizes,

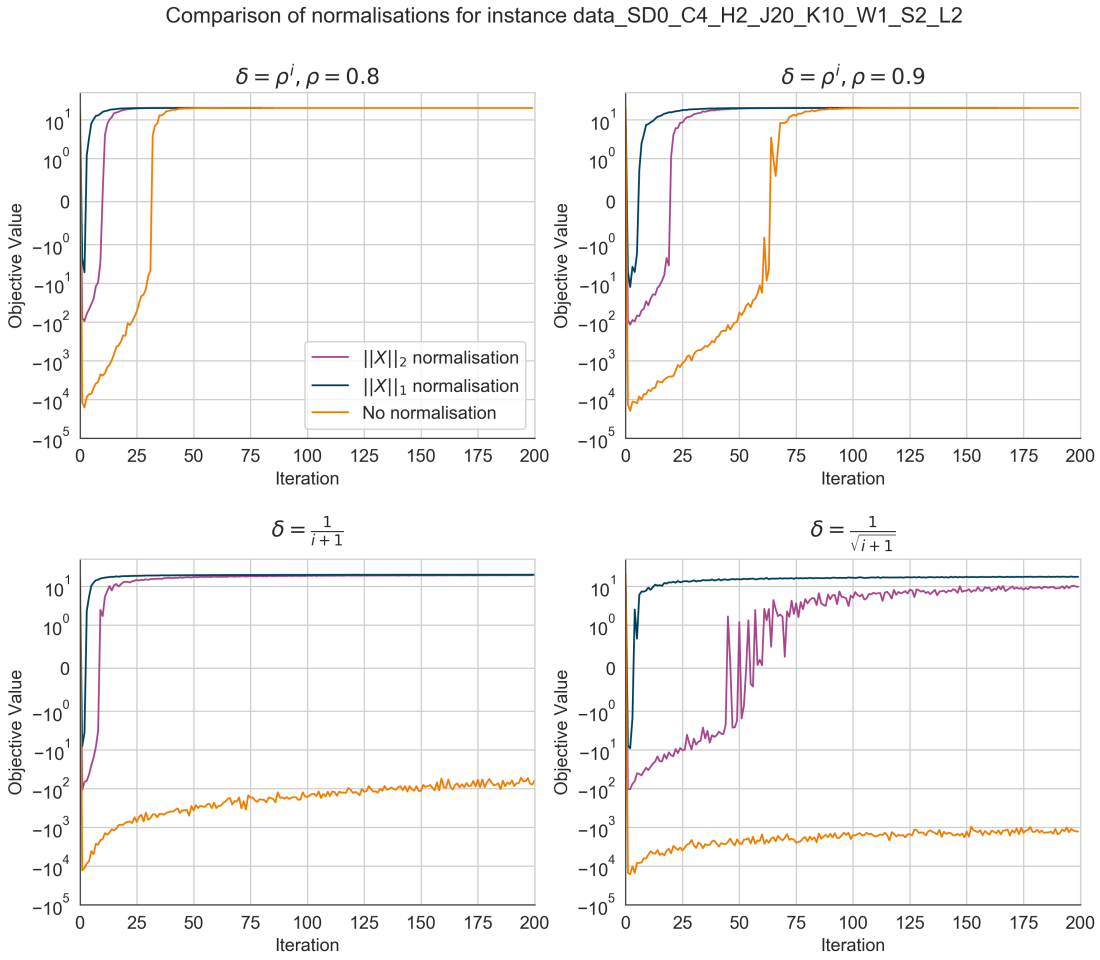


Figure 7.3: Comparison of subgradient normalisation by step size

so the changes are in fact larger than they appear initially. For all step sizes we see that both normalisations perform considerably better than their standard counterparts, with a much larger improvement for those step sizes which did not previously converge within 200 iterations ($\frac{1}{i+1}$ and $\frac{1}{\sqrt{i+1}}$). All four step sizes have a minimum objective value in the region of $-10,000$ when not normalised, but this minimum improves dramatically, to around -100 and -10 for $||X||_2$ and $||X||_1$ normalisations respectively. As hoped, this also has an impact on the number of iterations required to reach convergence. Table 7.9 shows the number of iterations required to reach an objective value within 1 of that which was achieved in the first iteration.

n.b. although convergence is normally considered to have been reached when the gaps between iterations are much smaller than 1, the integer value of all costs and weights in the objective function in these instances means objective values of the original problem will always be integer, so objective values from the subgradient can always be rounded up to achieve a lower bound to the problem, hence why one is used as the minimum difference for convergence in this case.

From fig. 7.3 we can see that all four step sizes converge much more quickly when normalised, but table 7.9 shows that even with normalisation $\frac{1}{i+1}$ still doesn't achieve convergence in 200 iterations. All the others do however, and all three perform best with the $||X||_1$ normalisation, with the number of iterations ranging from 16 to 35.

Following all of this analysis to identify the best step size and normalisation, we have the following final calculation of the lagrangian multipliers for the subgradient method:

$$\lambda_{j,i+1} = \max\left(0, \lambda_{j,i} + \rho^i \frac{G_j}{\sum_j G_j}\right) \qquad j = 1, \ldots, m \tag{7.6}$$

where $\lambda_{j,i}$ is the lagrangian mutliplier for constraint j at iteration i, and $\rho = 0.8$. Interestingly, of the step sizes tested this is the only one which does not satisfy either of the conditions for guaranteed convergence stated in section 5.2.1. These results clearly demonstrate that such conditions are not necessary for convergence to occur in some

cases, it simply means that this particular step size is not guaranteed to converge for all problems. In such cases, it is recommended to try one of the other step sizes tested in this work.

7.2.4 First Iteration Method

In this section we discuss the performance of the subgradient method in terms of the solution time. As has been shown in section 5.2.2, the best solution for the subgradient method is found in the first iteration, so in reality this is a test of the time taken to solve a single instance of the MILP problem with the dualised constraints removed. This was tested on 11 different instance sizes: $|H| = 4, |K| = 10, |S| = 2, |L| = 2, |W| = 1$, with $|J|$ varying between 20 and 40 in increments of two (even numbers considered to increase the range of sizes tested without having to test so many instances). This variation captures the part of the problem that is most likely to change from week to week: in a given division, the number of technicians and their skills will remain largely unchanged, but the number of jobs can vary greatly. The computational details of these experiments can be seen in table 7.10.

Table 7.10: Computational details for section 7.2.4

Software	Python (3.6.8)
Package	PuLP (1.6.9)
Solver	CPLEX (12.5.0.0)
Computer	Iridis 4
Processors	6 x 2.6GHz

At this stage, two time periods were measured: the total time for the Python script to run, and the time taken for CPLEX to solve the MILP. By consequence, the difference between these two measurements can be considered as the time taken to perform all the remaining actions in the script, which consist of building the model, reading the data file and saving the solution. All timings in this section will either represent CPLEX solution time or the combined time for all other actions. Due to the nature in which these times were recorded, it is not possible to provide a further breakdown between these steps. Figure 7.4 shows the average time taken for the two different stages across

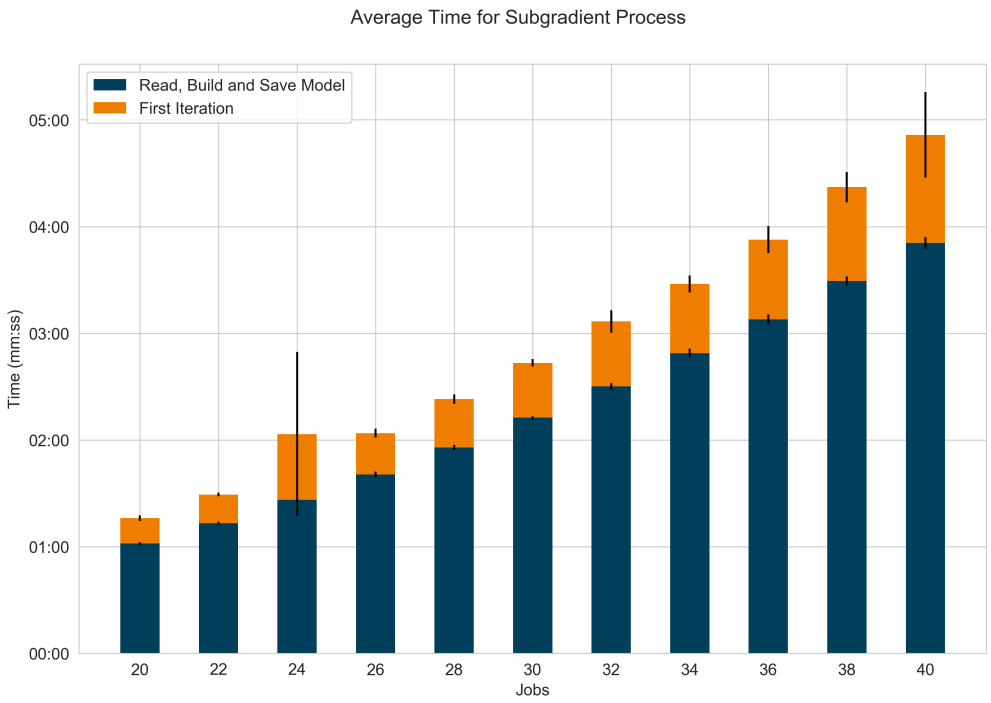


Figure 7.4: Time to find initial infeasible solution across varying problem sizes

Table 7.11: Computational details for section 7.3

Software	Python (3.7.4)
Computer	2.6GHz, 8GB RAM
Processors	2

different instance sizes.

