Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Data Analysis with Mixed-Integer Optimisation for Scheduling Royal Mail Deliveries

Author:
Athanasios Liaskas

Supervisor:
Ruth Misener & Dimitrios Letsios

Second Marker:
Panos Parpas

June 26, 2020

Abstract

Industrial Operations Research is one of the original and most critical fields highlighting the effectiveness of the theory of optimisation when applied to real-world applications. This work investigates the efficacy of Mixed-Integer Optimisation techniques in scheduling the routes performed by Royal Mail's fleet of postal service vehicles. Our findings although theoretical are designed to be easily transferable to Royal Mail's current operational infrastructure, to help the corporation fulfil its same-day mail delivery mandate.

We show that the current scheduling practices are open to improvement on various aspects through solving Mathematical Programs that generate significantly more optimal schedules with respect to three objectives. We subsequently, select one of those optimal schedules and subject it to an uncertainty study. A Mail Transportation system such as the one run by Royal Mail is vulnerable from numerous aspects to uncertainty-related events. The purpose of the study is to determine the schedules' level of robustness against perturbations that resemble real-life disturbance occurrences. We conclude our project by studying two additional scheduling methodologies within the framework of Robust Optimisation to determine the methodology that best balances the provision of an efficient schedule that also remains robust under the influence of uncertainty.

Acknowledgements

The first half of the dissertation was undertaken under the supervision of Dr Ruth Misener, while the remaining part was completed with help from Dr Dimitrios Letsios.

I would like to thank Dr Ruth Misener for her advice as well as for enabling my cooperation with Royal Mail by organising this project.

I would also like to express my thanks to my supervisor Dr Dimitrios Letsios for his continuous support and guidance throughout my MEng Thesis.

The dissertation was undertaken in consultation with Royal Mail's Data Science Group.

Contents

Chapter 1

Introduction

Royal Mail the national mail provider of the UK runs a hierarchically structured organisation. On any given day, the Royal Mail postal service workforce belonging to one of Royal Mail's 1,250 Delivery Offices (DO) collects mail from customers and aggregates it at their respective DO. The mail is then collected by a fleet of *Heavy Goods Vehicles* (HGV) (7,000 in total) belonging to one of 50 regional centralised Mail Centres where mail is gathered for sorting, ready for its redistribution the following day. After the sorting process, the aggregated mail at each mail centre is sent down either of two paths. It is either sent onwards to a region covered by a different mail centre or it is sent back to one of the DOs covered by the MC where it was sorted. This project focuses on the latter rather than the former path, attempting to establish a method for scheduling the routes from the regional MCs to the local DOs and back. The objective is to generate schedules that **optimise the use of Royal Mail's HGV drivers' time**.

This problem might appear to belong in the class of *Vehicle Routing* problems given that the majority of *delivery system* problems tend to focus on the optimisation of the routing. However, due to the repetitiveness of the daily routes to-and-from each mail centre, the project leverages the HGV drivers' prior familiarity with the routes to make efficiency gains.

The employees responsible for the scheduling of Royal Mail's operations are currently generating the daily schedules heuristically without any algorithm-based procedures associated with the process they employ. In essence, they are attempting to perform a **complete search** among the spectrum of feasible timetables for each day. Due to the complexity of such a task, they often compromise for a non-optimal arrangement that satisfies the legal requirements, with little to no regard as to whether it is optimal. As a result, the HGV drivers' time is not utilised as efficiently as it could be hence, resulting in additional costs for Royal Mail, and drivers' shifts often ending very late in the day. Given that there is currently no efficient method guaranteeing the provision of optimal schedules, even a marginal improvement in minimising the time drivers spend off the road could yield substantial cost savings to Royal Mail's budget as well as a more uniform timetabling of shifts.

Contributions The project's challenge is to construct an efficient method that will provide scheduling strategies by formulating and solving a *discrete optimisation problem*. In order to achieve this, we first need to understand whether true optimal solutions to the problem do exist. By formulating a mixed-integer linear program and analysing it over a dataset of real scheduling data, provided by Royal Mail, we will evaluate whether optimal exact solutions can be obtained. Moreover, we will generate exact solutions that are optimal with respect to a series of different objectives that have different qualitative effects when implemented. Following that, we must explore whether it is possible to incorporate aspects of Royal Mail's company policy to the problem in order to provide more realistic solutions. By analysing the drivers' workflow at a more detailed level we aim to determine whether, the removal of pre-existing tactics regarding the sequence with which drivers perform their assigned tasks can further minimise the time they spend performing non-critical activities.

The final contribution is arguably one of the most critical points in the study of the development of an efficient schedule for Royal Mail since it involves the incorporation of a taste of reality in our timetabling efforts. The schedules developed in previous sections operated in a deterministic *offline scheduling* environment. Namely, as the scheduler we are aware of every level detail with respect to the components that need scheduling, prior to attempting to create the schedule. However, for our final contribution we choose to go beyond such restrictions and allow various components to be perturbed up to a certain degree such that we can simulate the effects of various uncertainty components that are bound to occur in a real-life Mail Transportation environment such as that of Royal Mail. Consequently, we proceed to compare and contrast two different scheduling methodologies to determine which one can remain undeterred by the addition of the uncertain environment while still preserving a fairly optimal and efficient schedule.

A summary of the contributions of the project is found below:

- **Formulate Exact Models:** We formulate deterministic parallel machine models that accurately represent the context of the Royal Mail problem. Each model examines the problem through a different prism and attempts to optimise it under the affiliated objective.
- **Attainment of Exact Solutions:** Utilising the formulated models we obtain optimal solutions for each instance of the problem that highlight the opportunities for optimisation that exist within the domain of this problem.
- **Comparison of solutions to Historical Schedules of Royal Mail:** We compare the heuristically obtained schedule currently operated by Royal Mail to our optimised schedules for each objective. We evaluate the degree to which our solutions are out-performing the historical schedule for each objective.
- **Study of Uncertainty:** We conduct experiments that determine the effects of the application of uncertainty components that resemble real-life perturbations have on our schedules.
- **Robust Optimisation:** Having established the consequences of the disturbances on our schedules, we study two methodologies, Lexicographic Optimisation and Polynomial Objective Makespan Scheduling, from the field of Robust Optimisation to obtain efficient schedules that are also robust to uncertainty..

Organisation

- Chapter 2: outlines the background concepts that the reader needs to understand, in order to be capable of following the principles explored throughout this dissertation.
- Chapter 3: describes the problem of interest and presents the reader with an overview of the data provided by Royal Mail.
- Chapter 4: evaluates the current practices ran by Royal Mail, by performing simple Mixed-Integer Optimisation, and obtains optimal solutions through computer-based procedures for deriving solutions. It then compares the optimised solutions to the problem from the model, and subsequently validates and refines the model where needed.
- Chapter 5: introduces the study of uncertainty conducted in this dissertation. It evaluates the robustness of one of the optimal solutions obtained in Chapter 4 concerning uncertainty. It then studies robustness enhancing methodologies that aid in the development of more uncertainty robust schedules.
- Chapter 6: evaluates the body of work discussed in this dissertation
- Chapter 7: concludes with a synopsis of the key achievements of this dissertation and a series of ideas for future exploration, and improvements to the modelling that would extend the aspects of the problem captured.

Project Management The Gantt chart featured in Figure 1.1 provides an outline of the various steps that we implemented in order to complete the experiments in this dissertation. As one can observed from the chart, Milestone (2) lasted a while longer than originally planned for. This was mostly due to the fact that the analysis of the dataset was a much bigger task than initially anticipated. This had a substantial effect on our ability to complete Milestone (3) since the study of uncertainty was added in fairly late stages of the project and the pursuit of our full intentions for that chapter was not realised. Namely, we did not end up studying the effects of ellipsoidal uncertainty sets as we originally planned to.

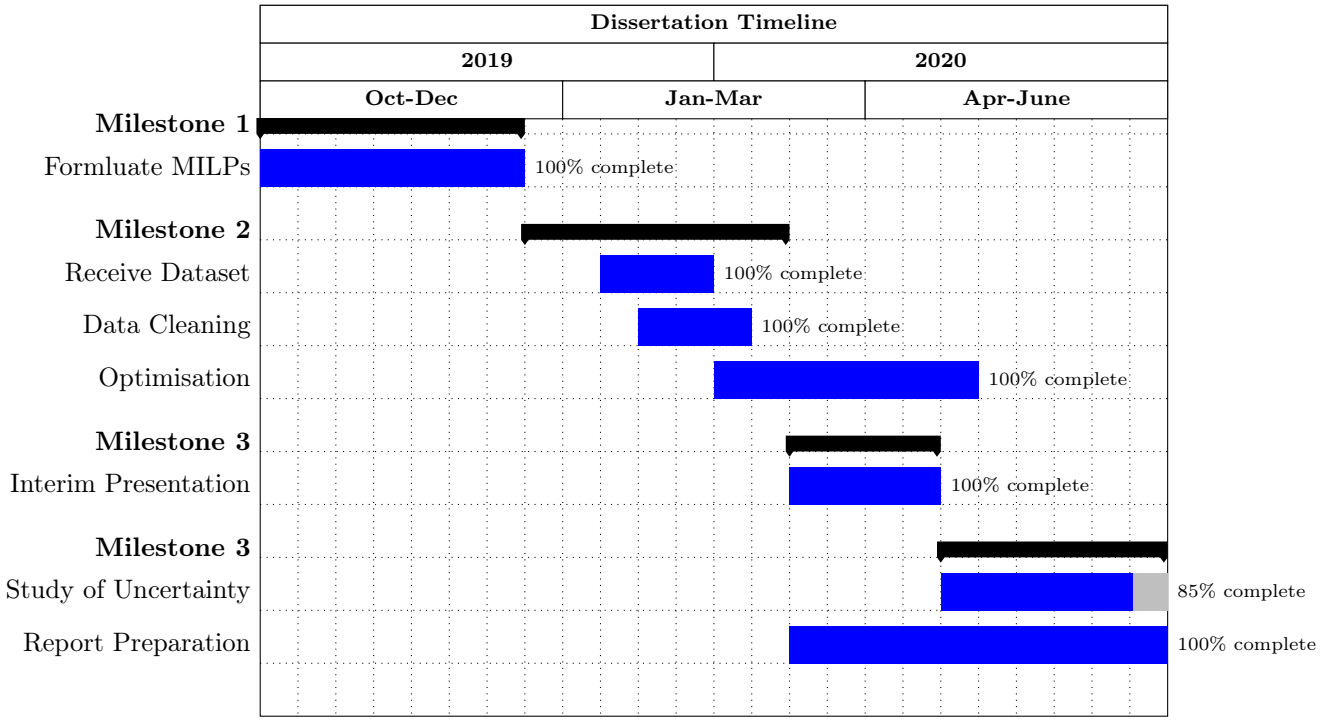


Figure 1.1: Gantt chart illustrating the various milestones that were performed to complete this dissertation.

Chapter 2

Background

In this chapter, we introduce the fundamental concepts of mathematical programming as well as simplified forms of the models on which our modelling efforts will be built upon throughout this dissertation. This material is organised in a way that hopefully can provide the reader with a high-level understanding of the necessary mathematical concepts that will be utilised throughout this dissertation. A literature review for each topic covered has been incorporated accordingly to each section of the chapter. To gain a deeper understanding of those principles, and a more accurate grasp of the state of the art our reader is advised to thoroughly consult the literature sources referenced throughout this report.

Sections 2.1-2.1.2 give a generic definition of Mathematical Programming, and its branches. They also mention the principal algorithms that are going to be utilised for the solution of the mathematical problems dealt with in this dissertation. Section 2.5 studies the class of problems most relevant to the context of our problem. Sections 2.2-2.4 describe the theoretical concepts used in the latter stages of the dissertation for the experiments with the higher degree of difficulty but also the most value-added, with respect to new insights gained.

2.1 Mathematical Programming

A mathematical program involves the maximizing or minimizing of an objective function by providing input values from within a defined domain, computing the value of the function that dictates the quality of each solution and subsequently choosing the best available outputs.

General mathematical programming models can be stated as [1]:

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) = 0 \quad i = 1, 2, \ldots, m \\ & h_j(x) \leq 0 \quad j = 1, 2, \ldots, r \end{aligned} \tag{2.1}$$

$$\text{where } x \in S \subset \mathbb{R}^n \ , \ f : \mathbb{R}^n \to \mathbb{R} \ , \ g : \mathbb{R}^n \to \mathbb{R}^m \text{ and } h : \mathbb{R}^n \to \mathbb{R}^r$$

The *decision variables* are represented by an n-dimensional vector $\boldsymbol{x} = x_1, x_2, \ldots, x_n$ and the *objective function* f is a function of the decision variables that we want to minimise. The *feasible set* S determined by the equations $g(x)$, inequalities $h(x)$, and set restrictions and it is a subset of the n-dimensional space containing all the admissible decisions. All the points $x \in S$ are feasible solutions, but sought after is an optimal solution, a vector $\boldsymbol{x^*}$ that satisfies the constraints of the feasible set, and achieves the best possible outcome i.e.:

$$f(x) \geq f(x^*) > -\infty \ \ \forall \, x \in S$$

The objective of mathematical programming is to utilise optimisation theory, and algorithms to obtain those optimal solution(s).

2.1.1 Linear Programming

A *Linear Program (LP)* is a type of mathematical program where the objective function and the constraints are **linear** over the feasible set of decision variables. The conventional form for representing LPs is the *standard form* below [1]:

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ \text{subject to} \quad & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ & a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \\ \text{and} \quad & x_1, x_2, \ldots, x_n \geq 0 \end{aligned} \tag{2.2}$$

where b_i, c_i and a_{ij} are fixed real constants and we require that $b_i \geq 0$

In principle, formulating the program as a maximisation problem is equivalent to equation (2.2) since through the simple transformation of the objective $maxf(x) = -min(-f(x))$ we can convert our problem. However, by convention the minimisation formulation in (2.2) has prevailed, and one can transform any maximisation problem into a minimisation one without any loss of generality. The purpose of establishing a *standard form* for all LPs is to create a default version that represents the standard for all LP formulations for which we can develop efficient solution algorithms. All differently formulated LPs can then be transformed to this *standard form* through a series of simple operations such as the addition of slack, surplus variables so that we can utilise the effectiveness of the *standard form's* universal algorithms. The formulation (2.2) above can also be reduced in vector notation to this compact version:

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.3}$$

where $b \geq 0$ and $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^{m \times 1}$, $c \in \mathbb{R}^{1 \times n}$

A program of any scientific interest is *well-defined,* and its feasible set is concretely **bounded** and **non-empty**. For LPs the feasible set circumscribed by the set of linear constraints, translates to a **convex polyhedron** containing an infinite number of solutions. According to the *Fundamental Theorem of LP*, at least one of the vertices of the polyhedron feasible set, contains the optimum solution to the LP**.**

Consequently, an initial idea of an algorithm that obtains the optimal solution from inside the feasible set is the procedure of examining which of the finite number of vertices of the polyhedron S produces the best objective value. However, computing the optimum through a *finite search* over all the vertices is usually computationally prohibitive especially since the number of vertices increases exponentially with the number of constrains and variables.

Simplex Algorithm

To combat this the *Simplex Algorithm*, the most frequently used method, starts from an initial corner point and then inspects a fraction of all possible vertices *pivoting* from the initial vertex to vertices with a guaranteed ever-improving objective value, until the optimum is found [2]. Despite its effectiveness in obtaining a solution the simplex is not guaranteed to solve an LP in *polynomial* time, especially considering that in some cases it is equally as hard to find an initial corner point as it is to find the optimum solution [3].

Interior point methods

Another class of algorithms used to deal with LPs is that of *interior point methods* such as *Karmarkar's Algorithm.* In contrast to the simplex, it explores the interior of the feasible space rather focusing on its corner points and identifies an optimum solution in *polynomial* time [4].

2.1.2 Integer Linear Programming

A *Pure-Integer Linear Program (PILP)* is a branch of linear mathematical programming in which decision variables are required to be non-negative integers.

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0, \qquad x \in S \subset \mathbb{Z}^n \subset \mathbb{R}^n \end{aligned} \tag{2.4}$$

$$\text{where } b \geq 0 \text{ and } A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^{m \times 1},\ c \in \mathbb{R}^{1 \times n}$$

A visualisation of the feasible set of a program that includes the integrality constraint can be seen in Figure 2.1. Assuming, that we consider an integer problem for which $\boldsymbol{x} \in \mathbb{Z}^2$, Figure 2.1(a) illustrates the spectrum of possible decisions that we can make for this problem. Those decision are the subset of integer pairs that belong in S.

In a *Mixed Integer Linear Program (MILP)* only a subset of the decision variables have an integrality constraint. Taking as an example the two-dimensional MILP of Figure 2.1(b) in which one decision variable is continuous ($x_1 \in \mathbb{R}$) whereas the other can only take discrete values ($x_2 \in \mathbb{Z}$).

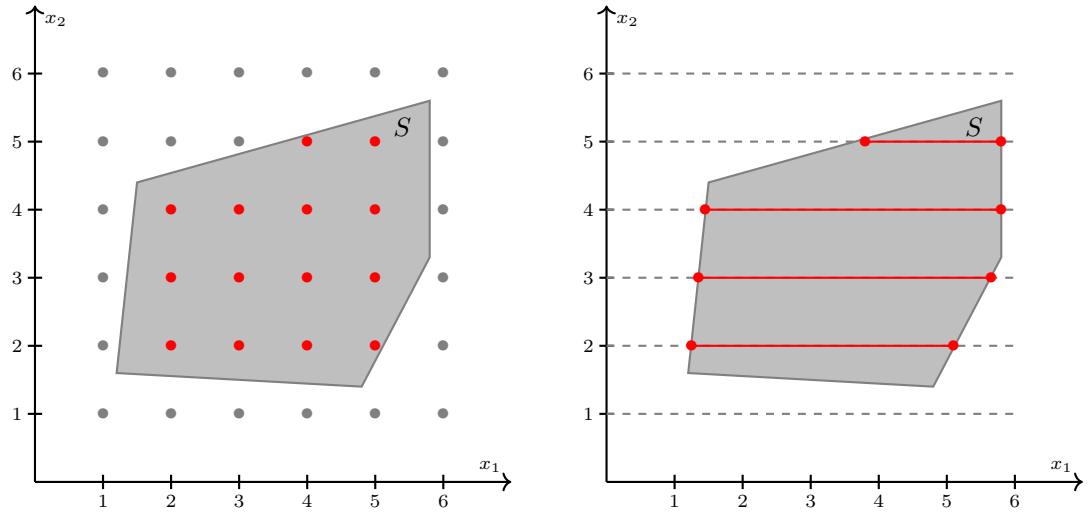
Many real-life problems can be formulated as integer linear programs. The study of this class of programs and its algorithms is particularly important for the purposes of this dissertation since we will be formulating our problem as a MILP. In particular, the problem that we will be studying from Chapter 3 onward, is a Binary Mixed-Integer Program where binary discrete variables are used to model whether or not some event takes place.

In reality, even though it seems slightly counter-intuitive at first, integer programs are considerably harder to solve. A naive assumption would be to claim that in an ILP, one only has a finite number of possible decisions (as seen in Figure 2.1(a)) so in principle one could essentially try them all. However, if the problem's scale is increased we will transition to a combinatorial problem where the number of solutions that one has to search over with this brute force approach becomes very large very quickly. We mentioned before that in the environment of LPs when we add new constraints the number of solutions increases exponentially. In an ILP specifically, the addition of n binary variables increases the solution space by 2^n [5]. As a consequence, a *polynomial* time algorithm for ILPs has not yet been proven to exist [3], and such MILPs are considered *NP-hard* [6].

To solve mixed-integer linear problems, we can either reuse or extend algorithms designed for LPs or construct new ones specifically for ILP problems. The algorithms studied below are of the former category and they largely exploit the efficiency of the simplex algorithm[1] by disentangling an integer LP into a corresponding linear program through relaxing[2] the integrality constraints.

This approach is based on the idea that the ILP's feasible set is a subset of the relaxed LP's domain[3], hence the integer optimum solution is guaranteed to be contained within the linear spectrum of solutions. Subsequently, through solving numerous iterations of the *standard form* LPs, utilising simplex, their feasible sets are gradually modified to converge towards an optimum solution that is also integer hence, obtaining a solution that is also the optimal for the original ILP. The reason for multiple such algorithms to exist is revolved around the way they modify the relaxed LPs feasible set to converge towards the integer solution.

[1] Designed for standard form LPs as was previously seen in Section 2.1.1.
[2] The concept of obtaining the relaxation of a program is further explained in Appendix D.2 for interested readers.
[3] $S_{ILP} \subseteq S_R$, where S_R is the relaxed LP's feasible set.



(a) Pure-Integer Linear Programming problem (PILP), where $\boldsymbol{x} \in \mathbb{Z}^2$.

(b) Mixed-Integer Linear Programming problem (MILP), where $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{Z}$.

Figure 2.1: Illustrations of the Feasible Sets (S) of an ILP and a MILP respectively.

Branch and Bound

The *Branch and Bound (BB)* follows this philosophy of modifying the feasible set until an integer solution is obtained. It begins by obtaining the optimum solution of the relaxed LP. In the case that this initial solution happens to be integer it will immediately terminate since that will mean that the optimum for the original ILP has also been found. If it is not however, the algorithm modifies the LP's feasible set until its solution satisfies the integrality constraints.

Those modifications take place at every iteration, and for the purposes of the BB we refer to them as *branching*. To carry out branching the algorithm divides the solution space into two LP subproblems, by eliminating a part of the LP relaxation's feasible set that did not contain any feasible integer solutions. For the ILP there is no loss of generality since the union of the subproblems' feasible sets contains the exact same feasible integer solutions. The two emergent subproblems are then solved separately as regular LPs. Due to the absence of any loss of generality the optimum ILP solution is guaranteed to be in just one of the two new emergent solution spaces of the subproblems. If the solution to either of the two subproblems is worse than any previously known solution, then the corresponding subproblem and its feasible space are disregarded as non-promising. This is known as the *fathoming* step. In essence through such iterations of the algorithm a lower possible bound for the objective value is eventually obtained.

By enforcing these two steps iteratively, a binary tree is constructed. Nodes with a non-integer solution have two branching subproblems et cetera. The algorithm is terminated when all nodes will either have an integer solution or a non-integer solution that is worse than the best integer solution of some other node. The node with the most optimum integer solution is the solution to the original ILP [3].

Cutting Plane

The *Cutting Plane (CP)* method is also concerned with the transformation of an ILP into a LP that has naturally an integer solution. The process behind this is to gradually cut out part of the relaxed LP's admissible set while leaving the feasible region of the ILP completely unchanged. Hence, eventually upon the implementation of a sufficient number of such *cuts* we will obtain an LP with an integer solution. However, the process of generating the cuts is not a trivial task, and that is the heart of the algorithm.

The algorithm starts by relaxing the integrality constraints to obtain the LP relaxation of the original ILP. Starting from the optimum solution of the LP relaxation, if it is integer, then the

cutting-plane algorithm terminates, and this optimum solution is also optimum for the ILP. If it is not, we *tighten* the feasible set of the LP and repeat this process.

The *tightening* is done by generating a *cut,* that is a new constraint that restricts unwanted non-integer solutions. This new linear constraint is added to the relaxed LP's solution space hence modifying S_R into a new set S_{CH}. This process of modifying S_R is repeated until the set converges to one of the ILP's solution space vertices, at which point the optimum integer solution will have been obtained. Through iteratively performing these steps we create the *convex hull* S_{CH} of the ILP's feasible set S.

Consequently, starting from the ILP below:

$$\underset{x}{\text{minimise}} \quad c^T x, \quad x \in S$$

Through generating multiple cuts at each iteration we can instead solve:

$$\underset{x}{\text{minimise}} \quad c^T x, \quad x \in S_{CH}$$
$$\text{since } S \subseteq S_{CH} \subseteq S_R$$

The ideal operating scenario for this method would be to generate a relatively small account of highly effective cuts. However, the effectiveness of a cut is a topic with a hard to obtain answer. A universally good cut, would be a cut where the vertex of S_{CH} on which the cut is performed is the furthest from the perimeter of S_R since that would automatically mean that the largest possible amount of unwanted feasible space has been cut off. In order generate such good cuts, the simplex algorithm can be utilised. Namely, at each iteration after locating a vertex, we can use the simplex algorithm at that point to determine the best point to pivot to in order to perform the following cut [7].

Overall, a pure cutting plane approach is considered computationally intensive because more often than not, multiple cuts are required. As a direct consequence multiple constraints will be added to the LP relaxation rendering it less computationally tractable. Especially, if we consider that the number of cuts required is not necessarily dependent on the problem's scale as explained in [2]. In general, there also exist many kinds of cuts such as *Mixed-Integer rounding,* or *Knapsack cover* cuts derived from the logic governing the class packing problems. This adds another variable to the algorithm regarding the choice of cuts. The most prominent are *Gomory mixed-integer* cuts [8]. Although at the time of their development they were unable to translate their theoretical effectiveness into practice they have recently been revisited and are now considered a go-to method. That is due to the following two reasons. Firstly, due to the emergence of more powerful and robust solvers [9]. Secondly, they have proven to outperform other methods when utilised with a hybrid implementation of the cutting algorithm instead of the pure cutting plane approach. Namely, when utilised a method that combines the *branch-and-bound* with the *cutting-plane* algorithm (i.e. a *branch-and-cut*)[10] they manage to generate substantially better bounds leading to a decreased computation time compared to the pure versions of those two algorithms [3].

Duality

According to the principle of *duality,* in linear programming every LP can be associated with a corresponding *dual* linear program. The *dual* of a problem is a LP derived from combinations of the constraints of an original LP model, referred to as the *primal.* The solution of the *dual* signifies the best possible bound of the original problem. In matrix form, we can express the *primal* problem as:

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

with the corresponding **symmetric** dual problem,

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & b^T y \\ \text{subject to} \quad & A^T y \geq c \\ & x \geq 0 \end{aligned}$$

The two problems are completely intertwined with each other. The objective of the primal program is transformed to its mirror-image counterpart in the dual, in a way that if the original is a minimisation problem its dual attempts to maximise the objective function and vice versa. Finding the dual of the dual problem will lead us back to the primal problem.

In Linear Programming specifically, the concept of *strong duality* states that the optimum solution to the dual problem is also the optimum for the primal and vice versa. We can leverage this concept when solving LPs where the number of variables is significantly smaller than that of the constraints, since solving the dual will be considerably more computationally efficient and its solution will also be optimum for the original problem.

A practical use of the duality concept is that it drives the *post-optimal analysis,* to determine the degree of sensitivity of the optimum solution to any changes in the input parameters.

Sensitivity Analysis

Sensitivity Analysis leverages the relationship between primal-dual problems to calculate the changes in the optimum solution as a result of changes in the input parameters, in an efficient manner. The utility of performing a sensitivity analysis is to gain insights of the relationships between the various components of the problem. For instance, in chapter 3 we will conduct a sensitivity analysis to find out the various the trade-off between the objective function and a variable of the problem. By probing the objective function for various values of that variable we are able to determine the thresholds at which the objective value changes as well as the degree by which it changes. Finally, while performing a sensitivity analysis it is often useful to determine the degree to which a component can remain robust with respect to some changes in another component. For instance, we could establish limits within which changing a certain variable will not cause a change in the optimal solution. Consequently, having done that we will have determined the sensitivity of the optimal solution to changes with respect to that variable.

Lagrangian Relaxation

The *Lagrangian dual* problem is a bounding technique for solving ILPs that does not disregard the integrality constraints as in previously stated techniques, but instead relaxes some of the main constraints. By forming a combination of the constraints through multiplying them with non-negative Lagrange multipliers, the dual of the original ILP is constructed. The relaxed problem is significantly more computationally tractable, and through solving the dual we can establish a best possible bound for the original ILP.

2.2 Multi-Objective Optimisation and Pareto Optimality

Taking a brief diversion from the context of Linear Programming, we look at *Multi-Objective Optimisation* problems where the objective function is comprised of a multitude of objectives. For such problems the *Pareto Optimality* principle proves particularly handy when looking for a **Pareto optimal** solution. In simple terms To define a Pareto optimal solution we first need to address the concept of a **dominated solution**:

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & f(x) \\ \text{subject to} \quad & h_i(x) = 0 \quad i = 1, 2, \ldots, m \\ \textit{where } x \in \mathbb{Z}^n \ , \ & f : \mathbb{Z}^n \to \mathbb{Z}^c, \ h : \mathbb{Z}^n \to \mathbb{Z}^m \end{aligned} \tag{2.5}$$

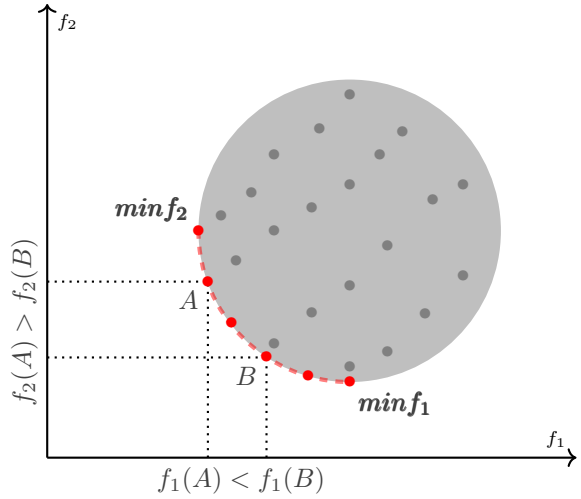


Figure 2.2: The image of the feasible set of a Bi-Objective Optimisation problem showcasing the Pareto Front.

Given the constrained multi-objective minimisation problem in (2.5), that has an objective function $f(x)$ which is a combination of a set of o objective functions $f_1(x), f_2(x), \ldots, f_o(x)$ that require simultaneous minimisation, and a feasible space defined by a set of constraints m, a feasible solution d is **dominated** by another feasible solution s if the following two conditions stand:

$$\begin{aligned} &(1) \quad f_j(s) \leq f_j(d) \; \forall \; j \in [1, \ldots, o] \\ &(2) \quad \exists \; k \in [1, \ldots, m] \mid f_k(s) < f_k(d) \end{aligned}$$

In simpler terms, these two conditions can be reduced down to: (1) solution s is *as good performing as* d at every objective o, and simultaneously, (2) solution s **strictly** better performing than d at at least one objective [11]. Using the definition of a *dominated* solution, we then define the Pareto optimal solutions as the set of **undominated** solutions from within the feasible set.