Here we see clearly that the majority of the time across all instance sizes is not spent on solving the problem, which has an average time of 34.76s across all instances tested, but instead on all the other actions performed. Despite this, the full solution time for each of these problems is still reasonable for a problem that will need to be solved on a regular basis, but further testing would be required to ensure this still applies in a real-life setting, in which access to high powered computers or commercial solvers may be limited.

7.3 Heuristic Results

Here we present the results of the heuristic method described in section 5.3, which includes both the phase for making the initial solution feasible, and the local search heuristic for improving the solution. As in section 7.2.4, this method was tested on 11 different instance sizes: $|H| = 4, |K| = 10, |S| = 2, |L| = 2, |W| = 1$, with $|J|$ varying between 20 and 40 in increments of two. Each instance size was tested on 10 different datasets, with random seeds ranging from 0 to 9, meaning a total of 110 problems were tested to produce the following results. Computational details can be seen in table 7.11.

As the initial solution for this stage is taken from the first iteration of the subgradient method, it is not necessarily feasible for the full problem formulation. Figure 7.5 shows one example of what this solution looks like before any changes have been made. It is clear that several technicians have overlapping jobs, with most jobs having a start time of 1, both of which violate the constraints that were removed from the lagrangian dual formulation. The improvement process from this initital solution is made up of three distinct stages, which are as follows:

1. Make solution feasible (example in fig. 7.6)
2. Move jobs within tours (example in fig. 7.7)
3. Move late jobs between tours (example in fig. 7.8)

Figure 7.9 shows the average objective function value and maximum completion time after these first three stages of the full heuristic method for each problem size. Stages 2 and 3 are those which make up the local search heuristic, and are repeated until no improvement is found or the maximum number of iterations is reached; as such, fig. 7.9 only shows the change in objective for the first full iteration of the heuristic. Despite this, we still observe a significant decrease across these three stages, particularly in the objective value. Moreover, this improvement becomes even greater as the problem size increases, with the average objective value halving after the first stage of the heuristic in problems with 40 jobs. Although the maximum completion time does not experience such dramatic improvements in these three stages, improvements are still observed, with

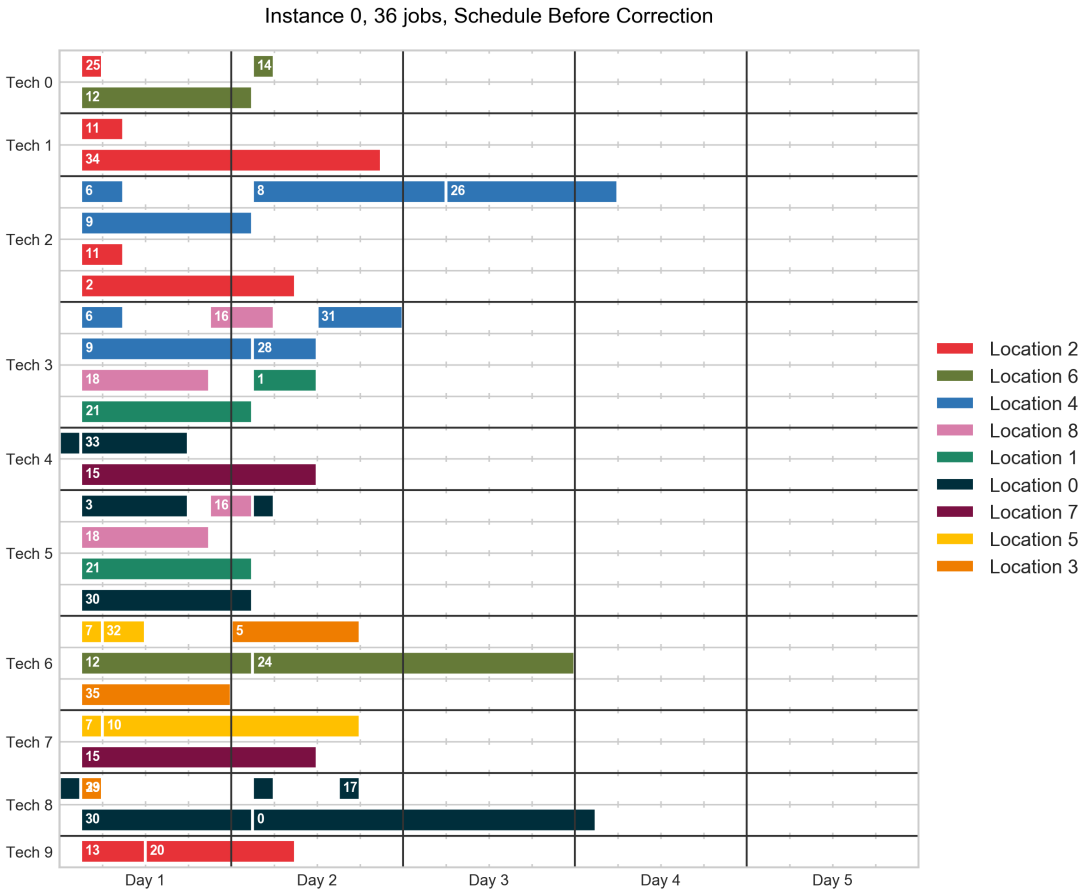
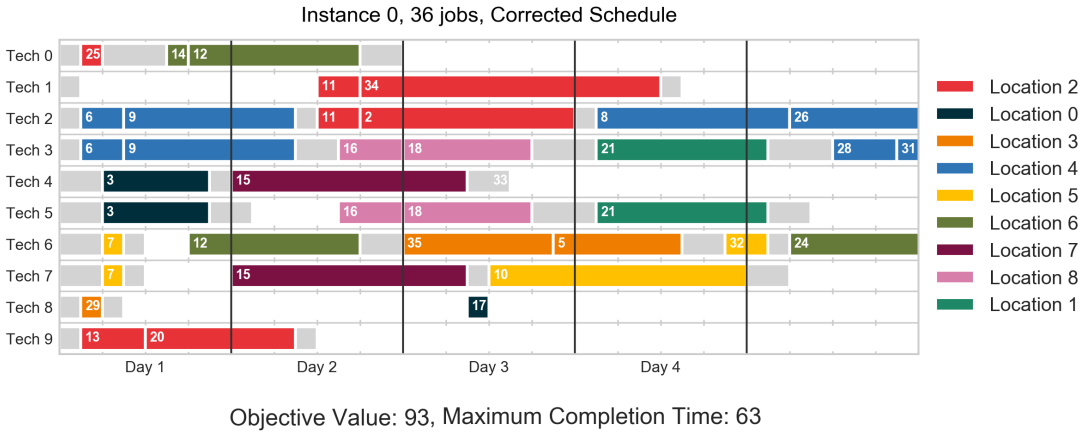


Figure 7.5: Infeasible result from Subgradient method



Objective Value: 93, Maximum Completion Time: 63

Figure 7.6: Schedule solution after making schedule feasible

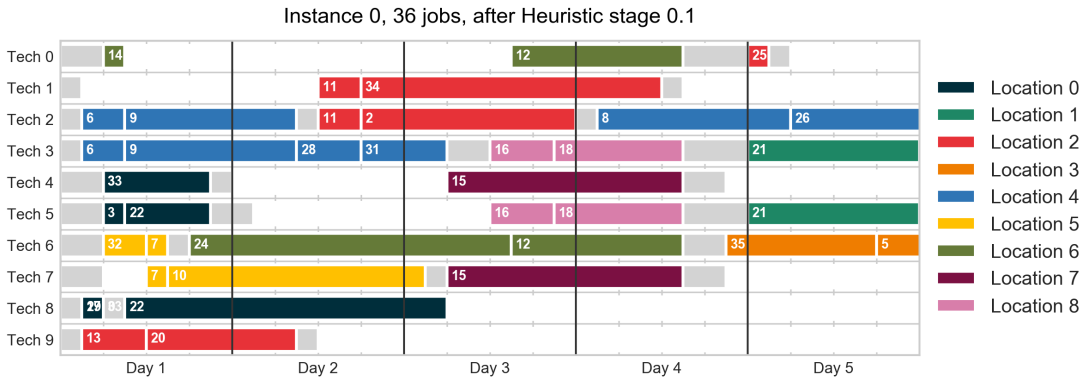


Figure 7.7: Schedule solution after second stage of heuristic, keeping all jobs with their original technician