In the context of a practical optimisation problem, finding a feasible point that is concurrently Pareto optimal means that it is only impossible to improve one objective at the expense of another. Hence, it captures the concept of a **trade-off** between the various objective in the objective function. The reason behind the efficacy of this concept is that more often than not, when presented with a multi-objective optimisation problem it is usually precarious to apply the philosophy of linear programming and trying to optimise multiple objectives in a brute-force way, as it is likely that the optimiser for one of the objective is not going to coincide with that of a different objective. That is because more often than not, especially in problems with numerous objective functions, some objective functions end up being in conflict with each other [12]. Hence, we need to address a multi-objective optimisation problem with a slightly different notion of optimality.

Suppose, we are given a constrained multi-objective optimisation problem similar to (2.5), where $o = 2$, (i.e. our problem has only two objectives). Plotting the image of the feasible set of this problem on the plane (f_1, f_2), as seen in Figure 2.2 [13] can help us locate the **minimum** for each objective with respect to all feasible solutions. However, upon trying to simultaneously minimise f_1, f_2 one can see from the figure that $minf_1$ is not particularly optimal for f_2 and vice versa. If one chooses to minimise f_1 by choosing point $minf_1$, they do so at the expense of f_2. Hence, according to the Pareto principle we characterise the set of points in between $minf_1, minf_2$ as *Pareto optimal*, representing the points for which one cannot improve any of the objectives without sacrificing one of the others. These set of points is represented by the red-coloured boundary of the image of the feasible set, and are often seen as **efficient frontier** in literature.

All in all, this new notion of optimality defined through the Pareto principle refers to the difference in the approach followed for the solution of a standard linear program compared to a multi-objective problem untangled with the Pareto concept. Instead of attempting to minimise all objectives at

once, which would most likely prove ineffective, we instead solve the problem for each objective separately to get an overall depiction of the optimal feasible solutions through a figure like that in Figure 2.2 then leaving it up to the optimiser to decide the best compromises to be made. This philosophy can be summarised in the following steps [11]:

Algorithm 1: How to obtain the **efficient frontier**

Result: End points of the efficient frontier $(minf_1, minf_2)$
initialization: i=1;
for $i<o$ **do**
  i+1;
  minimise f_i;
  find the value s_j for objective $j \neq i$;
  $(minf_i, s_j)$ is a point on the efficient frontier;
  if *there do not exist dominant points s with respect to objective j* **then**
    minimise f_j where $j \neq i$;
    to find $(minf_j, s_i)$;
  else
    points $(f_i(s), f_j(s))$ belonging on the efficient frontier;
  end
end

2.3 Optimisation under Uncertainty

2.3.1 Robust Optimisation

In general there are mostly two approaches to model uncertainty within the context of optimisation, Stochastic Optimisation and Robust Optimisation. For the purposes of this dissertation we focus on the *robustness* of our models which we define as the quality of a method to remain feasible even after the application of an **uncertain set** U on it. Utilising the formulation (2.3) we can write the robust version of an LP as [14]:

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \quad \forall a_1 \in U_1, \ldots, a_m \in U_m \\ & x \geq 0 \end{aligned} \tag{2.6}$$

$$\textit{where } b \geq 0 \textit{ and } A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m,\ c \in \mathbb{R}^n \text{ and } U_i \in \mathbb{R}^n$$

More specifically, in terms of comparing the *robustness* of two solutions to a problem we argue the following. Assume one has an optimisation problem, for instance a *MILP*, which has a certain objective function, and one can obtain a certain number of *solutions* for it through a certain set of *methods*. Upon, applying the uncertain set U on the different solutions, we characterise as **more robust** the method that after uncertainty provides the solutions with the best objective value.

Robust Optimisation under Uncertainty

One of the fundamental elements of Robust Optimisation are uncertainty sets. An interval-based uncertain set U, is defined by a *lower* and an *upper* bound, such that an instance of the random variable s is enclosed in the following [14], [15]:

$$U = \prod_{i \in n} \{s_i \in \mathbb{R}^n \mid \underline{s_i} \leq s_i \leq \overline{s_i}\} \text{ for Box Uncertainty Set} \tag{2.7}$$

Hence, one can now generate instances I of a problem that have this uncertainty set U applied to it, from the following **spectrum of Instances**: I_{lower} (every s is $\underline{s}$), I_{middle} (every s is $\frac{\underline{s}+\overline{s}}{2}$), and

I_{upper} (every s is $\overline{s}$). If we assume that s is enclosed in a ball of radius Ω centered at the origin [16], we can have the following [17],[18]:

$$U = \prod_{i \in n} \{s \in \mathbb{R}^n \mid \sum_{i=1}^{n} s_i^2 \leq \Omega^2\} \text{ for Ellipsoidal Uncertainty Set} \tag{2.8}$$

However, in order to use the concept of *uncertainty sets* to simulate the concept of uncertainty in real-life one should select values of s **at random** from within $[\underline{s}, \overline{s}]$, as this will be a fairer representation of the chaotic randomness that tends to occur in reality.

Generally speaking the generation of uncertainty set is not a trivial matter. D. Bertsimas and D. Brown develop a theory in [19] revolved around the generation of uncertainty sets that replicate explicit uncertainty sets as far as their structure and behavior.

2.4 Lexicographic Optimisation

The study of the method of Lexicographic Optimisation is a gateway that allows us to transition to the playing field of multi-objective optimisation. In utilising the concepts of lexicographic optimisation one hopes to minimise those multiple objectives while sequencing the objective functions according to a lexicographical ordering [20]. As mentioned in [21] using a lexicographic ordering in combination with the Longest Processing Time heuristic seen in 2.6, we expect to solve the Makespan Schedule problem with superior results both in terms of efficiency but also with respect to quicker solution times.

In general the purpose of using a lexicographically generated schedule is to receive a schedule designed ideally for the process of rescheduling once we determine that some disturbance has occurred [22]. However, [21] references us towards works such as [23],[24] that showcase the effectiveness of lexicographic optimisation not only in obtaining efficient schedules but also in doing so in a relatively quick fashion provided our mixed-integer problem satisfies certain requirements. Being aware of this reasoning behind the use of lexicographic optimisation, we expect to obtain favourable results compared to less state-of-the-art methodologies. Indeed, as we will see in 5 that is the case, as this methodologies tends to outperform other methodologies studied with respect to various aspects.

2.5 Vehicle Routing Problems

Our problem can be interpreted as a *Vehicle Routing Problem (VRP)* with collection and distribution[4] [25]. We do not explicitly focus on the study of the routing of the problem, but instead focus on the scheduling aspect and only utilise some concepts derived from the *VRP* discipline in our project. Nevertheless, we propose the application of the routing aspect of the *VRP* philosophy on our problem as one of the future directions that need to be explored in the context of our problem. Hence, we include below a brief study, that aims to give the reader a high-level overview of this class of problems.

Vehicle Routing Problems (VRP) have been studied intensely over the years, and well known survey papers such as [26] can give an interested reader further information on the problem. Moreover, the following paper [27] outlines the most frequently used algorithms towards efficient solving of problems of this context. Such algorithms are used for the purposes of a formulation seen in Section 4.5 of Chapter 4, where the vehicle routing aspect of our problem is studied. Moreover, given that the application of the vehicle routing concepts on our problem is presented as one of the directions for future research, we believe that a careful study of efficient approximation algorithms such as those studied in [28] would be purposeful.

[4]Namely, a problem that is capable of incorporating a delivery and a pick of an item to and from a location in the same trip.

2.6 Multiprocessor Scheduling Problems

Scheduling problems are a common class of decision-making problems that deal with the search for the most efficient method of allocating resources to tasks. This search for efficiency is explored by attempting to optimise a single or multiple objectives related to the problem which eventually leads us to the most suitable schedule for each problem. In real world professional environments, suitable task schedules are critical to ensure a good balance of the load amongst the parallel machines [3].

There are numerous kinds of scheduling problems. This dissertation focuses on the class of deterministic problems that are supplied as input a collection of tasks requiring processing on an environment comprised of a bank of identical parallel machines. In *deterministic* problems, the input data is made available prior to the optimisation process and the aim is to come up with the most efficient sequence of these jobs, subject to a set of constraints, that optimises one or more performance criteria. This task can be broken down to, deciding which tasks have to be allocated to each machine as well as how to sequence the jobs dispatched to each machine. Machines can process at most one job on any given moment in time, and will see a job to its completion once its execution has begun.

Makespan Scheduling

In *makespan* scheduling problems, the objective function to be minimised is the *makespan* ($C_{\max}$). The *makespan* defined as $max\,(C_1, C_2, ..., C_n)$ is equivalent to the point in time that signifies the **termination** of the **last job** to be **completed**. Our goal is to sequence the jobs in question, in an efficient manner that enables the last job to finish processing as early as possible, while respecting the specified set of constraints. In essence, this type of problems resemble the generalised bin-packing problems in the context of scheduling [29]. The minimisation of the makespan is one of the most utilised optimality criteria [30] because striving to minimise the overall makespan of a schedule more often than not, results in a schedule that achieves a well-balanced sequence of jobs [3].

Set of Input Data

Set $M = \{1, 2, ..., m\}$ specifies an environment of m parallel and identical machines.

Set $J = \{1, 2, \ldots, n\}$ represents the number of jobs that await to be sequenced on each machine.

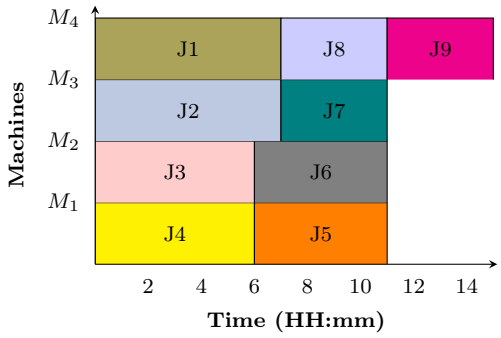
The n-dimensional vector $\mathbf{p}$ contains the processing time p_j of job j.

Parallel Machine model

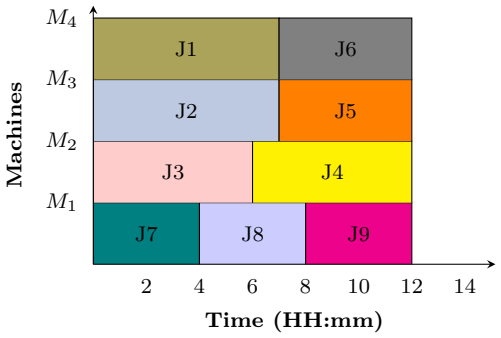
$$
\begin{aligned}
\underset{x_{ij}}{\text{minimise}} \quad & C_{\max} \\
\text{subject to} \quad & C_{\max} \geq \sum_{j=1}^{n} x_{ij} \cdot p_j && \text{for } i \in M \\
& \sum_{i=1}^{m} x_{ij} = 1 && \text{for } j \in J \\
& x_{ij} \in \{0, 1\} && \text{for } i \in M,\ j \in J
\end{aligned}
\tag{2.9}
$$

The first constraint reflects the definition of the *makespan*, and makes sure that it takes the value of the completion time of the last job to be completed. The second constraint enforces that each job is executed once and only once throughout the spectrum of all the machines. Finally, the third constraint is the integrality constraint of such a *MILP* problem.

As we can see, Figure 2.3 contains an example of two schedules[3]. The example is meant to illustrate the effect of applying a Makespan Scheduling formulation on the non-optimised schedule seen in 2.3(a). In this case the set of input data is the following:



(a) Un-optimised that our model receives as input for rescheduling.



(b) Rescheduled and Makespan optimised Schedule.

Figure 2.3: Illustrations showing the process of **Rescheduling** implemented throughout this dissertation.

We have four parallel and identical machines.

We have nine jobs each with their own processing time p_j.

We can see that by applying this set of input data to formulation 2.9 we get a reduction in the makespan of the schedule in 2.3(a) from 15 hours down to 12 hours in in 2.3(b). Moreover, as expected the workload of all machines is considerably more balanced since they are all of a duration equal to 12 hours. Indeed, it is true however, that those duties that we originally below the 12 hour mark have seen their duration increased (up to the 12 hour mark).

Longest Processing Time

In order to achieve an effective sequencing of the jobs, heuristics such as the *Longest Processing Time first (LPT)* rule have been developed. According to this algorithm, the m first jobs with the longest processing times p_j are assigned at the start to the m available machines. The first machine, out of the batch of those m machines to finish its execution is subsequently assigned the job with the longest processing time amongst those remaining, and so forth. It turns out by definition, that the job with the shortest processing time is the last one to begin its processing.

Time-Indexed Models

In *time-indexed* formulations, we choose to add the dimension of *time* to our model. We can express the same problem with a *discrete time* formulation, by splitting continuous time t into T discrete intervals. Consequently we have that, $t \in T = \{0, 1, \ldots, T-1\}$. We then use binary variable x_{ijt} which equals to 1 if job j starts at time t on machine i and equal to 0 the rest of the time [3].

$$
\begin{aligned}
\underset{x_{ijt}}{\text{minimise}} \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{t=0}^{C_{\max}-1} x_{ijt} \cdot (t + p_j) \\
\text{subject to} \quad & \sum_{i=1}^{m} \sum_{t=0}^{C_{\max}-1} x_{ijt} = 1 && \text{for } j \in J \\
& \sum_{j=1}^{n} \sum_{s=max(t-p_j,0)}^{t-1} x_{ijt} = 1 && \text{for } t \in T,\ i \in M \\
& \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{s=max(t-p_j,0)}^{t-1} x_{ijt} \leq m && \text{for } t \in T \\
& x_{ijt} \in \{0, 1\} && \text{for } i \in M,\ j \in J,\ t \in T
\end{aligned}
\tag{2.10}
$$

With the first constraint we make sure that each job is executed once and only once throughout all the machines. The second constraint enforces the definition of the environment of machines, namely that each machine does not process more than one job at any given moment in time. The third constraint is in place to prevent the possibility of more jobs being executed at some point in time, than there are machines available in the environment. Finally, the last constraint reflects the integrality constraint of such a *MILP* problem.

Comparing this time-indexed formulation with that in (2.5), it is visible that (2.6) will result in substantially more variables being used, $(m{\cdot}n{\cdot}C_{max})$-many x_{ijt} variables to be exact. Come solution time, this will be a factor that slows down the process of obtaining a solution to (2.6).

Pareto Optimal Schedule

Returning to the context of the scheduling problem, we can characterise a schedule Pareto optimal provided we cannot minimise one of its objectives without incurring an expense for another objective, as expected from the definition of a Pareto optimal solution in Section 2.2.

As with any other Pareto relevant problem, we will be hence able to plot all Pareto Optimal schedules on the $f_1, f_2, \ldots, f_n$ plane. In the majority of problems such as those studied in Chapter 4 only Pareto Optimal schedules are of interest. On the contrary a schedule that cannot be placed on a Pareto frontier is of little interest to us. The process is usually to create the Pareto frontier such that the scheduler can observe all the Pareto points and subsequently select the most preferable schedule for their requirements

2.7 System Specifications

All computations are processed with a 6-core Intel Core i7 CPU running at 2.60GHz with a 16GB RAM memory on a macOS Catalina (version 10.15.4) machine. The models were implemented using C++11 that was compiled with the GNU Compiler Collection (GCC). The solutions were obtained through using the commercial solver CPLEX 12.9. Unless it is specifically stated, all solutions obtained are the optimal ones for the given objective function, set of constraints and decision variables.