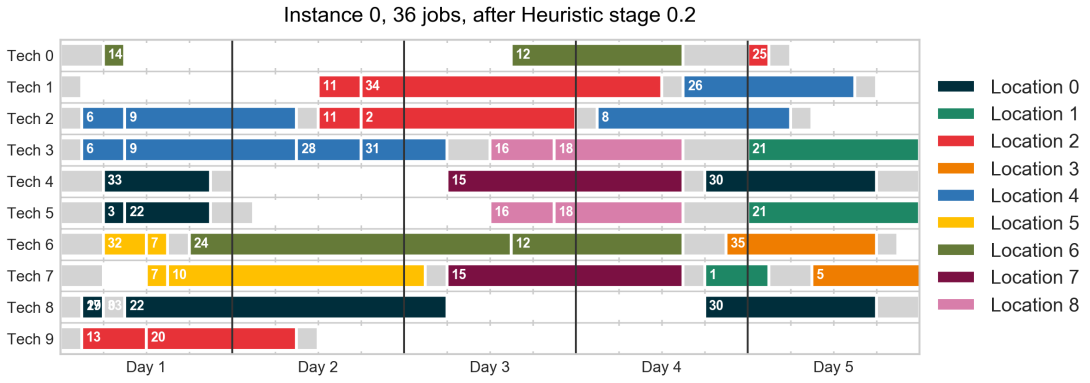


Figure 7.8: Schedule solution after third stage of heuristic moving late lobs to different technicians

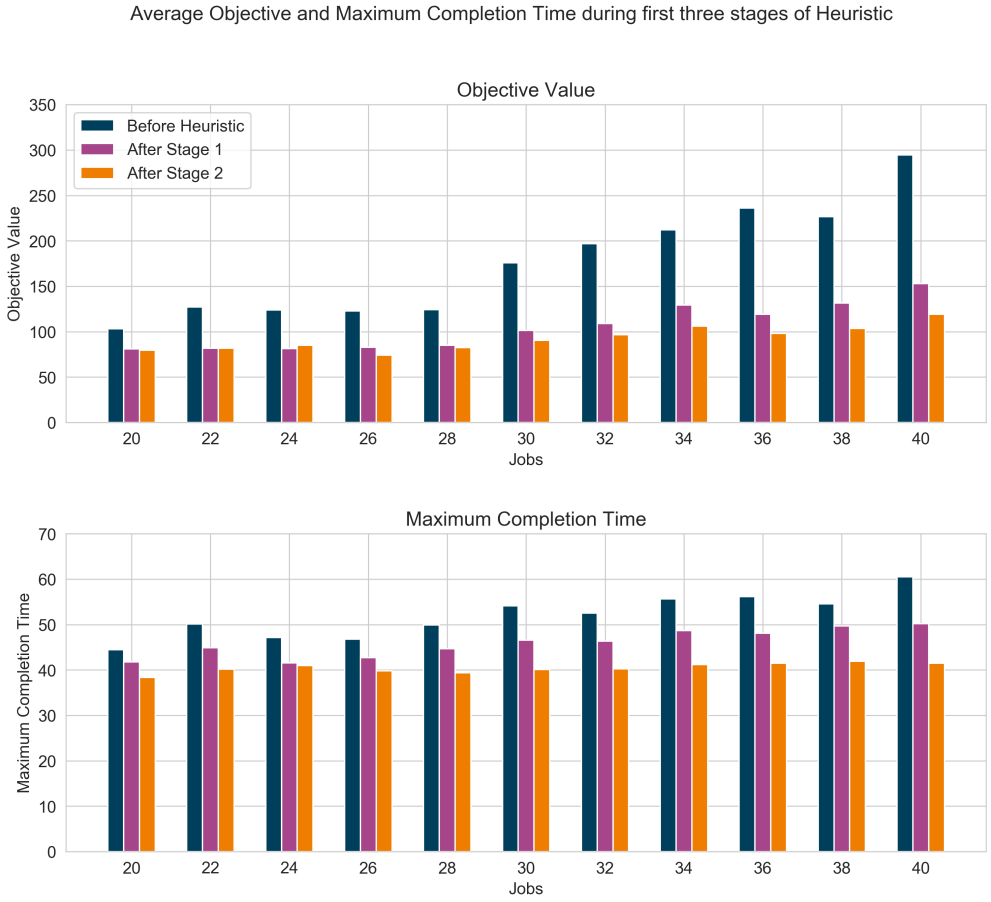


Figure 7.9: Average Objective Value and Maximum Completion Time across first three stages of Heuristic

the same trend of greater improvements for larger problem sizes.

This figure shows the first three stages of the solution process because these are the only stages that all problems are guaranteed to pass through; in some instances no improvement can be found after this point, whilst others continue for more iterations of stages 2 and 3. The maximum number of iterations of stages 2 and 3 reached by any of the instances tested was three, and the average across all instances tested is 1.57. Similarly, the maximum number of iterations reached within stage two is four. As both iterated parts of the heuristic meet their stopping criteria in just a handful of iterations across all instances tested, it is recommended to use a maximum of ten iterations for both parts, which is more than double the maximum observed in any instances tested in this work. See appendix A.2 for the full results of the heuristic.

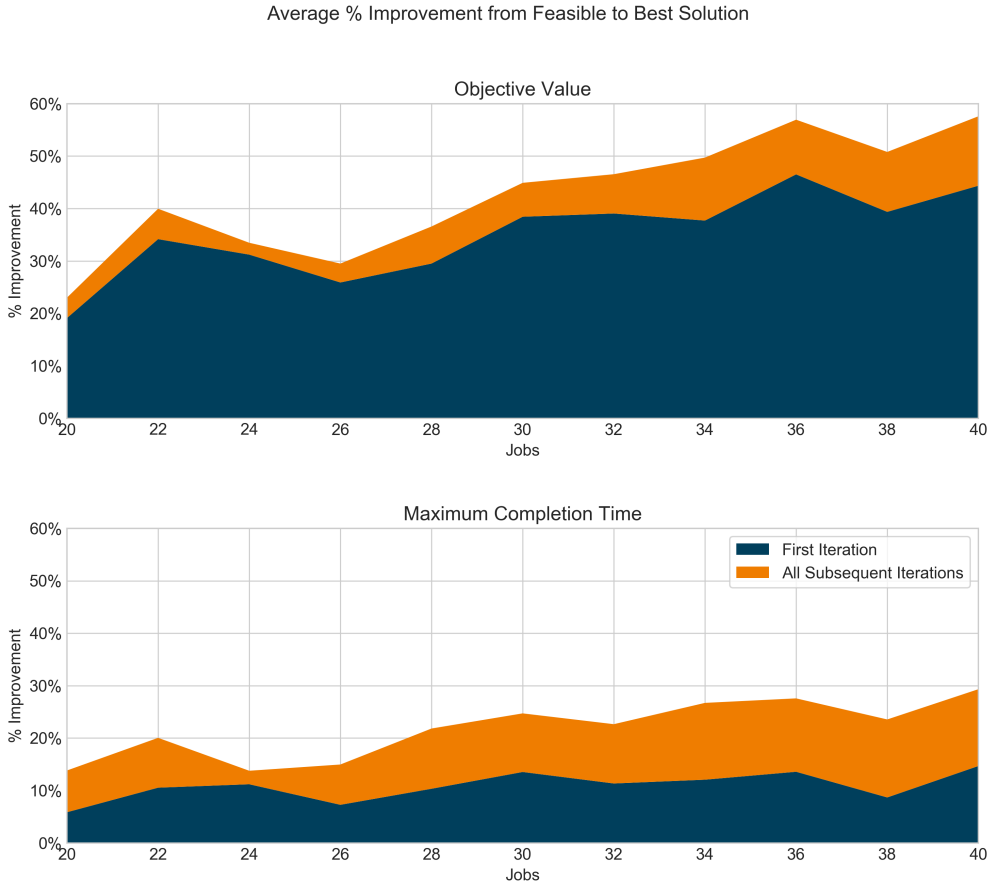


Figure 7.10: Average Improvement of Objective Value and Maximum Completion Time during Heuristic

Considering the improvement made from the initial solution to the final objective value and maximum completion time, fig. 7.10 shows how this improvement varies with the number of jobs. From this we see clearly that for the objective value, the biggest improvement comes in the first iteration for all problem sizes. The average improvement after the first iteration is 34.9%, and 42.5% overall, meaning the first iteration contributes 82.1% to the overall improvement of the objective function value. The improvement in maximum completion time is more balanced, with 10.8% coming from the first iteration, compared to 21.7% overall.

Beyond the improvement in the objective function that can be obtained using a heuristic, another important measure of performance is the time such heuristic methods take to reach a solution. Figure 7.11 shows the average time taken for each instance size for

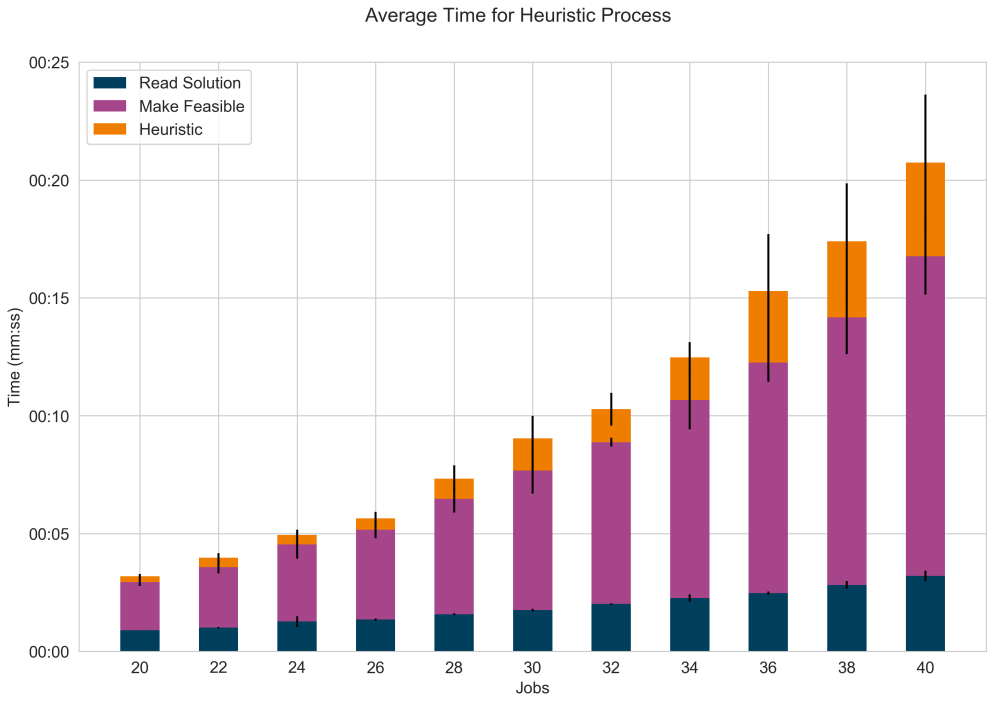


Figure 7.11: Time taken for heuristic method across varying problem sizes

three key parts of the heuristic process: reading results, making the solution feasible and applying the heuristic. As with the time taken to solve the first iteration of the subgradient method shown in section 7.2.4, here we observe a steady increase in the overall time as the number of jobs increases, but for all problem sizes the solution time is suitable for the intended application. It is also worth noting that the stage for reading the solution would not be necessary if the initial solution to the MILP and the heuristic were being performed on the same computer as part of a single process. In this case, initial solutions were found using the University's computing cluster because of the computational intensity of the full subgradient method, of which this was only one iteration. These results were then saved and transferred to a standard computer for the heuristic stage, resulting in writing and reading of results which could be avoided if all stages were implemented together.

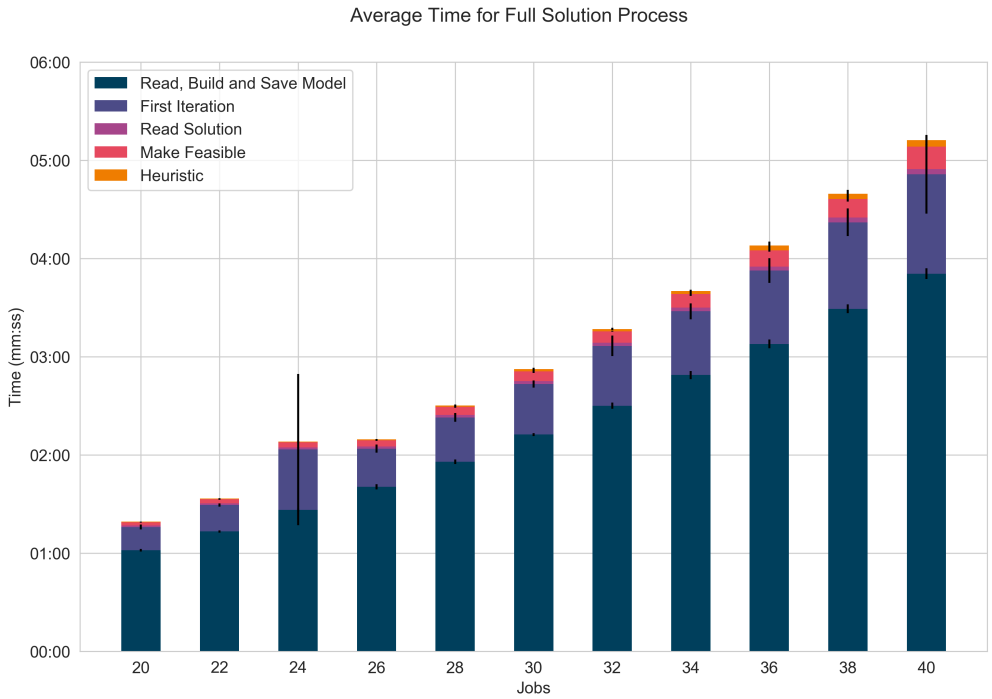


Figure 7.12: Time taken for full solution process across varying problem sizes

7.4 Full Solution Process

Combining the solution time for the results describes in sections 7.2.4 and 7.3, we obtain the total solution time from instance data to optimised schedule. Figure 7.12 shows the average time across all stages instance sizes, and a summary is also provided in table 7.12. A full table of results can be seen in appendix A.2.

When combining all stages of the full solution process, it becomes clear that the large

Table 7.12: Average time for each part of the solution method

Stage	**Average Time (mm:ss)**
Build Model and Save Solutions	02:17.9
Subgradient First Iteration	00:34.8
Read Solutions	00:01.9
Make Solution Feasible	00:06.6
Heuristic	00:01.6
Total	03:02.7

majority of the time is spent in building and solving the MILP model: across all the instance sizes an average of 94.86% of solution time was spent in these two stages. In addition, these are the two where the most uncertainly lies, as the computing power used would not necessarily be available in a situation where such a method was being implemented. As such, any further work to investigate and reduce computation times should be focussed on these stages.

We now use these results to make predictions about the total solution time for increased problem sizes in order to gain an understanding of the scalability of the final method. For each element of the total solution time, we produce a cubic line of best fit (found to be the most appropriate through testing) of the form

$$t = c_3|J|^3 + c_2|J|^2 + c_1|J| + c_0$$

Where t is the solution time in seconds and $|J|$ the number of jobs in the problem instance. These lines and their associated predictions can be seen in fig. 7.13, and the line equations can be seen in table 7.13. The error of these predictions against the known solution times is low, particularly for that which makes up the biggest proportion of the time, building the model, with an error of 0.16%. The highest error is seen when predicted the time for the heuristic, at 11.22%, but as this makes up the smallest proportion of the total time, its contribution to the total error is still very small. Whilst we can not be certain that the solution time for increasing problem sizes will follow the trend identified by extrapolating from the tested sizes, it suggests that these larger instances will still be solvable in a relatively quick time.

As these results have shown, the proposed method can solve instances up to a size of $|J| = 40, |H| = 4, |K| = 10, |S| = 2, |L| = 2, |W| = 1$ in a matter of minutes. Although this was tested on fabricated data, the size of the problem and the methods used to generate data mean it is representative of the RNLI problem at a divisional level, and as such we are confident that it would be able to solve such problems given real life data in a reasonable time. These short computation times would allow the algorithm to be rerun where necessary to incorporate unplanned jobs, with little impact on the work of

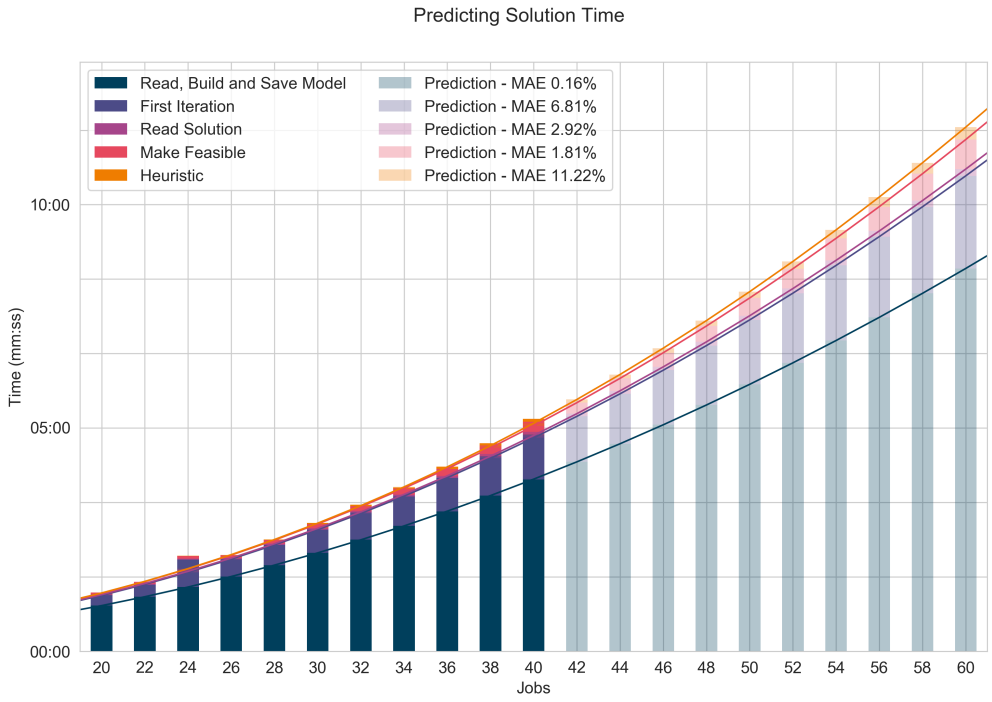


Figure 7.13: Predicted solution time for increased problem sizes

Table 7.13: Coefficients of line equations for computation time

	c_0	c_1	c_2	c_3	**Mean Abolsute Percentage Error**
Read, Build and Save Model	4.7723	0.0112	0.1414	0.0000	0.16%
Subgradient First Iteration	-0.4998	0.0109	0.0351	0.0000	6.81%
Read Solution	0.6597	0.0000	0.0001	0.0000	2.92%
Make Solution Feasible	0.0000	0.0000	0.0021	0.0001	1.81%
Heuristic	0.5113	0.0000	-0.0032	0.0001	11.22%

those responsible for the ST's routes.