Chapter 3

Data Exploration and Scheduling Problem Building Blocks

In this chapter we re-introduce the problem assigned to us by Royal Mail in a greater level of detail. Sections 3.1-3.2 walk the reader through a more detailed description of the problem to introduce them to the main elements that constitute the problem as well as give them an idea of Royal Mail's current practices through a series of examples. They also outline the motivation behind solving this problem, and provide arguments justifying why this is a problem worthwhile solving, with great potential for efficiency gains despite the complexity involved in obtaining a solution.

Following the description of the problem, Section 3.3 continues by outlining the structure of the historical data supplied by Royal Mail, and the steps taken to bring the dataset in the form that was utilised in the modelling portion of the project. The **finalised** form of the dataset described at the end of the Data Cleaning section 3.4 constitutes the foundation upon which this dissertation will derive feasible, and efficient schedules. The chapter concludes with Sections 3.5-3.7 describing the processes behind the generation of two unique instances of the problem that will be utilised to conduct more targeted experiments in Chapter 4.

All in all, this chapter gives a more detailed overview of the problem, as well as a detailed description of the dataset provided by our Industrial Liaison, and the steps taken to prepare it for use in our experiments.

3.1 General Context

Each of Royal Mail's **Mail Centres (MC)** upon receiving the mail collected from each **Delivery Office (DO)** will sort it and then decide which portions of it need to be redistributed to DOs belonging to the same MC the following day. Our goal is to structure the delivery itinerary for each MC, for the portion of the post that will remain within the circulation of that MC.

The day before a piece of mail is delivered, the MC responsible for it will have a set of round-trips that need to be completed. At its disposal, each MC has a crew of **HGV drivers**, that will complete those trips. Our objective, is to schedule those trips in a more efficient manner than Royal Mail's current scheduling practices, that minimises the time that drivers spend performing **non-essential** activities that could be performed by lower-grade employees. The process of creating those schedules can be resolved into two sub-components. **Firstly**, decide which trips are completed by each driver. **Secondly**, decide how to sequence the trips assigned to each driver. In allocating trips to a driver we need to respect some restrictions imposed by certain parameters on whether a trip can be allocated to each specific driver, on a particular instant in time. These restrictions are driven by the EU regulations[1] around HGV drivers, in particular the *driving* and *working* time directives. However, for the purposes of this dissertation we neglect those, hence our focus is solely concerned with the task of finding the **room** available **for optimisation**. Following that, it should be straightforward to make the modelling more realistic by introducing those important

[1] Further explained in Appendix D.1

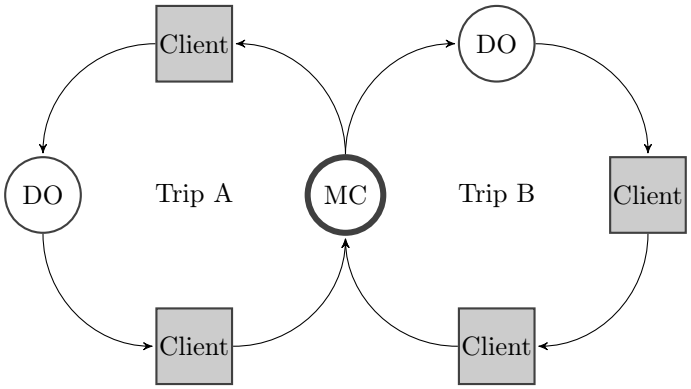


Figure 3.1: An example of a typical duty consisting of two **round-trips** (i.e. atomic blocks), both starting and concluding at the MC.

constraints that ensure the utmost legality of the schedules and it is one of the direction considered for future work[2] on the problem.

As mentioned in the Introduction chapter of the report, we only focus on the **scheduling** aspect of the problem, and consider the routes to be fixed, hence we do not focus on the *Vehicle Routing* point of view.

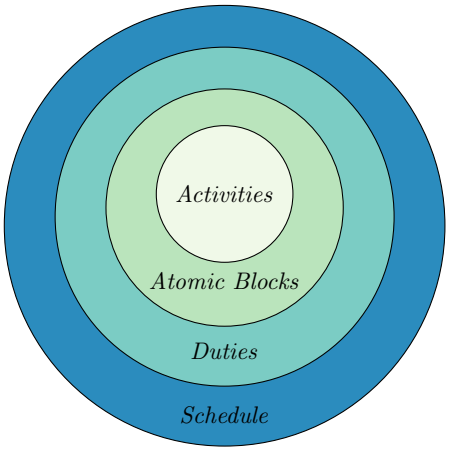
At the start of every working day, a driver is assigned their day's itinerary. The dataset provided to us, contains such historical itineraries for every driver per each day of the week. The task at hand, is to use those historical itineraries to **propose optimised schedules for each day**, which will **be optimal in their use of the drivers' time.**

3.2 Building Blocks

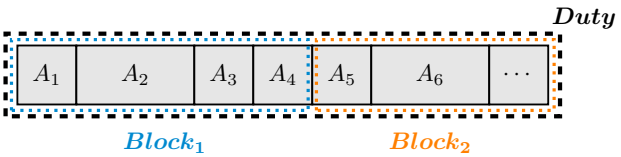
The optimised schedules will be based on historical schedules supplied in the form of a dataset[3] by Royal Mail. The context of the problem involves historical schedules for a **model week** for the Exeter MC. The *definition* of a **schedule** is the collection of duties that are assigned to drivers for each day of the week. Each driver is assigned their **duty** (or else their shift) for the day which constitutes their itinerary for that day. Their duty instructs them to fulfill a number of round-trips each day. Their first trip of the day commences from the Exeter MC, and following a number of visits to external locations their final trip concludes at the Exeter MC, at the end of their day's itinerary. The duties are composed of one or more units of the following two key data points featured throughout the dataset:

- **Activity**: The main *unit of information* signifying the completion of a task. Each activity has a `processing_time` associated with it, informing us of how long the job that it contains takes to be completed. The jobs contained in an activity could be loading/unloading of the mail, driving time between two locations etc[4].
- **Atomic Interval (block)**: The building blocks that make up a **duty**. A block represents a **single round-trip** that commences at the MC and through completing stops at various external locations, concludes again at the MC, see Figure 3.1. Those locations are either Royal Mail's or clients' premises. Mathematically it is defined as a *block of time*, during which a collection of **activities** take place. Multiple such atomic intervals are often featured inside a single duty. The intervals are completely *tightly packed* in the sense that activities are rigidly in place, with no idle time going to waste in between two adjacent activities inside a block. The intervals are characterised as *atomic*, to highlight their firm structure that cannot be broken into sub-components i.e. smaller sets of activities. To move a block, one cannot simply move a portion of it but has to move it in its entirety.

[2]Further explained in Section 7.2 of Chapter 7
[3]Further explained in Section 3.3
[4]A comprehensive list of all the instances of activities is cited in Table B.2 of Appendix B.2.



(a) The set relationship between the various Input Parameters of our problem.



(b) Structure of a Duty, showing *Activities* (A_i) inside *Atomic Blocks* which themselves are inside a *Duty*.

Figure 3.2: Illustrations explaining the nature of the Input Parameters of the problem.

We now focus on two simplified examples that are meant to give the reader a more practical overview of the problem in question. We start off with a dreamt-up example that serves as an illustration that hopefully highlights the foundations of the problem, then transitioning to a more reality-based example that is meant to resemble what typically occurs in a Royal Mail MC.

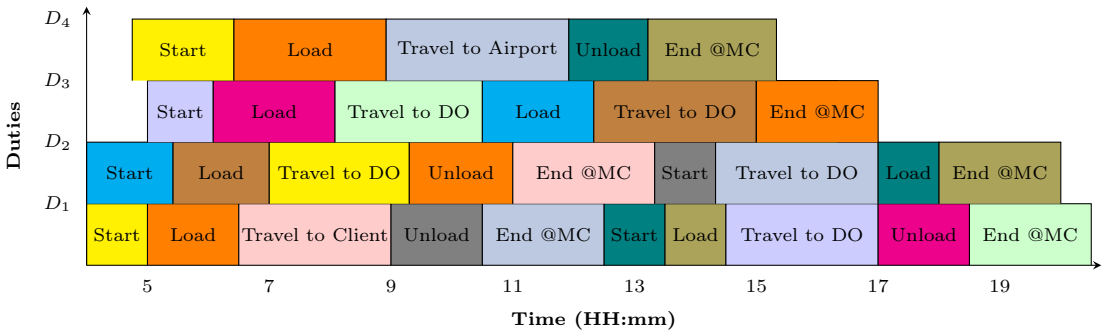


Figure 3.3: Gantt chart of abstract exemplary case.

Generalised Schedule Template

We start with an abstracted example that resembles our problem very closely. The goal of this section is to formalise the problem and give the reader a general sense of the specifics of the problem. We have a set of duties and a set of blocks. In particular, we have 4 duties at our disposal and 6 atomic blocks that require scheduling. The details of the blocks schedule in each of the four duties are seen in the table below:

Block	**Characteristics**			
	Start Time	End Time	Number of Activities	Perfored by Duty
1	04:00	12:30	5	1
2	12:30	20:30	5	1
3	04:00	13:20	5	2
4	13:20	20:00	4	2
5	05:00	17:00	6	3
6	04:45	15:20	5	4

In this particular example we show the process of creating a schedule through an instance of blocks and duties. Namely, the act of scheduling is to place the blocks inside duties in a particular

sequence. In this case that sequence is performed heuristically without the help of an algorithm.

Nonetheless certain interesting facts can be determined from this example. To start with, we can see that the `End` activity incorporates the travel leg from the external location back to the MC. Moreover, we can see that the length of the trip back to the MC is more often than not shorter than that of the trip to the external location. This is observed, because usually the initial travel leg from the MC is conducted during rush hours since the mail needs to be delivered first thing in the morning. In contrast the travel leg back to the MC is not particularly rushed and the driver can determine the best time to complete it.

The typical blocks that occur most often are blocks such as block 1-3. They contain the typical sequence of activities: the `start` to the duty, the `loading` of the mail, the `travel leg` to the external location after the `unloading` of the mail a return back to the MC to `end` the duty. In contrast blocks 4-6 are slightly unique in their sequence of events. Block number 4 for instance is a *container repatriation* trip. The HGV leaves the MC to pick up containers that were previously left at an external location, in order to bring the back to the MC for their use the following day.

Historical Schedule Example

We present an isolated version of our problem that only concerns a small part of the overall problem. The thought process behind this example is to simulate for the reader what typically occurs in a Royal Mail MC. More specifically, we present part of the schedule that dictates the Monday morning operations at a MC. This is illustrated through a Gantt chart in Figure 3.4 which aims to highlight a small snippet of what would occur in a MC at the start of a week.

Taking a more detailed look one can see that maybe this historical schedule is **not so balanced**. The majority of people look like they work for around 8 hours such as the first shift (D_1) that began at 4:00 AM and concluded at 12:00 PM. For confidentiality purposes, the names of locations, clients, and jobs undertaken by each drivers have been omitted and generic placeholder names have been used instead. For instance, shift D_1 consisted of 4 **atomic blocks**, hence 4 **round-trips**. The first atomic block, is the largest one in this duty and consisted of visiting 6 external locations, after the start of the trip from the MC. Following, that 3 smaller atomic blocks that consisted of short-lasting round trips were completed by this driver, until h(er)is end of the duty.

However, taking a more macro outlook on this schedule we can see that there are also people who are required to do overtimes which might be costly for the company, such as shift (D_{14}) that begins at 5:00 AM and lasts until 5:00 PM. This particular shift, consisted of **6 atomic blocks**, where the first one, was also a fairly long-lasting one. It could be possible to obtain a more uniform allocation of blocks per driver, that would result in a more uniform allocation of workload. That is the purpose of the experiments, we then run in Chapter 4.

3.3 Dataset Exploration

Royal Mail have provided historical data from the Exeter MC, that shows the itineraries that the HGV drivers follow over a week. Namely, a **model week** that represents the normal **mode of operation** of a Royal Mail MC over a typical week of the year. The main data points featured in the dataset stem from the Input Parameters outlined in Section 3.2, and below we mention the attributes associated with each *Input Parameter* from within the dataset:

- **Activity**: Each unit of activity is described by a *tuple* $a/b/c/d$, where a, b are the origin and destination locations where the activity takes place, and c, d are its `start` and `end` time. The difference $|d - c|$ therefore, shows the `duration` or `processing time` of the activity.

- **Atomic Interval (block)**: In the dataset, a **block** is constituted by a set of **activities** from table B.2. We identify a block by its first, and last activity, and more specifically merely by the `start`, and `end` time of the `start/end` activity since the locations at the start and end will be the MC itself, by the definition of a block.

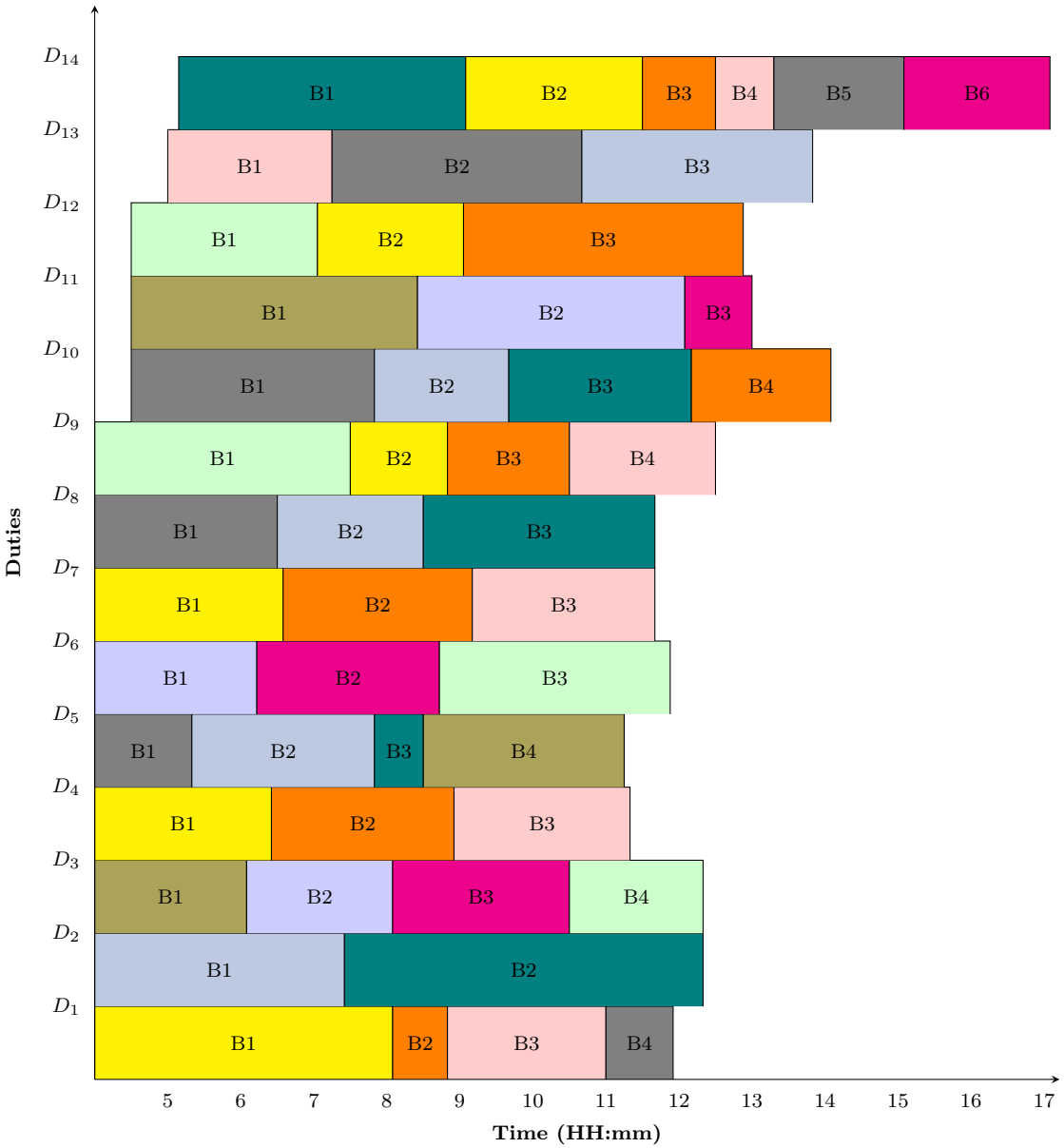


Figure 3.4: Gantt chart showing *Blocks* (B_j) allocated to *Duties* (D_i) as in the Monday morning schedule of the MC.