Chapter 8

Conclusion and Further Work

8.1 Conclusion

In this thesis, we have presented the Vehicle Routing Problem with Time Windows, Skill Levels, and Synchonisation, an extension of the traditional VRP. As with most work published around the VRP this was completed with a particular application in mind, in this case the maintenance of lifeboats operated by the RNLI, the UK's largest coastal lifesaving charity. As a prevalent part of Operational Research, the existing literature on the VRP is very wide ranging, covering a number of different extensions and adaptations, some of which are relevant to the full problem of technician routing in the RNLI. Those extensions which are most relevant in this case became the focus of the literature review, where we presented a comprehensive overview of existing publications and the different combinations of extensions presented in each case. In addition, we considered the dominating techniques used for VRP problems as a basis for beginning the process of developing a solution method for the given problem.

We have proposed a matheuristic algorithm, comprised of the subgradient method, branch and bound and a local search heuristic, designed specifically for solving the VRP variant presented in chapter 4 which incorporates time windows, skill levels and synchronisation of technicians, a combination of VRP variants that is not tackled in the existing literature in this way. Through extensive testing of the Lagrangian dual

problem, we were able to prove that a sufficient lower bound to the full problem can be obtained through solving just one iteration of the subgradient method, with the solution at this stage providing a starting point for the subsequent local search heuristic. Results from the second stage showed that the heuristic presented provides strong improvements for all instances tested, with an average reduction in the objective value of 42.5%. This improvement is magnified for larger problem sizes, with the largest reduction of 57% for instances with 40 jobs.

One of the key considerations when developing this formulation and its associated matheuristic algorithm was creating something that could be solved quickly and on an ad-hoc basis; when considering the application in lifeboat maintenance, we have scenarios in which the jobs to be completed in a given week may change at short notice, and as such those managing the technician schedules would need to be able to regenerate a schedule for the week at short notice, and in a short amount of time. As shown in chapter 7, the algorithm was tested on instances of a varying sizes (by number of jobs) and the average solution time for the largest instances tested was just over five minutes, with an overall average of 182.7s. With solutions times such as these, this algorithm could be implemented into the working practices of Divisional Maintenance Managers (DMMs) at the RNLI and it would be able to produce technician work routes and schedules in a quick time, both at the initial planning stage or when more corrective jobs emerge.

With all of this in mind, we have addressed the original research question of this thesis. We have presented a formulation which captures the most important problem requirements, a combination not presented in the existing literature, and developed a matheuristic algorithm which achieves feasible solutions in a time suitable for the given application. If implemented, this solution approach would allow DMMs at the RNLI to free up time that has previously been spent on the task of manually scheduling technicians and therefore allow this time to be spent on other vital tasks. Creating efficiencies such as these is crucial in all organisations, but it is particularly important for an organisation which relies entirely on public donations to ensure resources are being used in the best possible way.

8.2 Further Work

As is almost always the case with research such as this, although the original questions have been answered there is more work that could be done to improve the output and its usability within the RNLI.

From a mathematical perspective, further work on the hierarchical approach mentioned in section 5.4 to break down the problem would create a solution method which could take data for a much larger problem, and provide a seamless transition between assigning work to weeks and optimising for those weeks individually, instead of relying on the manual assignment of work to weeks as is the case now. By ensuring that the assignment of jobs to weeks is also optimal (or at least optimised to some degree), we could also expect to see improvement in the objective values of each week, although to what degree would depend on the construction of the higher level problem. Developing a hierarchical approach may also allow for additional constraints that were not incorporated into the lower level formulation to be considered, such as reducing return visits to locations by minimising the number of different job locations assigned to each week. These improvements to the mathematical formulation would also be beneficial to stakeholders, as it would further reduce the number of manual task required of them by automating an extra stage of the decision making process.

The scope of this work was to formulate and solve the problem of ST routing in the RNLI, which has been addressed. There is however a disparity between the current process to use this method, and the technical knowledge of those who would be using it. More work is required in order to make this method usable to the RNLI, in particular the development of a user interface or integration with existing software so it can be used by the DMMs who currently create routes and schedules for technicians.

Appendix A

Appendices

A.1 Job Data

Table A.1: % of jobs/work hours by job type

	% of Jobs	% of Work Hours
Corrective	34.01%	21.71%
Preventative	65.99%	78.29%

Table A.2: Number of jobs by priority

Number of Jobs									
		Priority 1		**Priority 2**		**Priority 3**		**Priority 4**	
	Total	#	%	#	%	#	%	#	%
Corrective	4721	447	9.47%	1315	27.85%	2541	53.82%	418	8.85%
Preventative	9322	26	0.28%	1069	11.47%	8090	86.78%	137	1.47%
All Jobs	14043	473	3.37%	2384	16.98%	10631	75.70%	555	3.95%

Table A.3: Work hours jobs by priority

Work Hours									
		Priority 1		**Priority 2**		**Priority 3**		**Priority 4**	
	Total	#	%	#	%	#	%	#	%
Corrective	13945.5	1660	11.90%	3846.55	27.58%	7172.45	51.34%	1266.5	9.08%
Preventative	50987.5	140.5	0.28%	5211.55	10.22%	45211.45	88.67%	424	0.83%
All Jobs	64933	1800.5	2.77%	9058.1	13.95%	52383.9	80.67%	1690.5	2.60%

Table A.4: Descriptive Statistics of Jobs (all units in hours). Numbers in brackets indicate figures for when durations of value 0 are ignored.

Priority	**Count**	**Avg Duration**	**St Dev of Duration**	**Min Duration**	**Max Duration**	**# Jobs > 1 Week**
1	473 (328)	3.81 (5,49)	4.93 (5.08)	0.00 (0.50)	37.00	0
2	2384 (1684)	3.80 (5.38)	6.64 (7.35)	0.00 (0.30)	80.00	16
3	10631 (8375)	4.94 (6.25)	7.19 (7.57)	0.00 (0.20)	120.00	38
4	555 (279)	3.05 (6.06)	5.57 (6.60)	0.00 (0.50)	60.00	3
All Jobs	14043 (10666)	4.62 (6.09)	7.00 (7.46)	0.00 (0.50)	120.00	57

Table A.5: Number of jobs created per month

Month	Corrective	Preventative	Total	Corrective (without 0 duration)	Preventative (without 0 duration)	Total (without 0 duration)
January	542	3392	3934	295	2712	3007
February	311	258	569	225	182	407
March	412	538	950	312	490	802
April	340	858	1198	270	779	1049
May	383	575	958	269	503	772
June	539	980	1519	378	745	1123
July	590	476	1066	352	335	687
August	532	384	916	314	269	583
September	474	685	1159	264	586	850
October	366	563	929	192	473	665
November	325	632	957	192	567	759
Total	4814	9341	14155	3063	7641	10704

A.2 Heuristic Results

The values in the Heuristic Iteration column follow the format $a.b(.c)$, where a is the number of the iteration for the whole heuristic process, b indicates the stage within the heuristic (stage one being moving jobs within tours, and two moving jobs between tours), and c is the iteration number within the first part of the heuristic. The objective function and maximum completion time after each iteration is shown in the last two columns.

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
1	20	26	00:13.9	01:15.6	00:00.9	00:02.1	83	44	00:00.2	0.1.1	79	43
										0.2	73	40
										1.1.0	73	40
1	22	26	00:16.0	01:29.0	00:01.0	00:03.0	157	55	00:00.4	0.1.1	87	51
										0.2	68	40
										1.1.0	68	40
1	24	26	00:18.6	01:44.6	00:01.2	00:03.0	91	45	00:00.4	0.1.2	55	39
1	26	28	00:21.3	02:01.3	00:01.4	00:04.2	88	45	00:00.6	0.1.1	76	44
										0.2	72	38
										1.1.0	72	38
1	28	26	00:25.2	02:22.0	00:01.6	00:04.0	76	46	00:00.2	0.1.1	74	46
										0.2	58	37
										1.1.0	58	36
1	30	26	00:32.1	02:43.9	00:01.8	00:05.4	133	53	00:00.6	0.1.1	70	47
										0.2	56	39
										1.1.0	56	39
1	32	24	00:32.6	03:01.6	00:02.0	00:06.8	184	52	00:01.8	0.1.2	100	44