Schedule	Characteristics			Duties (HH:mm)			Total Time
	Duties	Blocks	Activities	Average	Minimum	Makespan	
Original	200	514	3,555	07:50	02:50	11:50	1,657:51

- **Duty**: The largest unit of time seen in the dataset that consists of a collection of **blocks**. Duties in the dataset correspond to a *driver* taking the *HGV* assigned to them, and completing the collection of **blocks** (i.e. round-trips) required.

The historical schedules in the dataset, are the current feasible itineraries that Royal Mail operates. After careful analysis of the **200** duties in the dataset, we can infer that the Exeter MC serves 30 DOs, 38 client locations with a fleet of 65 total HGVs split between 17, 7.5 tonne lorries, and 3.5 tonne vans. Moreover, we deduce that the Exeter MC opens for business at 4:00 AM in the morning on all weekdays, as well as on Saturday. Sunday is an exception, with a single shift commencing at 8:00 AM and finishing at 6:00 PM in the afternoon. The **absolute latest finishing time** of a shift for weekdays as well as for Saturday is at 3:00 AM in the morning which constitutes the *close of business (COB)* for the day. The tuning of the absolute **earliest starting time** and **latest finishing time** are two of the most important optimisation decisions to be made since, opening for business slightly earlier than the current 4:00 AM time set by company policy, could result in an overall reduction of the duration of the each day's schedule.

Taking a closer look at the idiosyncrasies of the current set of schedules we can extract the following **insights** for each class of activities:

1. `Start/End`: Duties tend to begin in a wave-like fashion. We observed three waves of starting times, with waves occurring near the opening of business, midnight and mid-day[5]. The drivers are split in those three groups, with approximately 59 drivers beginning their shift in the **morning** wave and 61, 63 in the **afternoon** and **night** waves respectively.

2. `Travel`: For each, of the travel activities, we are also supplied with data regarding the mileage covered during each trip as well as, a description regarding the reason for that leg of the trip (e.g. mail, time-sensitive mail, repatriation of mail containers etc.). Leveraging this attribute of travel legs we can see that a certain subset of atomic blocks that contain time-critical tasks occurring within a certain time-span of the day tend to take place at the beginning of a duty. These are the blocks containing the *delivery* of post itself as well as, delivery of time-critical parcels to the airport. In contrast, other blocks that involve non-time-constrained tasks for example *container repatriation* or when an HGV returns with an *empty* cargo bay following a delivery, can be scheduled interchangeably within a shift, and are hence more open to optimisation.

3. `Load/Unload`: A *load/unload* activity takes place, immediately prior and directly subsequent of a travel leg, as illustrated in Figure 3.5. Although this activity could be considered non-useful time as it is time not spent on the road, it cannot nonetheless be optimised away, since company policy states that the driver must be present, and should not be occupied with any other activity in parallel to a `load/unload` of the mail units. Consequently, for the purposes of our optimisation problem we consider this to be **useful** time.

4. `Meal-Relief`: The meal allowance occurs immediately before or after the beginning or end of a block. It is freely interchangeable within the space of a duty but given that it must occur at the premises of the MC, it always occurs at the start or at the end of a block.

5. `Processing/Distribution`: There are cases when a gap occurs between two activities due to the finishing time of the prior activity not completely coinciding with the starting time of the following one. Given that both activities are rigidly scheduled in place and cannot be altered to coincide, drivers are heuristically instructed to fill in the time until their next *useful* activity by assisting with administrative tasks that ideally should be carried out by lower-skilled employees. As a result, unnecessary costs are incurred by Royal Mail due to the

[5]Detailed evidence from the dataset highlighting this phenomenon can be seen in Figure B.1 of Appendix B.3

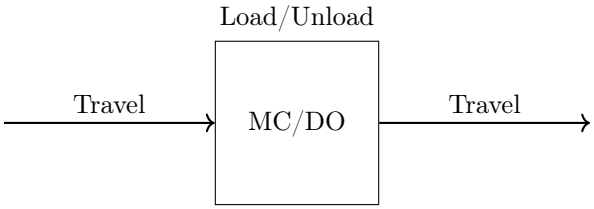


Figure 3.5: Illustration of the pattern of activities that directly succeeds and precedes any travel-leg of an atomic interval.

inefficiency of the schedule. These activities provide us with the opportunity to minimise the makespan of each duty by attempting to reduce their rate of occurrence, as seen in Section 3.5.

6. `Park Vehicle`: The parking activity occurs four times within the dataset whenever a `3.5 Tonne van` arrives at the National Distribution Center. For the majority of duties, the time taken to complete the parking activity is incorporated in the duration of the End activity. It is only in these four duties that parking is included as a separate activity. As a result, we can assume that we can incorporate it ourselves into the `End` activity in those four duties to simplify the dataset as a whole, and equivalently then render the Park Vehicle activity as **non-useful** time.

7. `Check`: They occur as standalone activities in a handful of shifts that are carried out by the `3.5 Tonne Vans`. In the majority of cases they are included in the time that the `Start` activity takes, as explained by the description attached to each activity. As a result, they occur at the beginning of each duty, and tend to take less than 25 minutes. Hence, similarly to the Park Vehicle activity, we can incorporate them into the Start activity, and characterise `Check` as **non-useful**.

8. `Clean`: These take place at the *beginning* or *end* of blocks, and after more critical activities have been completed. For example, if a `Meal-Relief` and a `Clean` are both scheduled to take place exactly at the end of a block, priority will be given to the `Meal-Relief`, since it is a more time sensitive activity. In the majority of cases, it is not entered as a standalone entry and is described as part of the time taken for `Start` or *End* activities.

3.4 Data Cleaning

The quality and effectiveness of the schedules that we hope to generate, will depend significantly on the quality of the dataset from which our schedules will be derived. Consequently, an extensive process to clean the dataset from outliers and general noise is undertaken with the goal of filtering down the schedules to show the most frequently occurring duties which accurately depict what typically occurs in a MC such that an optimisation would really make a difference to Royal Mail.

To retain the reproducibility of the dataset, the specific steps that were taken to clean the dataset are outlined below:

1. **Elimination of unwanted attributes**: The dataset contained numerous attributes attached to each activity, that are utilised in accordance with internal company policy. For instance, some of those attributes contained information regarding the level of driving skills required for each duty. However, our problem operates within the context of *multiprocessor scheduling*, where all machines are considered to be identical, and is hence agnostic to the level of skill of the HGV drivers. We view each task as if it is processed by a machine so we choose to neglect the information provided by those fields and focus merely on scheduling **legal** and **feasible** duties. Consequently, the following fields were eliminated from our dataset `driver_grade, sort_order, date_amended`. A more detailed list of the attributes contained in the dataset can be seen in Appendix B.2.

2. **Incorporation of sparsely occurring activities into `Start` and `End` activities**: Within the dataset, we observed that the parking activity occurs on its own, in only a handful of cases and is instead incorporated into the `End` activity in the majority of cases (as we were informed by the description of the task undertaken, attached to some of the activities). Hence, to avoid focusing on **unwanted** noise cases, we chose to incorporate the duration of the parking activity into the end activity for those cases. In a similar fashion we include, `Clean` and `Check` to the `Start`, and `End` activities as is appropriate.

3. **Standardisation of standalone entries**: In a small subset of cases, an extra level of detail is added onto the duties specifying whether a `load` or `unload` activity is being undertaken as opposed to labelling it as a more general `load/unload` operation. Given, that this extra layer of detail does not concurrently add any extra level of understanding to the problem we choose to label all separate `load` and `unload` operations as a more abstract `load/unload` activity. The same procedure is carried out for separate entries of `Processing`, and `Distribution` activities the distinction of which is similarly of no significance to us.

4. **Neglected duties that involve trips to out-of-scope Royal Mail premises**: Our focus is placed on optimising the scheduling of the routes that occur between the Exeter MC, and the DOs as well as clients for which it is responsible. Hence, duties that involve trips to the *Distribution Centres* and other MCs are out of our scope for optimisation, hence we do not consider trips to those locations.

5. **Elimination of Outliers**: We did not consider entries that involved duties finishing after the *close of business (COB)* as we assumed they involve out of the ordinary situations where duties run late and hence, do not reflect normal business practice.

6. **Neglected data related to Time-Constrained mail**: We ignore data that is related to **time-constrained packages**, as the modelling of such block is out of scope for this project. This decision was made in conjunction with the Royal Mail Data Science group, since the study of such specific trips was not at the time the top priority. The thought process behind this action is that for the purposes of our project we want total freedom as far as moving the blocks and these duties reflect **fixed blocks** hence, represent constraints to our optimisation problem. In practice, to avoid considering such trips occurring in the dataset that was going to be used for the modelling portion of the project, we choose to neglect duties that involve trips to the Exeter Airport, as it is such trips that often have a time-limit attached to them.

7. **Neglected flawed data entries**: In a handful of duties, no presence of an atomic block was observed, instead merely activities were scheduled without a `Travel` leg that constituted the *round-trip* of an **atomic block** being present. Such cases were also considered to be abnormal entries and were not taken into consideration.

The resulting **Finalised Dataset (Cleaned)** that was obtained after subjecting the original dataset to the aforementioned data cleaning processes, schedule the activities in Table 3.1, and has the following characteristics:

Schedule	**Characteristics**			**Duties (HH:mm)**			**Total Time**
	Duties	Blocks	Activities	Average	Minimum	Makespan	
Original	200	514	3,555	08:18	02:50	11:50	1,657:51
Cleaned	183	462	3,285	07:50	02:50	11:50	1,435:22

3.5 Eliminating Redundant Activities

The finalised dataset presented above will be used to solve models that will provide us with a primary level of understanding in regards to the opportunities for optimisation. We then decided to artificially create some more space for optimisation in the historical data by implementing the first step in the procedures implemented on a daily basis by Royal Mail Scheduling operators to create the actual schedules utilised in MCs. That step involves neglecting activities within the atomic blocks that involve the completion of tasks considered **non-useful** time for the drivers. The

Activity	Description
`Start/End`	Indicates the *beginning*, and *end* of a duty.
`Travel`	The *travel leg* from one location to the next.
`Load/Unload`	The *loading* and *offloading* of mail units before leaving or after arriving to a designated location respectively.
`Meal-Relief`	The *meal allowance* break to meet EU *driving time* regulations.
`Distribution/Processing`	Non-essential administrative tasks.
`Park Vehicle`	*Parking* of HGV at end of duty.
`Check`	Scheduled *servicing* of HGV.
`Clean`	Scheduled *cleaning* of HGV.

Table 3.1: List of activities in the **Finalised Dataset (Cleaned)**.

activities, are generally classified in two categories, *useful* and *non-useful*. *Useful* time for *HGV* drivers, from the perspective of Royal Mail, is time related to activities during which, the driver **must be present** and **in control** of the task (e.g. activities involving driving). On the contrary *non-useful* time refers to activities that are put in place as padding time between two successive *useful* activities, that have been schedule to start at a particular time. To fill the time between those *useful* activities drivers end up helping with other activities that are primarily designed for employees of different specialties.

The classification was performed based on the frequency with which activities occurred in the dataset. Sparsely occurring activities were labelled as *non-useful* as they did not carry the vast portion of the workload and were hence deemed non critical. As a result, the activities were distinguished in accordance with Table 3.2. The reasoning behind distinguishing the activities in this manner is to signify which activities must be maintained in place during our quest to allocate as much of the time available within a shift to *useful* activities, (i.e. minimise the *non-useful* time). This operation gives us the necessary space, or else **idle time** within the historical duties to re-schedule the blocks in a more efficient manner. We now have a revised historical dataset, that we refer to as **Redefined Historical Schedules**.

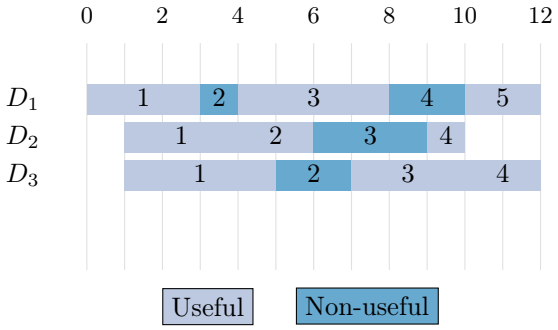


Figure 3.6: Gantt chart showing the split of *useful* and *non-useful* activities inside each block of each duty D_i.

Activity	Useful Time
`Start/End`	✓
`Travel`	✓
`Load/Unload`	✓
`Meal Relief`	✓
`Distribution/Processing`	✗
`Park Vehicle`	✗
`Check`	✗
`Clean`	✗

Table 3.2: List of *useful* or *not* activities.

By deleting theses activities we will observe an initial *step change* reduction in the overall labour hours of the new schedules. This systemic change[6], is obviously not due to the efficiency of the new schedules but due to the deletion go of the redundant activities. As a result, we must not consider it when determining the quality of our new schedules, but actively distinguish it when evaluating the performance of the new schedules. In total, this operation results in a 435 hour reduction of the total labour hours, as well as a 1 hour and 44 minutes reduction in the average duration of a shift. This, systematic reduction of *occupied time* constitutes the space for optimisation that our models can then exploit to provide even more optimal schedules.

[6]Interested readers can see it in more detail in Figure C.1 of Appendix B

Schedule	Characteristics			Duties (HH:mm)			Total Time
	Duties	Blocks	Activities	Average	Minimum	Makespan	
Original	200	514	3,555	08:18	02:50	11:50	1,657:51
Cleaned	183	462	3,285	07:50	02:50	11:50	1,435:22
Redefined	183	462	2,850	06:06	02:00	10:45	1,118:58

Idle Time Gained (HH:mm) per Duty	Overall Time Reduction (hours)
01:44	435

3.6 Departure Waves

The final modification that was applied to the finalised dataset was performed after discussions with our industrial liaison in order to establish a **more practical** and **realistic** direction for our study. To achieve that we decided to implement a new kind of constraint, that limits the space within which we can move a block. This philosophy stems from company policy that is in place to protect the fleet of drivers belonging to a MC, who tend to have set-up their lives around their jobs. Hence, people are in favour of a **consistent starting time**, which means that for instance, people that have historically started their duties in the mornings will be unwilling to change to a night start[7].