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
										0.2	98	41
										1.1.1	96	40
1	34	24	00:50.0	03:38.0	00:02.2	00:07.0	172	51	00:01.0	0.1.1	97	46
										0.2	84	42
										1.1.0	84	42
										1.2	84	42
1	36	22	00:41.7	03:48.1	00:02.5	00:11.2	173	52	00:01.7	0.1.1	88	48
										0.2	75	43
										1.1.1	73	42
										1.2	73	42
1	38	26	01:03.1	04:31.3	00:02.8	00:10.7	142	50	00:01.9	0.1.2	84	43
										0.2	75	41
										1.1.1	74	41
										1.2	74	41
1	40	22	00:48.7	04:36.9	00:03.1	00:14.0	226	53	00:02.4	0.1.2	135	50
										0.2	112	42
										1.1.1	110	42
										1.2	110	42
2	20	27	00:16.1	01:18.4	00:00.9	00:01.8	74	40	00:00.1	0.1.1	64	36
2	22	24	00:18.6	01:32.5	00:01.0	00:02.0	51	41	00:00.1	0.1.1	44	38
2	24	27	00:30.9	01:58.1	00:01.2	00:02.9	75	42	00:00.1	0.1.2	65	40
2	26	26	00:23.5	02:07.3	00:01.3	00:04.0	84	44	00:00.2	0.1.2	62	43
										0.2	60	36
										1.1.0	60	36
2	28	24	00:26.2	02:22.2	00:01.6	00:05.0	106	53	00:00.5	0.1.1	80	50
										0.2	69	35

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
										1.1.1	66	35
2	30	24	00:28.2	02:40.2	00:01.8	00:07.0	185	58	00:01.3	0.1.3	113	46
										0.2	98	40
										1.1.0	98	40
2	32	24	00:33.0	03:06.5	00:02.0	00:06.8	176	51	00:00.5	0.1.1	87	45
										0.2	82	40
										1.1.0	82	40
2	34	24	00:43.5	03:35.6	00:02.2	00:09.6	194	58	00:02.1	0.1.3	126	49
										0.2	102	39
										1.1.0	102	39
2	36	26	00:48.5	04:00.6	00:02.5	00:08.7	169	49	00:01.7	0.1.2	110	44
										0.2	101	40
										1.1.0	101	40
2	38	24	00:45.9	04:19.2	00:03.0	00:13.0	152	47	00:03.6	0.1.2	111	44
										0.2	96	40
										1.1.1	95	41
										1.2	95	41
2	40	22	00:50.5	04:45.7	00:03.1	00:14.0	242	58	00:02.6	0.1.2	169	52
										0.2	112	43
										1.1.1	99	43
										1.2	99	43
3	20	27	00:13.0	01:14.8	00:00.9	00:02.1	84	43	00:00.3	0.1.1	71	42
										0.2	68	39
										1.1.0	68	39
3	22	26	00:15.8	01:28.9	00:01.0	00:02.7	125	52	00:00.4	0.1.2	71	42
										0.2	68	40

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
										1.1.2	65	40
3	24	26	00:18.4	01:45.2	00:01.2	00:03.1	155	51	00:00.6	0.1.2	95	43
										0.2	93	41
										1.1.0	93	41
										1.2	93	41
3	26	27	00:26.1	02:06.5	00:01.4	00:03.1	112	48	00:00.5	0.1.1	94	46
										0.2	83	40
										1.1.1	79	37
3	28	26	00:29.6	02:24.3	00:01.6	00:04.3	114	48	00:00.4	0.1.1	77	43
										0.2	71	40
										1.1.0	71	40
3	30	28	00:32.6	02:45.2	00:01.8	00:03.8	88	47	00:00.1	0.1.1	73	40
3	32	25	00:43.7	03:12.5	00:02.0	00:06.7	94	43	00:00.4	0.1.1	71	39
3	34	25	00:37.0	03:25.6	00:02.2	00:10.7	175	55	00:01.0	0.1.1	122	49
										0.2	98	43
										1.1.1	94	39
3	36	24	00:41.4	03:49.5	00:02.6	00:09.2	193	54	00:01.6	0.1.1	107	50
										0.2	78	40
										1.1.2	73	40
3	38	23	00:59.0	04:30.3	00:02.7	00:09.6	118	46	00:01.2	0.1.2	71	42
										0.2	68	40
										1.1.0	68	40
3	40	24	00:49.6	04:43.2	00:03.4	00:11.9	219	53	00:03.3	0.1.2	148	51
										0.2	127	40
										1.1.1	124	40
4	20	29	00:13.4	01:15.2	00:00.9	00:02.3	117	44	00:00.4	0.1.2	83	38

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
4	22	29	00:15.4	01:29.0	00:01.0	00:02.5	148	55	00:00.4	0.1.2	95	46
										0.2	87	39
										1.1.1	85	39
4	24	31	00:19.7	01:46.0	00:01.2	00:03.1	138	49	00:00.7	0.1.1	92	41
										0.2	87	39
										1.1.1	84	38
4	26	29	00:21.7	02:02.6	00:01.3	00:04.0	126	40	00:00.5	0.1.2	94	38
4	28	31	00:29.3	02:25.6	00:01.6	00:05.8	155	48	00:00.9	0.1.2	115	41
										0.2	115	41
4	30	28	00:31.8	02:44.1	00:01.7	00:06.2	196	60	00:02.4	0.1.3	113	47
										0.2	107	41
										1.1.0	107	41
										1.2	107	41
4	32	28	00:51.1	03:20.9	00:02.0	00:06.7	196	49	00:02.5	0.1.2	136	48
										0.2	117	41
										1.1.0	117	41
										1.2	116	41
										2.1.0	116	41
										2.2	116	41
4	34	28	00:35.3	03:23.1	00:02.7	00:07.9	176	48	00:01.5	0.1.1	131	45
										0.2	125	40
										1.1.0	125	40
4	36	30	00:39.1	03:46.2	00:02.5	00:09.9	282	55	00:03.5	0.1.2	166	50
										0.2	127	40
										1.1.2	120	38
4	38	29	00:42.6	04:09.9	00:02.7	00:10.6	261	52	00:02.0	0.1.2	152	47

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
										0.2	130	39
										1.1.2	128	40
5	20	26	00:13.2	01:14.9	00:00.9	00:01.9	90	44	00:00.2	0.1.1	79	43
										0.2	74	38
										1.1.0	74	38
5	22	24	00:15.5	01:28.7	00:01.0	00:02.6	93	46	00:00.2	0.1.2	60	40
5	24	26	00:18.6	01:44.8	00:01.2	00:03.0	87	42	00:00.1	0.1.1	79	42
										0.2	76	40
										1.1.0	76	39
5	26	24	00:22.8	02:03.8	00:01.4	00:03.4	82	45	00:00.3	0.1.3	52	41
										0.2	52	41
5	28	24	00:25.4	02:22.9	00:01.6	00:04.4	103	50	00:00.4	0.1.2	48	40
5	30	22	00:27.4	02:40.1	00:01.8	00:05.2	139	53	00:00.4	0.1.2	58	44
										0.2	56	39
										1.1.0	56	39
5	32	22	00:32.2	03:03.0	00:02.0	00:07.0	157	56	00:01.4	0.1.1	94	50
										0.2	74	39
										1.1.2	69	39
5	34	22	00:35.2	03:23.3	00:02.2	00:06.7	151	47	00:01.5	0.1.1	107	42
										0.2	100	40
										1.1.0	100	40
5	36	22	00:42.2	03:49.5	00:02.5	00:10.1	191	57	00:01.6	0.1.1	96	45
										0.2	91	45
										1.1.1	90	45
										1.2	88	40
										2.1.1	87	40

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
5	38	24	01:06.2	04:35.1	00:02.7	00:09.1	158	50	00:02.4	0.1.2	123	50
										0.2	105	44
										1.1.0	105	44
										1.2	103	39
										2.1.1	101	40
6	20	22	00:13.2	01:16.1	00:00.9	00:02.0	150	56	00:00.3	0.1.2	80	50
										0.2	60	35
										1.1.1	57	35
6	22	24	00:15.4	01:29.7	00:01.0	00:02.8	80	42	00:00.2	0.1.1	62	37
6	24	26	00:18.9	01:45.8	00:01.2	00:03.5	90	41	00:00.2	0.1.2	77	40
6	26	25	00:28.1	02:09.6	00:01.3	00:04.0	82	44	00:00.5	0.1.2	66	44
										0.2	64	42
										1.1.0	64	42
										1.2	64	42
6	28	26	00:27.2	02:24.7	00:01.6	00:05.2	114	49	00:00.6	0.1.2	78	43
										0.2	73	40
										1.1.1	71	39
6	30	26	00:33.2	02:47.8	00:01.8	00:07.1	101	43	00:00.7	0.1.1	89	43
										0.2	88	40
										1.1.0	88	40
6	32	24	00:36.5	03:08.3	00:02.1	00:06.7	115	50	00:00.6	0.1.1	81	44
										0.2	70	38
										1.1.1	67	38
6	34	24	00:41.4	03:32.6	00:02.3	00:07.5	247	58	00:01.9	0.1.3	103	45
										0.2	93	40
										1.1.1	92	40

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
6	36	26	02:53.7	06:03.4	00:02.4	00:08.6	49055	3116	00:02.4	0.1.3	112	47
										0.2	97	42
										1.1.1	95	42
										1.2	95	42
6	38	22	00:44.2	04:15.7	00:02.8	00:10.9	144	51	00:01.5	0.1.3	87	49
										0.2	75	42
										1.1.0	75	42
										1.2	73	41
										2.1.0	73	41
										2.2	73	41
6	40	25	01:54.4	05:47.4	00:03.0	00:11.0	153	51	00:00.9	0.1.1	86	46
										0.2	71	40
										1.1.0	71	40
7	20	27	00:13.6	01:16.2	00:00.9	00:02.3	118	43	00:00.3	0.1.1	105	43
										0.2	102	40
										1.1.0	102	40
7	22	26	00:16.0	01:29.9	00:01.0	00:02.7	130	49	00:00.7	0.1.1	85	46
										0.2	83	41
										1.1.2	81	41
										1.2	81	41
7	24	26	00:18.5	01:46.2	00:01.1	00:03.3	172	54	00:00.6	0.1.2	79	42
										0.2	79	42
7	28	25	00:32.5	02:26.6	00:01.5	00:04.8	119	52	00:01.7	0.1.2	90	47
										0.2	82	40
										1.1.2	79	40
7	30	26	00:32.7	02:45.8	00:01.7	00:06.3	162	50	00:02.1	0.1.3	117	45

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
										0.2	110	43
										1.1.1	109	43
										1.2	109	43
7	32	24	00:35.6	03:06.9	00:02.0	00:07.0	182	50	00:01.8	0.1.2	123	46
										0.2	116	42
										1.1.0	116	42
										1.2	116	42
7	34	22	00:36.5	03:28.3	00:02.2	00:08.9	326	71	00:03.1	0.1.3	180	58
										0.2	114	42
										1.1.1	108	42
										1.2	108	42
7	36	25	01:03.8	04:15.4	00:02.4	00:08.8	443	73	00:09.2	0.1.3	128	54
										0.2	100	43
										1.1.1	96	41
										1.2	95	40
										2.1.0	95	40
7	38	24	00:58.2	04:29.8	00:03.2	00:11.4	275	59	00:01.7	0.1.1	144	52
										0.2	110	40
										1.1.0	110	40
7	40	22	00:50.3	04:44.2	00:03.1	00:14.3	81386	2106	00:09.0	0.1.2	466	78
										0.2	305	56
										1.1.0	305	56
										1.2	305	56
8	20	24	00:14.1	01:15.0	00:00.9	00:02.0	83	39	00:00.1	0.1.2	68	37
8	22	24	00:16.5	01:29.3	00:01.1	00:02.4	138	50	00:00.6	0.1.2	69	44
										0.2	61	41

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
										1.1.1	60	41
										1.2	60	41
8	24	22	00:21.5	01:47.5	00:01.9	00:04.9	145	50	00:00.4	0.1.2	69	42
										0.2	69	42
8	26	22	00:22.4	02:01.7	00:01.3	00:04.1	96	41	00:00.2	0.1.1	84	38
8	28	20	00:24.4	02:20.6	00:01.5	00:05.7	129	48	00:01.3	0.1.2	101	46
										0.2	91	39
										1.1.0	91	39
8	30	20	00:28.1	02:40.4	00:01.7	00:06.5	260	65	00:03.0	0.1.2	141	58
										0.2	96	39
										1.1.1	95	43
										1.2	89	40
										2.1.0	89	40
8	32	20	00:32.3	03:01.4	00:02.0	00:06.8	180	51	00:02.0	0.1.2	123	45
										0.2	109	41
										1.1.0	109	41
										1.2	109	41
8	34	20	00:36.0	03:24.3	00:02.2	00:08.0	217	52	00:01.7	0.1.3	131	52
										0.2	111	44
										1.1.1	105	43
										1.2	100	40
										2.1.0	100	40
8	36	20	00:42.1	03:49.4	00:02.4	00:10.6	226	57	00:02.2	0.1.2	119	47
										0.2	93	41
										1.1.0	93	41
										1.2	93	41

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
8	38	20	00:44.6	04:12.1	00:02.7	00:12.6	406	66	00:09.1	0.1.4	196	57
										0.2	139	49
										1.1.1	137	48
										1.2	135	48
										2.1.0	135	48
										2.2	135	48
8	40	20	00:49.3	04:38.0	00:03.6	00:15.7	429	84	00:04.7	0.1.5	135	48
										0.2	116	39
										1.1.0	116	39
9	20	34	00:17.7	01:18.0	00:00.9	00:01.9	128	47	00:00.3	0.1.2	101	44
										0.2	100	38
										1.1.0	100	38
9	22	32	00:15.6	01:27.1	00:01.0	00:02.4	222	61	00:00.6	0.1.1	161	60
										0.2	124	40
										1.1.1	119	40
9	24	32	02:47.2	04:11.7	00:01.2	00:02.7	163	50	00:00.5	0.1.1	121	45
										0.2	105	42
										1.1.0	105	42
										1.2	105	42
9	26	30	00:20.2	01:58.1	00:01.4	00:03.7	311	67	00:01.1	0.1.1	133	48
										0.2	114	42
										1.1.0	114	42
										1.2	114	42
9	28	30	00:24.2	02:17.7	00:01.6	00:04.8	202	55	00:01.8	0.1.3	103	46
										0.2	101	43
										1.1.0	101	43

Continued on next page

Seed	Jobs	Subgradient 1st Iteration Objective	Subgradient 1st Iteration Time	Total Time	Read Time	Make Feasible Time	Feasible Objective	Feasible Maximum Completion Time	Heuristic Time	Heuristic Iteration	Iteration Objective	Iteration Maximum Completion Time
										1.2	101	40
9	30	31	00:31.6	02:42.8	00:01.7	00:05.7	317	58	00:01.8	0.1.1	138	49
										0.2	112	40
										1.1.1	109	40
9	32	26	00:31.8	02:58.7	00:02.0	00:07.3	487	71	00:01.6	0.1.1	164	56
										0.2	108	40
										1.1.1	103	40
9	34	26	00:35.4	03:19.1	00:02.2	00:09.3	249	61	00:02.5	0.1.2	167	52
										0.2	126	41
										1.1.2	109	40
9	36	27	00:39.8	03:42.8	00:02.4	00:09.7	211	52	00:02.8	0.1.2	139	47
										0.2	120	40
										1.1.1	114	40
9	38	27	00:52.0	04:16.2	00:02.8	00:14.2	382	70	00:05.7	0.1.2	215	63
										0.2	132	42
										1.1.1	129	42
										1.2	129	42
9	40	26	00:52.1	04:38.0	00:03.0	00:14.8	497	64	00:09.9	0.1.3	244	54
										0.2	176	45
										1.1.1	173	45
										1.2	173	45

Continued on next page

Bibliography

Sohaib Afifi, Duc-Cuong Dang, and Aziz Moukrim. Heuristic solutions for the vehicle routing problem with time windows and synchronized visits. *Optimization Letters*, 10 (3):511–525, 2016.

Guilherme Bastos Alvarenga, Geraldo Robson Mateus, and G De Tomi. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research*, 34(6):1561–1584, 2007.

Claudia Archetti, Natashia Boland, and M Grazia Speranza. A matheuristic for the multivehicle inventory routing problem. *INFORMS Journal on Computing*, 29(3): 377–387, 2017.

Athanassios N Avramidis, Wyean Chan, Michel Gendreau, Pierre L'ecuyer, and Ornella Pisacane. Optimizing daily agent scheduling in a multiskill call center. *European Journal of Operational Research*, 200(3):822–832, 2010.

Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202(3):756–763, 2010.

Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research*, 41:167–173, 2014.

Edward K Baker. Technical note—an exact algorithm for the time-constrained traveling salesman problem. *Operations Research*, 31(5):938–945, 1983.

Michel L Balinski and Richard E Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.

John E Beasley. *Advances in Linear and Integer Prgoramming*. Oxforf University Press, 1996.

Sachidanand V Begur, David M Miller, and Jerry R Weaver. An integrated spatial dss for scheduling and routing home-health-care nurses. *Interfaces*, 27(4):35–48, 1997.

Tolga Bektaş and Luis Gouveia. Requiem for the miller–tucker–zemlin subtour elimination constraints? *European Journal of Operational Research*, 236(3):820–832, 2014.

Onder Belgin, Ismail Karaoglan, and Fulya Altiparmak. Two-echelon vehicle routing problem with simultaneous pickup and delivery: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, 115:1–16, 2018.

Odile Bellenguez and Emmanuel Néron. Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 229–243. Springer, 2004.

Odile Bellenguez-Morineau and Emmanuel Néron. A branch-and-bound method for solving multi-skill project scheduling problem. *RAIRO-Operations Research*, 41(2): 155–170, 2007.

Sandjai Bhulai, Ger Koole, and Auke Pot. Simple methods for shift scheduling in multiskill call centers. *Manufacturing & Service Operations Management*, 10(3):411–420, 2008.

Fred Blakeley, Burçin Argüello, Buyang Cao, Wolfgang Hall, and Joseph Knolmajer. Optimizing periodic maintenance operations for schindler elevator corporation. *Interfaces*, 33(1):67–79, 2003.

Lawrence Bodin, Bruce Golden, Arjang Assad, and Michael Ball. Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Research*, 10(2):63–211, 1983.