For the purposes of our model this translates to assigning a certain *degree of freedom* or else a **slack** to atomic blocks within which they are allowed to move. More specifically, a job that historically started at time t is now given a *window* within which we can change its starting time. That window is the length of time of a **wave**, where waves are the clusters during which duties tend to start. There are three waves observed in the dataset, a `morning, afternoon and night` wave as seen in Figure B.1 of Appendix B.3.

Our quest to model this more pragmatic and realistic version of our problem, motivates the forming of instances of the problem where only a certain wave in itself is concerned. To create those instances, we take the **Finalised** dataset outlined in Section 3.4, and split it into **three sub-instances**, each representing one of the **three waves**, observed in Figure B.1 in Appendix B.3. The effect of splitting our dataset into **three separate,** and **autonomous** instances, is observed in Figure 3.7.

All in all, this results in a minor transformation of our problem. We are still able to freely decide which *round-trip* gets assigned to which duty, but we are only allowed to choose from a smaller pool of blocks from within each wave instance. Hence, in contrast to previous cases, a block is prohibited from being given a `start` time that belongs to a different wave.

[7]This phenomenon was also observed when analysing the dataset in Section 3.3.

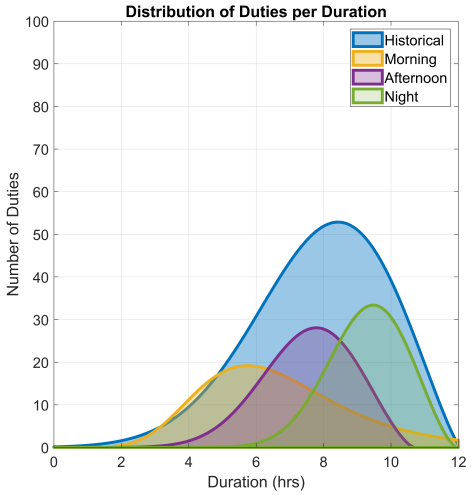


Figure 3.7: The effect of splitting the Historical dataset into **wave instances** of the problem.

Chapter 4

Mixed-Integer Linear Modelling and Analysis of Historical Schedules

This chapter concerns the formulation of an initial set of models used to provide a first assessment on the opportunity for optimisation that is available in our problem. The goal of this analysis is to corroborate the quest of this dissertation to investigate the opportunity for efficiency gains and optimisation of the current practices run by Royal Mail.

Section 4.1 starts by defining the playing field within which our models will be developed. It also provides an overview of the actual historical practices implemented by Royal Mail building upon the high-level overview seen in Chapter 3. Sections 4.2-4.5 each involve the study of a different model, and they all follow a similar organisation. They begin by outlining the motivation behind the use of each of the models, and the goal we wish to achieve with each one. They then present the first set of results of this dissertation, outlining the efficiency gains that can be made for the instances of the supplied dataset.

4.1 Modelling Environment

To develop this first set of model formulations, we assume that the schedule instances contain **fixed duties** with a concrete `start_time`, and `end_time`. Multiple duties are being run in parallel each day. Our goal is to allocate the trips (i.e. atomic blocks) efficiently to those available duties. We assume carte blanche in regards to our flexibility in allocating a block to a duty by neglecting any restrictions as to whether or not we can move a block. We also render our data points agnostic to any semantics associated with each trip[1] and assume that all blocks are equivalent in their significance, and are merely characterised by their parameters. These abstractions will enable us to determine the maximum room for improvement that exists in efficiently allocating blocks to each duty establishing an upper bound in regards to the degree to which we can aim to optimise.

The set of assumptions mentioned are succinctly summarised below:

Simplifying Assumptions

(1)	The **duration** of a **block** stays **constant**, irrespective of the timing of its occurrence.
(2)	The `start`, `end` times and **durations** of duties remain fixed.
(3)	**Atomic blocks** are **freely interchangeable**, within the spectrum of a schedule.
(4)	The **routes** are **fixed**, and we **maintain** the **order** with which we visit external locations.

The first two assumptions, capture the fact that our proposed schedules respect the principle of **conservation of time**. Our models receive as input an instance that contains blocks inside duties

[1] e.g. the distance between locations or the day on which each shift should occurs

the sum of the durations of which gives us the `overall labor time` to be scheduled. The schedules that are generated from our algorithms have the same blocks, covering the same `overall labor time` with the only difference being the **arrangement**, of those blocks in the duties. The third assumptions, refers to our *assignment policy* of blocks to duties, and the fact that it is completely open-ended giving the schedule total freedom over which duty, and at what point in time to assign a block. Finally, the last assumption reiterates that we do not focus on the routing aspect of this Vehicle-Routing like problem but only study its scheduling portion.

Evaluation of Historical Schedules

To begin assessing the performance of our optimised schedule we must first look at the schedules currently run by Royal Mail which we refer to as the **historical** instance.

Instance	**Characteristics**			**Blocks (HH:mm)**		
	`Duties`	`Blocks`	`Activities`	`Average`	`Minimum`	`Maximum`
Historical	183	462	3,285	03:05	00:40	08:25

The historical schedules which our model is given as an input are driven by an instance $I = \langle m, B \rangle$ of the problem where 462 blocks are allocated among a set of 183 duties over the period of a week[2]. The minimal processing time p_j was that of a round-trip lasting only 40 minutes while the longest trips lasted up to 8 hours and 25 minutes. On average a round-trip is expected to last 3 hours and 5 minutes among the historical schedules.

Schedule	**Duties (HH:mm)**			**Total Time (HH:mm)**
	`Average`	`Minimum`	`Makespan`	
Historical	07:50	02:50	11:50	1,435:22

We proceeded to plot a histogram of the duty structure of the historical schedule, seen in Figure 4.1. In that diagram we have fitted the historical workload into class intervals or histogram bins that represent durations of duties according to the duties of the historical schedule. The purpose of this figure is to show us how much each driver works historically. Indeed, there is a **noticeable variation**. We see some drivers work for around 2 hours whereas others for close to 12 hours and in general there are some **significant fluctuations**, which provokes a waste of labour hours. In general, there are people asked to perform duties in excess of 9 hours, hence requiring **overtimes**, which might prove costly for the company. Hence, envisioning a more balanced schedule we wanted to see if it is possible to have a solution that balances the workload more uniformly. This motivated the experiment run in Section 4.2, where since our goal is a balanced schedule we choose a formulation based on the *Makespan Scheduling* model first seen in Section 2.6. The motivation behind this choice is that as was explained in Section 2.6, a schedule with a minimised *makespan* usually achieves a **well-balanced** sequence of the tasks processed [3].

4.2 Load Balancing

With the goal of finding a more balanced schedule in terms of duty length the *Makespan Scheduling* formulation attempts to minimise the overall duration of the schedule, the **makespan**. The goal is for the resulting optimised schedule to decrease the amount of time for which a driver is generally occupied for, a practical advantage that we hope will be gained by a **balanced schedule**.

The two evaluation criteria that we need to examine to determine the success of our model are the degree to which the *makespan is decreased* and a *measure of uniformity* of the optimised schedule. Two ways to practically evaluate these two criteria is to look at the **(%) makespan reduction** and the **(%) standard deviation reduction (σ)** of the distribution of duties, respectively, compared to the instance that was provided as input to the model. Hence, for the experiments involving the

[2] As input we provide the Finalised Dataset (Cleaned) from Section 3.4 of the previous chapter that we henceforth refer to as the Historical Schedule.

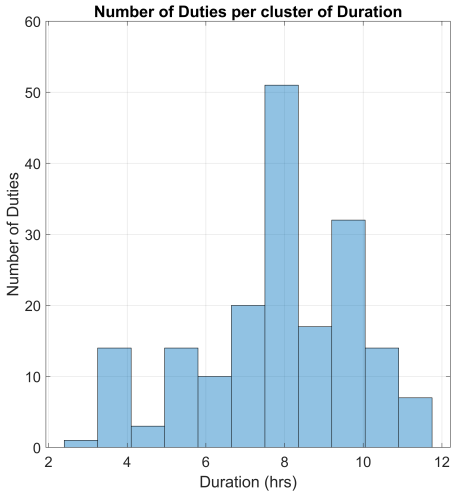


Figure 4.1: The histogram provides an overview of the overall *Duty* lengths featured in the **Historical Schedule**.

Makespan model, we compare every schedule generated at each stage with respect to those two measures of success.

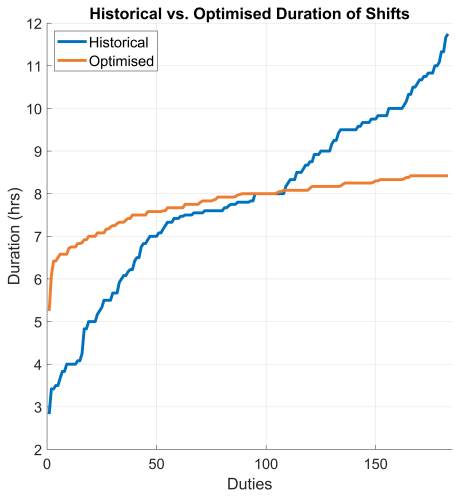
An instance $I = \langle m, B \rangle$ of the problem involves an environment of $D = \{1, ..., m\}$ parallel and identical duties, and a set of blocks $B = \{1, ..., n\}$. Each duty may be assigned at most **one block per unit of time**, and once assigned a block it must execute it until its completion with no interruptions. In other words, this is a *non-preemptive* model. The blocks $j \in B$ are all associated with processing time p_j as their one and only attribute. Mathematically the goal for this first stage of optimisation is to evenly allocate the blocks to each duty in a way that minimises the completion time of the last duty to be completed.

The formulation utilised to perform this task is an adapted version of the makespan scheduling model seen in Chapter 2. We parallelise[3] our problem with the problem observed in Section 2.6. We consider a **duty** as the *machine* from formulation 2.6 and a **block** as a *job*, respectively. Hence, our formulation assigns blocks to duties. Binary variable $x_{i,j}$ dictates the assignment of a block j to a duty i. The binary variable $x_{i,j}$ is equal to 1 if the atomic block $j \in B$ is assigned to duty $i \in D$, and 0 otherwise. Continuous variable y contains the **makespan** of the schedule, which is what we are trying to minimise.

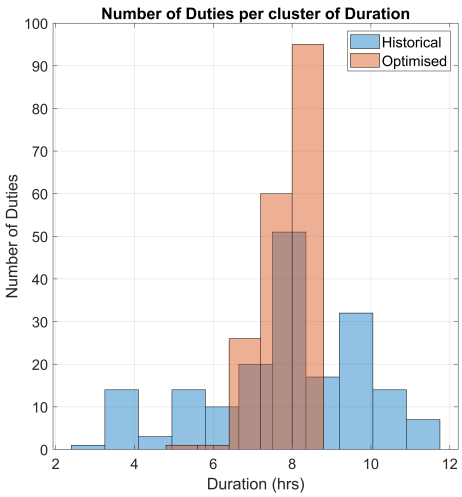
$$
\begin{aligned}
\text{minimise} \quad & y \\
\text{subject to} \quad & y \geq \sum_{j=1}^{n} x_{i,j} p_j && \forall\, i \in D \\
& \sum_{i=1}^{m} x_{i,j} = 1 && \forall\, j \in B \\
& y \geq 0 \\
& x_{i,j} \in \{0, 1\} && \forall\, j \in B,\ i \in D
\end{aligned}
\tag{4.1}
$$

The objective is to get the **minimal** possible **makespan** that allows a feasible schedule. The first constraint assigns y its definition of the **makespan** by making it equal to the completion time of the last atomic block to be executed. With the second constraint we make sure that each block

[3]The purpose of this parallelisation is to transform our problem into an equivalent problem to that studied in [31], for which efficient algorithms for solving it have already been studied.



(a) The duration of each duty before and after optimisation sorted in increasing duration.



(b) Histogram showing the improvement in the **uniformity** of duties.

Figure 4.2: Figures illustrating the effects of the optimisation model on the historical duties.

is assigned to at least one but only one duty. The third and the fourth constraints enforce the continuous and integrality nature to the y, x variables respectively, rendering this a *MILP* problem.

Evaluation

In this initial model, we choose to focus on trying to shorten the **completion time** of the **longer lasting duty**, with the goal of obtaining **fairer shifts** that will have a more **uniform** allocation of the workload among drivers. In the historical schedule, the shortest shift lasts for 2 hours and 50 minutes while the longest one (i.e. the makespan) takes 11 hours and 50 minutes, as seen before. The historical allocation of blocks has been done heuristically, for the historical schedules, which is why cases of longer and shorter shifts than the average tend to occur. Taking as an example, the extremely short and long cases of shifts described above a closer observation shows that more than seven blocks (i.e. round-trips) were allocated to a driver as part of that longest shift, whereas just a single one was assigned to the shortest shift. Such observations motivate our efforts in striving to improve efficiency by minimising the duration of every duty to ensure a fairer allocation of the work load, since there seems to be a chance to remedy all these oddities of the historical schedule.

With regards to the objective of our model, the makespan, we need to be aware of a practical constraint that places a lower bound regarding the most minimum makespan that can be achieved in a feasible schedule. This constraint stems from the fact the there exist blocks of duration 8 hours and 25 minutes. According to assumption (4) of Section 4.1, we cannot split a block into smaller sub-components so as to reduce its duration. Consequently, the `absolute theoretical limit` in regards to the most optimal schedule that we can hope to achieve will have a makespan equal to the duration of those maximum-lasting blocks. Such a schedule would contain duties that solely complete these 08:25 block and no other. Proposing a schedule with an 08:25 makespan or one close to it will constitute an efficient optimisation of the current practices.

Qualitative Results

The application of the Makespan Scheduling model on the Historical dataset has the effect of *flattening* the historical curve as seen in Figure 4.2(a), and hence concentrating the majority of the workload on duties that last as long as the makespan of the schedule, or less.

The effect of our model is most noticeable in Figure 4.2(b) which plots the duration of each duty before and after the optimisation. We can observe how this first attempt at optimising the historical schedule results in a fairer distribution of labor time, since the majority of duties are now of similar

length. There is also a big improvement in the **reduction of extremities** vis-à-vis extremely long and extremely short shifts, allowing us to prevent the occurrence of a large group of very long duties that are not efficient since they require drivers to complete overtimes. In a nutshell a reduction of the order of **72%** in the **standard deviation (σ)** of the optimised schedule compared to the input instance encapsulates the fact that we have achieved our main goal of obtaining a more uniform schedule.

Quantitative Results

Schedule	Duties (HH:mm)			Total Time (HH:mm)
	`Average`	`Minimum`	`Makespan`	
Historical	07:50	02:50	11:50	1,435:22
Historical_Optimised	07:50	05:15	08:25	1,435:22

We have also achieved our main objective of minimising the *maximum duty length* since the historical makespan was 11 hours and 45 minutes whereas in our revised schedule it is exactly 8 hours and 25 minutes hence, achieving the `absolute theoretical limit`. This accomplishment amounts to a **28% reduction of the makespan** compared to the historical instance received as input. In essence, we have redistributed the load[4] so that some people will not need to be working **overtimes** hence resulting in efficiency cost cuts for Royal Mail. A side-effect of our scheduling is that some drivers will be working a bit more. As far as Royal Mail is concerned this is a welcome change since it means we will be utilising the drivers' time better.