Nathalie Bostel, Pierre Dejax, Pierre Guez, and Fabien Tricoire. *Multiperiod planning*

and routing on a rolling horizon for field force optimization logistics, pages 503–525. Springer, 2008.

Edward H Bowman. The schedule-sequencing problem. *Operations Research*, 7(5):621–624, 1959.

Julien Bramel and David Simchi-Levi. On the effectiveness of set covering formulations for the vehicle routing problem with time windows. *Operations Research*, 45(2):295–301, 1997.

Ulf Brännlund. A generalized subgradient method with relaxation step. *Mathematical Programming*, 71(2):207–219, 1995.

David Bredström and Mikael Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1):19–31, 2008.

Ulrich Breunig, Verena Schmid, Richard F Hartl, and Thibaut Vidal. A large neighbourhood based heuristic for two-echelon routing problems. *Computers & Operations Research*, 76:208–225, 2016.

Teobaldo Bulhões, Minh Hoang Ha, Rafael Martinelli, and Thibaut Vidal. The vehicle routing problem with service level constraints. *European Journal of Operational Research*, 265(2):544–558, 2018.

Paolo M Camerini, Luigi Fratta, and Francesco Maffioli. On improving relaxation methods by modified gradient techniques. In *Nondifferentiable Optimization*, pages 26–34. Springer, 1975.

Paola Cappanera, Luís Gouveia, and Maria Grazia Scutellà. The skill vehicle routing problem. In *Network Optimization: 5th International Conference, INOC 2011, Hamburg, Germany, June 13-16, 2011. Proceedings*, pages 354–364. Springer Berlin Heidelberg, 2011.

Massimiliano Caramia and Stefano Giordani. A new approach for scheduling independent tasks with multiple modes. *Journal of Heuristics*, 15(4):313–329, 2009.

I-Ming Chao. A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51, 2002.

Masoud Chitsaz, Jean-François Cordeau, and Raf Jans. A unified decomposition matheuristic for assembly, production, and inventory routing. *INFORMS Journal on Computing*, 31(1):134–152, 2019.

Transparent Choice. Transparent choice AHP software. URL `https://www.transparentchoice.com/ahp-software`.

Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

Stefano Coniglio, Jörg Fliege, and Ruth Walton. Facility location with item storage and delivery. In *International Conference on Optimization and Decision Science*, pages 287–294. Springer, 2017.

Jean-Francois Cordeau, Michel Gendreau, Gilbert Laporte, Jean-Yves Potvin, and Frédéric Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research society*, 53(5):512–522, 2002.

Jean-François Cordeau, Gilbert Laporte, Federico Pasin, and Stefan Ropke. Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13 (4):393–409, 2010.

George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

Mauro Dell'Amico, Giovanni Righini, and Matteo Salani. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation science*, 40(2):235–247, 2006.

Martin Desrochers and Gilbert Laporte. Improvements and extensions to the miller-

tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10(1): 27–36, 1991.

Jacques Desrosiers, François Soumis, and Martin Desrochers. Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984.

Michael Drexl. Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

Patrik Eveborn, Patrik Flisberg, and Mikael Rönnqvist. Laps care—an operational system for staff planning of home care. *European Journal of Operational Research*, 171(3):962–976, 2006.

Murat Fırat and CAJ Hurkens. An improved mip-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15(3):363–380, 2012.

Brian A Foster and David M Ryan. An integer programming approach to the vehicle scheduling problem. *Journal of the Operational Research Society*, 27(2):367–384, 1976.

Antonio Frangioni, Bernard Gendron, and Enrico Gorgone. On the computational efficiency of subgradient methods: a case study with lagrangian bounds. *Mathematical Programming Computation*, 9(4):573–604, 2017.

Ralph Gomory. An algorithm for the mixed integer problem. Technical report, RAND CORP SANTA MONICA CA, 1960.

Martin Grötschel and Manfred W Padberg. Partial linear characterizations of the asymmetric travelling salesman polytope. *Mathematical Programming*, 8(1):378–381, 1975.

Gabriel Gutiérrez-Jarpa, Guy Desaulniers, Gilbert Laporte, and Vladimir Marianov. A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, 206(2):341–349, 2010.

Michael Held, Philip Wolfe, and Harlan P Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, 1974.

Alessio Ishizaka and Philippe Nemery. *Multi-criteria decision analysis: methods and software*. John Wiley & Sons, 2013.

H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer Berlin Heidelberg, 2004. ISBN 9783540247777.

Çağrı Koç, Tolga Bektaş, Ola Jabali, and Gilbert Laporte. A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. *Computers & Operations Research*, 64:11–27, 2015.

Antoon WJ Kolen, AHG Rinnooy Kan, and Harry WJM Trienekens. Vehicle routing with time windows. *Operations Research*, 35(2):266–273, 1987.

Attila A Kovacs, Sophie N Parragh, Karl F Doerner, and Richard F Hartl. Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15(5):579–600, 2012.

Raphael Kramer, Anand Subramanian, Thibaut Vidal, and F Cabral Lucídio dos Anjos. A matheuristic approach for the pollution-routing problem. *European Journal of Operational Research*, 243(2):523–539, 2015.

Eduardo Lalla-Ruiz, Christopher Expósito-Izquierdo, Shervin Taheripour, and Stefan Voß. An improved formulation for the multi-depot open vehicle routing problem. *OR Spectrum*, 38(1):175–187, 2016.

AH Land and AG Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

Gilbert Laporte, Martin Desrochers, and Yves Nobert. Two exact algorithms for the distance-constrained vehicle routing problem. *Networks*, 14(1):161–172, 1984.

Gilbert Laporte, Yves Nobert, and Martin Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5):1050–1073, 1985.

Gilbert Laporte, Michel Gendreau, J-Y Potvin, and Frédéric Semet. Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, 7(4-5):285–300, 2000.

Ran Liu, Yangyi Tao, and Xiaolei Xie. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. *Computers & Operations Research*, 101:250–262, 2019.

François V Louveaux and Juan-José Salazar-González. Exact approach for the vehicle routing problem with stochastic demands and preventive returns. *Transportation Science*, 52(6):1463–1478, 2018.

Thibaut Lust and Jacques Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research*, 19(4):495–520, 2012.

Alan S Manne. On the job-shop scheduling problem. *Operations Research*, 8(2):219–223, 1960.

Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

Muhammad Nawaz, E Emory Enscore Jr, and Inyong Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

George L Nemhauser and Laurence A Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1999.

Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, pages 1–41, 2020.

Victor Pillac, Christelle Gueret, and Andrés L Medaglia. A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7(7):1525–1535, 2013.

Jakob Puchinger, Günther R Raidl, and Ulrich Pferschy. The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing*, 22(2):250–265, 2010.

Sebastian Reil, Andreas Bortfeldt, and Lars Mönch. Heuristics for vehicle routing problems with backhauls, time windows, and 3d loading constraints. *European Journal of Operational Research*, 266(3):877–894, 2018.

Benjamin C Shelbourne, Maria Battarra, and Chris N Potts. The vehicle routing problem with release and due dates. *INFORMS Journal on Computing*, 29(4):705–723, 2017.

Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

Éric D Taillard, Gilbert Laporte, and Michel Gendreau. Vehicle routeing with multiple use of vehicles. *Journal of the Operational Research Society*, 47(8):1065–1070, 1996.

Hamza Ben Ticha, Nabil Absi, Dominique Feillet, and Alain Quilliot. Multigraph modeling and adaptive large neighborhood search for the vehicle routing problem with time windows. *Computers & Operations Research*, 104:113–126, 2019.

Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002.

Edward Tsang and Chris Voudouris. Fast local search and guided local search and their application to british telecom's workforce scheduling problem. *Operations Research Letters*, 20(3):119–127, 1997.

Thibaut Vidal, Gilbert Laporte, and Piotr Matl. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 2019.

Juan G Villegas, Christian Prins, Caroline Prodhon, Andrés L Medaglia, and Nubia Velasco. A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, 230(2):231–244, 2013.

Jyrki Wallenius, James S Dyer, Peter C Fishburn, Ralph E Steuer, Stanley Zionts, and Kalyanmoy Deb. Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. *Management science*, 54(7):1336–1349, 2008.

Martin Weber, Franz Eisenführ, and Detlof Von Winterfeldt. The effects of splitting

attributes on weights in multiattribute utility measurement. *Management Science*, 34 (4):431–445, 1988.

Don Weigel and Buyang Cao. Applying gis and or techniques to solve sears technician-dispatching and home delivery problems. *Interfaces*, 29(1):112–130, 1999.

Laurence A. Wolsey. *Integer Programming*. Wiley, 1998.

Fulin Xie, Chris N Potts, and Tolga Bektaş. Iterated local search for workforce scheduling and routing problems. *Journal of Heuristics*, pages 1–30, 2017.

Jiyang Xu and Steve Y Chiu. Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7(5):495–509, 2001.

M Yokoyama. The nature of the affective judgment in the method of paired comparisons. *The American Journal of Psychology*, 32(3):357–369, 1921.

Yu Zhang, Roberto Baldacci, Melvyn Sim, and Jiafu Tang. Routing optimization with time windows under uncertainty. *Mathematical Programming*, 175(1-2):263–305, 2019.