It is evident in the historical schedule in Figure 4.2(b) that duties lasting between 8-11.7 hours take a larger load of the work while the rest of the shifts carry out a smaller portion of the overall workload. By making an effort to redistribute the load in a more uniform fashion we are able to achieve a **fairer allocation** of the amount of work required by assigning the load of the long lasting shift to the frequently occurring shifts of up to 8 hours and 25 minutes. We distributed the load from the previously shorter and longer lasting shifts to those frequently occurring ones. The outcome, was an increase in the duration of the shorter lasting shifts resulting in a fairer allocation of the load. However, this also resulted in an increase in the frequency with which the around 8 hour lasting shifts occur. This is an unwanted side-effect of this redistribution of load since even though we reduce the toll taken by the most frequently occurring shift, we distribute a portion of it on the longer-lasting shifts which are already overwhelmed.

A synopsis of the statistics that highlight the performance of our model for the Historical schedule as the input instance is found below:

Schedule	Makespan Reduction (%)	Maximum Difference-Reduction [32] (%)
Historical_Optimised	28%	72%

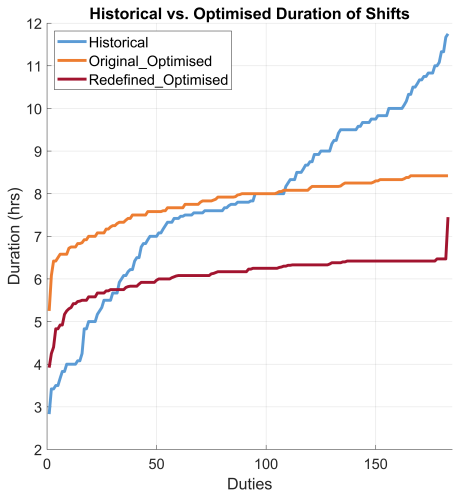
Evaluation - Optimising Redefined Instance

Instance	Characteristics			Blocks (HH:mm)		
	`Duties`	`Blocks`	`Activities`	`Average`	`Minimum`	`Maximum`
Historical	183	462	3,285	03:05	00:40	08:25
Redefined	183	462	2,850	02:25	00:30	06:35

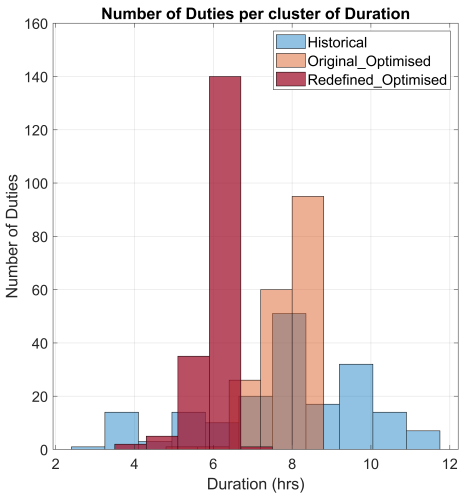
We now apply model (4.1) to an instance of the Redefined[5] blocks and compare the model's performance with the optimal schedule, previously generated. As mentioned in Section 3.5, the

[4]The principle of **conservation of time** is respected as observed on the `Average` and Total Time columns of the table.

[5]Introduced in Section 3.5 of the previous chapter.



(a) The duration of each duty before and after optimisation sorted in increasing duration.



(b) Histogram showing the improvement in the uniformity of duties.

Figure 4.3: Illustrations of curves indicating our results.

Redefined instance is expected to lead us to a better solution, with respect to makespan, since it contains **less activities** and consequently **less labour hours to be scheduled**, due the deletion of the non-useful activities.

Schedule	**Duties (HH:mm)**			**Total Time**
	`Average`	`Minimum`	`Makespan`	
Historical	07:50	02:50	11:50	1,435:22
Historical_Optimised	07:50	05:15	08:25	1,435:22
Redefined_Optimised	06:06	03:55	07:27	1,118:58

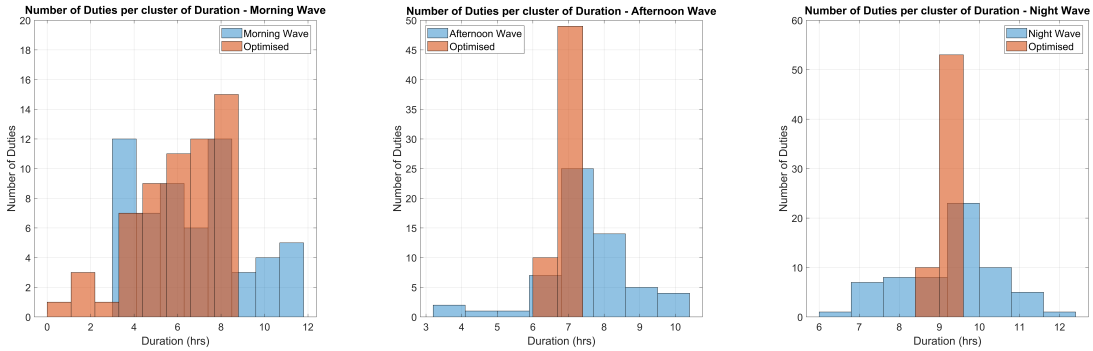
Observing the Histogram chart in Figure 4.3(b), we can see that long lasting shifts have been reduced down to duty lengths which are at the very least, smaller than 7 hours and 27 minutes which is the longest lasting shift (i.e. makespan). The majority of the workload has been assigned to such shifts lasting longer than 6 hours but also generally below the 7 hour mark. With respect to the makespan we see a **36% reduction** compared to the historical instance.

The duty that pushes the makespan up to the 07:27 mark occurs only once, and the reason behind its increased duration relative to the bulk of shorter-lasting duties is because it holds the longest lasting block of 6 hours and 35 minutes. As a result, we could argue that the effective makespan of the Redefined schedule is slightly lower in reality since the majority of the duties last approximately 6 and a half hours. Hence, practically the whole cohort of drivers would have to work for 6 and a half hours, except for a single driver that would have to perform the 7 hours and 27 minutes lasting duty.

In terms of encapsulating the performance of generated schedule in a more concrete manner we list the following table outlining how the optimised schedule fairs against our two evaluation criteria:

Schedule	**Makespan Reduction (%)**	**Maximum Difference-Reduction (%)**
Historical_Optimised	28%	72%
Redefined_Optimised	36%	78%

Comparing the optimised schedules that are created from the Historical and Redefined Instances respectively we see that as expected there is a further reduction of 8% in the makespan of the Redefined schedule, and also the Redefined schedule has a 6% lower standard deviation, capturing



(a) Morning (b) Afternoon (c) Night

Figure 4.4: The effects of Makespan Scheduling when considering our problem in Wave Instances.

the fact that it is even more uniform than the Historical schedule.

Evaluation - Optimising Departure Wave Instances

As a final experiment on the Makespan Scheduling model we break our problem into separate instances that are designed to resemble the observation made through analysing the dataset[6] concerning the fact that duties tend to start in **wave**-like clusters. This observation is an important, design parameter that needs to be taken into serious consideration, as it automatically renders any results that we are able to obtain more *practically oriented* and easier to realise from the perspective of Royal Mail. As outlined in Section 3.6 we split our dataset into three autonomous sub-problems and we once more run the Makespan Scheduling model.

Schedule	**Characteristics**		**Duties (HH:mm)**			**Total Time**
	`Duties`	`Blocks`	`Average`	`Minimum`	`Makespan`	
Historical_Optimised	183	462	07:50	05:15	08:25	1,435:22
Morning_Optimised	59	114	06:46	00:00	08:25	393:40
Afternoon_Optimised	61	145	07:32	03:25	07:19	444:53
Night_Optimised	63	203	09:27	08:40	09:30	596:34

The effects of re-organising our problem in such a fashion can be observed in Figure 4.4. Upon splitting our data into the three wave instances (`morning, afternoon, night`)[7] one can see that overall, the wave sub-instances do not consistently behave either better or worse than the schedule of the whole dataset seen previously in Figure 4.2(b). More specifically, in the `morning, afternoon` waves we can create schedules with a *makespan* of around or below the makespan of the whole instance at the **08:25** mark, hence corroborating the fact that in an ideal schedule for the whole instance, the majority of duties should be of length below the 8 and a half hours mark obtained previously.

It is only in the `night` wave that we observe a sub-optimal makespan, relative to the other two. The `night` wave's makespan is beyond the 08:25 hour mark at 9 and a half hours. A sound argument behind this sub-optimal performance is to observe that the `night` wave has considerably more overall labor time to allocate than the other two instances. Moreover, it only has at most four more duties to allocate that time to, compared to the other two instances. Hence, the combination of more time to be scheduled and the limit amount of flexibility provided by the small amount of extra duties that the model has at its disposal to allocate that additional time to, lead to this sub-par performance with respect to the makespan.

On the other hand, the reason behind the great performance of our model in the `afternoon` instance could be zeroed down to the fact that the `afternoon` wave has the largest timespan out

[6]The `starting-time` analysis of the dataset mentioned is presented in detail in Appendix B.3 for interested readers.

[7]The instances are listed in detail in Appendix C.1

of the three waves (09:00-16:00), as can be seen in Table B.3, of Appendix B.3. Given, that it has a drastically bigger timespan, in fact equal to the sum of the other two instances, it might prove slightly easier to assign very late and very early starting duties since there will be more flexibility due this increased timespan.

Finally, we can see in Figure 4.4(a) that although the `morning` wave performs respectably with respect to its makespan, it lacks in its uniformity of the schedule compared to the other two wave instances, which enjoy a much tighter bandwidth. The non-optimised historical morning wave instance that our model receives as input could be identified as the cause for this lack of uniformity. More specifically, 4.4(a)-(c) we can see that the historical input instance for the `morning` is the least uniform out of the three. More importantly, it is *right-skewed* when compared to the other two, hinting at the existence of numerous shorter-lasting shifts. As a result, our model is largely struggling to allocate the redistributed load from the longer lasting shifts to these shorter-lasting due to their small durations.

These individual findings for each wave sub-instance justify the fact that the most frequently occurring duties in the whole dataset tend to have a duration between 6-8 hours. In addition, it is the shifts of the `night` wave, which contain the biggest portion of schedulable time, that end up contributing to pushing the makespan beyond 6 hours up to the 08:25 mark.

All in all, by approximating the real-world through adding such constraints to our problem that make it more realistic in terms of its ease of implementation, for our Industrial Liaison, we have determined that the efficiency of our schedules is **not** significantly **compromised**. The *makespan*, of the optimal schedule compared to that of the historical one is **equal or smaller** for two of the three waves. In addition, the optimised schedules are much more **uniform** than their historical equivalent, as seen in the table below[8]:

Schedule	**Makespan Reduction (%)**	**Maximum Difference-Reduction (%)**
Historical_Optimised	28%	72%
Morning_Optimised	28%	14%
Afternoon_Optimised	29%	76%
Night_Optimised	19%	80%

4.3 Trade-off between Number of Completed Duties and Maximum Duty Length

Upon proving that there exists significant room for improvement with respect to the historical schedule's makespan we re-examined the historical schedule to see if there are other aspects of it that could be improved. After a careful observation of the dataset we noticed that there are potentially too many duties in the historical schedule. Consequently, people have to work a lot to carry out all those duties. To remedy this, we attempt to minimise the number of duties that are required to complete the same amount of workload (i.e. same amount of blocks).

Provided we are able to decrease the number of duties for the same amount of blocks, this would translate to people working less, since there would be less duties requiring completion. In practical terms this amounts to a transformation of our problem to a Bi-Objective Generalised Assignment Problem [31] since it aims to **minimise** the **maximum duty length**, while also **minimising** the **number of duties**.

To solve this problem we need to develop an algorithm that will provide us with the Pareto Optimal points of the problem. Those points will represent Pareto Optimal schedules, and our Industrial Partners can evaluate each of those schedules to select the one that best fits their company policy requirements. We rely upon the formulation of the previous section but shift the focus of the minimisation problem onto minimising the number of duties that need to be operational to carry

[8]Each wave-optimised schedule is compared with its wave sub-instance for the table's evaluation criteria. Effectively, for each chart of Figure 4.4 we compare the optimised schedule with the input wave instance.

out the required tasks. We hypothesise that we can fix an upper threshold for the duty length (L). We hence assume that people have to work L hours, and not any more than that.

For the purposes of our solver that means that it can schedule duties that are only as long as that threshold. The solver hence, has at its disposal duties that are subject to this upper bound to and has to attempt to fit all the blocks inside of them. In practice this is enforced by creating a constraint that applies this upper bound restriction to the duties' lengths. We hence, take the minimisation of the duration out of the objective function of this problem but it remains in spirit as the secondary objective that we will attempt to minimise through our sensitivity analysis.

The variable y_i contains a value that specifies the number of duties i that are **required**. Similarly to the previous formulation binary variable $x_{i,j}$ dictates the assignment of a block j to a duty i and is equal to 1 if the atomic block $j \in B$ is assigned to duty $i \in D$, and 0 otherwise.

$$
\begin{aligned}
\text{minimise} \quad & \sum_{i=1}^{m} y_i \\
\text{subject to} \quad & y_i \geq x_{i,j} && \forall\, i \in D,\ j \in B \\
& \sum_{i=1}^{m} x_{i,j} = 1 && \forall\, j \in B \\
& \sum_{j=1}^{n} x_{i,j} p_j \leq L && \forall\, i \in D \\
& y \geq 0 \\
& x_{i,j} \in \{0, 1\} && \forall\, j \in B,\ i \in D
\end{aligned}
\tag{4.2}
$$

The model minimises the number of active duties. The first constraint makes sure that a duty i is counted as active if at least one of the $x_{i,j}$ is equal to 1. The second constraint makes sure that each atomic block is executed by one and only one duty. Finally, the third constraint makes sure that the contents of a duty do not exceed the maximum duty length threshold L. This maximum duty length is determined by the hyper-parameter L that we supply as input to the model 36 times to complete the sensitivity analysis.

Evaluation

Using this formulation we investigate the number of active duties that can be reduced for various levels of the maximum duty length. This investigation is carried out by conducting a sensitivity analysis, where we vary the hyper-parameter L and measure the number of duties required. A particularly important threshold that we choose to focus on is that of setting L equal to the maximum duty length observed in the historical schedule. This will allow us to determine the optimal number of duties that could have been used instead of the historically used 183 duties utilised by Royal Mail.

Qualitative Results

In Figure 4.5 we plot the result of the sensitivity analysis. In more detail, through utilising the principle of the Pareto front we computed the efficient frontier seen in Figure 4.5 by applying algorithm (1) 36 times[9] between a minimum and maximum L. Hence, the trade-off curve shows the sensitivity analysis in its entirety applying the Pareto algorithm from a maximum L of 24 hours down to a minimum L of around 8 and half hours, that still provides a feasible schedule. Obviously, we cannot expect an individual to complete duty lasting 24 hours, however, we present the Pareto front to allow the flexibility for our industrial partner. Figure 4.5 shows that if the maximum duty length increases the number of duties required decreases significantly.

[9]Seen in Section 2.2 of Chapter 2

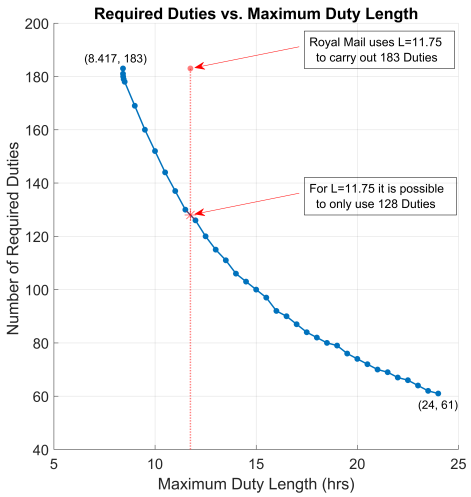


Figure 4.5: Depicts the number of duties required as a function of the Maximum Duty Length varied in 30 minute intervals, up to an *upper threshold* of **24 hours** per duty.

Quantitative Results

As mentioned before although it might not be directly reflected in equation (4.2) this is principally a Bi-Objective problem where the two objectives minimised are the maximum duty length (L) and the number of duties. As result, on the left end of the curve one can see the value of L for which the objective regarding the minimisation of L is best satisfied. As observed in Figure 4.5 that minimal value of L is equal to 8.417 hours[10] and this allows us to complete all blocks in the original 183 duties. Hence, we have identified an optimisation opportunity since it is possible to allocate the same workload into a schedule with **shorter maximum lasting duty** compared to the historical practices. A careful observer will recognise that this exact finding was been seen in an earlier stage of this report. Namely, this value for the maximum duty length, coincides exactly with the value found for the Makespan of the optimal schedule in Section 4.2. Consequently, we can be sure that the schedule with $L = 8.417$ hours is a Pareto optimal schedule that satisfies the minimisation of the maximum duty length objective of this problem.

Moreover, looking at the rest of the curve we can see that we have determined that we are able to reduce the amount of duties required significantly, as we increase hyper-parameter L. Using the L-**threshold** of around 12 hours currently utilised by Royal Mail allows us to use a mere 128 duties compared to the 183 required by Royal Mail. This translates to a **30%** reduction in the duties that need to be fulfilled for the same maximum duty length (L). Practically this means that if a small portion of the drivers is able to complete overtime duties (up to that 12 hours threshold) the remaining cohort of drivers would be able to work less since there would be less duties to be completed.

The realisation that Royal Mail's current operating scenario is not on the Pareto frontier, is another indication that there is a substantial need for optimisation that will most likely lead to opportunities for cost cuts once Royal Mail places itself on the frontier. Taking into account the reality-based constraints that would constrict the provision of more efficient scheduling, we proceed to propose the most optimal but realistic schedule that Royal Mail can aim towards. That practical limit is founded in the fact that the maximum number of labor hours per day for an individual is a total of 13 hours as determined by EU rules[11]. If we proceed to utilise a maximum duty length (L) of 13 hours the Pareto optimal schedule that we can achieve requires merely 115 duties to be completed. This would be a significant decrease over the 183 duties of the historical solution.

Finally, looking at the **theoretical optimal schedule** with respect to the other objective of this problem (i.e. minimal L) we claim in theory we can do even better than a schedule with

[10] Which amounts to 8 hours and 25 minutes.
[11] Seen in Appendix D.1

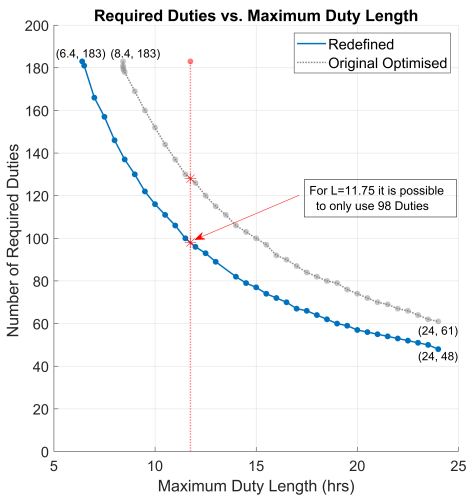


Figure 4.6: Compares the number of duties required as a function of the Maximum Duty Length for the Redefined and Historical instances.

$L = 8.417$ hours. The most optimal schedule that we can hope to achieve is a schedule with a maximum duty length equal to that of the average duty duration of the input instance. That is because such a schedule would be equivalent to spreading the total schedulable hours equally to all 183 available duties. The historical schedules contained 183 duties with an average duty length 7 hours and 50 minutes, equivalent to 1435 hours and 22 minutes of total workload. We can see that the lowest feasible threshold for our maximum duty length in Figure 4.5 is that of 8 hours and 25 minutes. This indicates that there an optimality gap between the ideal maximum duty length and the practical one. This is to be expected, due to the fact that the theoretical optimum could only be achieved if we neglect assumption (1) from Section 4.1. In reality the atomic blocks in the input instance have an arbitrary size and hence we cannot obtain an exact fit of the blocks to all duties that sums to 7 hours and 50 minutes.

Evaluation - Optimising Redefined Instance

We proceed to apply the Redefined instance, to model (4.2) in order to determine whether applying this more realistic instance will further improve the two objectives of our problem. For the historical instance, our optimisation was able to minimise the number of duties by around 30% for an L equal to Royal Mail's historical threshold. When applying the Redefined instance we expect an additional decrease in the number of required duties for the same L. The expectation is founded in the fact that, in this new instance there is less overall schedulable time. Hence, that means that for the same maximum duty length, we can fit more blocks per duty which will directly result in less duties being used overall.

Indeed, were able to carry out the same amount of block with **47%** less duties, namely merely 96 duties as seen in Figure 4.6. This result justifies our quest for optimisation, since it identifies that Royal Mail's current practices are once again outside the *efficient frontier* line.

Observing the Figure 4.6, we can see that this additional reduction in required duties, is mostly due to the step change in *idle time* that we are able to achieve from deleting the *non-useful* activities within the blocks. However, as is shown in Section 3.5 the idle time that we gain is on average 1 hour and 44 minutes per duty. Moreover, we saw in Section 4.2 that the average size of a block is on average 2 hours and 25 minutes for the Redefined instance. As a result of those two facts we can determine why the application of the model on the Redefined yields only an additional small improvement of 17% in the number of duties deleted compared to the 30% improvement obtained in the Historical instance. The explanation is found in the fact despite the increase in idle time there is still not enough space to re-allocate the deleted duties' blocks to other duties, since the average idle time gain (of 01:44) per duty, is significantly smaller than the average size of the block (02:25), hence there are generally not too many blocks that can be relocated to other duties so

that we can delete their original duty.

Once again, if we attempt to determine the **theoretical optimal schedule** for the Redefined instance we would expect to obtain a schedule with maximum duty length (L) of 6 hours and 06 minutes[12]. Given that as we saw in Figure 4.6 we can only go as low as an $L = 6.4$ hours[13] we can see there exist an optimality gap between the schedule with the ideal maximum duty length and the one we can practically obtain. This is once again due to the fact that atomic blocks have an arbitrary size and hence the solver more often than cannot find a duty length that fits all blocks exactly.

4.4 Maximising the Number of Completed Duties with Limited Driver Time

Having created the sensitivity analysis that identified the minimum number of required duties for each value of L we proceeded to solve the directly symmetric problem. More specifically, in Figure 4.2 of Section 4.3 the left end of the curve informed us that the most tight feasible schedule we can possible create is one where the maximum duty length (L) is equal to 8.417 hours. Being aware aware of that we would like to move towards infeasibility and discover the schedules that are to the left of that Pareto optimal schedule with $L = 8.417$ hours.

The reasoning behind conducting this experiment is to determine the effect the decrease of L and the transition towards infeasibility will have on the processing capacity of the MC. More specifically, given that we are moving towards infeasibility, we are aware that we will no longer be able to create a feasible schedule, i.e. a schedule that manages to fit in all blocks, since we have determined that an $L = 8.417$ hours is the least maximum duty length that can fit all the 183 blocks. Hence, further decreasing L will certainly result in not enough schedulable time to fit all the blocks inside the schedule.

The practical purpose of this experiment is to investigate the degree to which we see a degradation in the processing capacity of the MC as we decrease L. In practice to perform this experiment we maximise the number of blocks that can be completed for each value of L, by creating another sensitivity analysis. By decreasing L we obviously will no longer be able to complete everything, so the question posed is what is the maximum amount of work (i.e. number of blocks) that we can manage to accomplish for each L.

$$
\begin{aligned}
\text{maximise} \quad & \sum_{i=1}^{m} x_{i,j} \\
\text{subject to} \quad & \sum_{i=1}^{m} x_{i,j} \leq 1 && \forall\, j \in B \\
& \sum_{j=1}^{n} x_{i,j} p_j \leq L && \forall\, i \in D \\
& y \geq 0 \\
& x_{i,j} \in \{0,1\} && \forall\, j \in B,\ i \in D
\end{aligned}
\tag{4.3}
$$

The model maximises the number of blocks processed, for the given constraints. The first constraint applies the allowance of non-feasible schedules. It allows for a block to not be processed by a duty. The third constraint makes sure that the contents of a duty do not exceed the maximum duty length threshold L as seen in the previous formulation. This maximum duty length is varied to then complete the sensitivity analysis.

[12] Such that L coincides with the Redefined schedule's average duty length seen in Section 3.5

[13] Which amounts to 6 hours and 24 minutes.

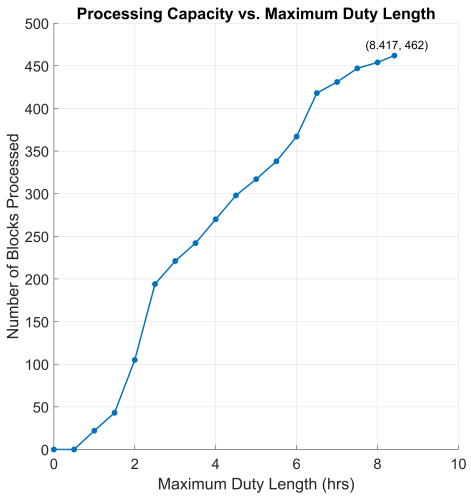


Figure 4.7: Depicts the number of blocks that can be processed as a function of the Maximum Duty Length varied in 30 minute intervals, up to an *upper threshold* that provides the first schedule that is feasible.

Evaluation

Applying equation (4.3) to the **historical** instance and by repeating this process in conjunction with the Pareto algorithm (1) 18 times we receive the curve 4.7 depicting the results of the sensitivity analysis. In the study of this problem we keep the amount of duties fixed to 183 available, and are only concerned with the effects on the processing capacity of the schedule (i.e. the amount of blocks that can fit in those duties).

Qualitative Results

The trade-off curve shows the efficient frontier that we have created in its entirety. The frontier, shows us that as expected while we decrease the value of L the schedules that we are able to create have an ever-decreasing processing capacity since as we can see they are able to process a continuously decreasing amount of blocks. The right end of the curve shows the only feasible schedule featured on this curve. That point with $L = 8.417$ hours is the meeting point between the problem that we are currently solving and its symmetric that was studied in Section 4.3. The symmetry of the two problems is confirmed through this function since the schedule rightmost schedule of Figure 4.7 coincides with the leftmost schedule of Figure 4.5 since they both refer to schedules with 183 duties, 462 blocks processed and an $L = 8.417$ hours.

Quantitative Results

Another interesting finding concerns the relationship between the degradation of the processing capacity as a function of L. As we can see in Figure 4.7 if we reduce the duty length by 2 hours, (**25% reduction**), the decrease in duties that can be performed is only of the order of magnitude of **10%**. This finding is in the direction of obtaining the trade-off relationship between finding the maximum duty length and the effect it has on the processing capacity of the MC. A practical understanding provided by this finding is that if we decrease L by this 2 hour interval i.e. down to around 6 and a half hours, we will not have a proportional decrease in the processing capacity. From the perspective of Royal Mail, this insight could mean that we could allow a small portion of people to work part-time if they so wish. This insight is explained by the fact that this group of part-time employees could carry out that **10%** of blocks that are left unscheduled. Moreover, since they would only work part-time they would work considerably less hours, hence allowing for the decrease seen in L of the order of textbf25%. These insights are not concrete proof that this would work for Royal Mail, but they show that significantly reducing drivers' work time results in a not so significant reduction in the number of blocks that can be processed.

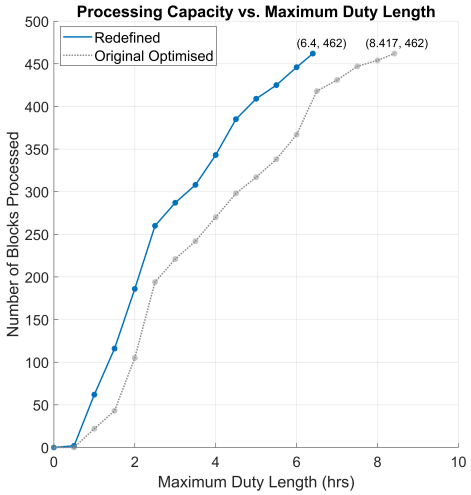


Figure 4.8: Compares the number of blocks that can be processed as a function of the Maximum Duty Length for the Redefined and Historical instances.

Evaluation - Optimising Redefined Instance

In a similar fashion to the case of the results from the Redefined instances in Section 4.3, the application of the Redefined instances provides marginally better results. In more detail, as seen in Figure 4.8, the processing capacity of the schedules generated based on the Redefined instances is hindered ever so slightly less, in comparison to the Historical instance's. That is an expected phenomenon, since as we mentioned before the blocks of the Redefined instances, contain naturally smaller blocks, since they do not consider the redundant activities. As a result, more of them can generally fit in the same 183 duties, which explains the smaller degradation in the processing capacity for the same level of L compared to the Historical instance's schedules.

4.5 Load Balancing with Pre-emptions

The final experiment run in this chapter involves the further investigation of the **theoretical optimal schedule** discussed in the final parts of the evaluations of the prior sections. In all three prior sections, we saw that there exists an absolute theoretical limit that would give us the corresponding **theoretical optimal schedule** with respect to the makespan objective. However, we also saw that the most optimal schedules that our solver was able to come up with would always have a certain optimality gap to that theoretical schedule. That theoretical schedule has been obtained through theoretically hypothesising that we can split the total schedulable labor hours equally in duties of equal length. Unlike, the other schedules discussed in this chapter it has not been obtain as an output from our solver. The purpose of exploring this topic is to investigate how close we can get to that theoretical limit with a schedule generated by our solver if we relax some of the design principles of our makespan formulation.

It is normal to expect that completely achieving the theoretical limit is not a possibility. That is because as explained in the previous sections to obtain that theoretically optimal schedule we would have to completely disregard the structure of our components. Namely, we would have to discretise our activities and blocks with a very high frequency, effectively rendering them a collection of time instants that can exactly fit in duties of any length. However, that experiment is of little utility to us. In investigating how close we can get to approaching the theoretical limit, we choose to only breakdown the size of the blocks, but maintain the structure and processing times of the activities intact. Hence, we still expect not to be able to completely approach the theoretical limit, because cases where activities lengths' do not exactly fit in duties are expected to occur.

To develop a schedule that approaches the theoretical limit we ran the MIP seen in Section 4.2,

but by allowing the model to act **preemptively**. Non-pre-emptive formulations were used in all previous three models. By disregarding this design parameter we grant the model complete freedom in terms of the policy with which it can process each block. Effectively, we provide it with an additional degree of freedom that allows it to switch between blocks while processing one already. Effectively, it is allowed to process activities from different blocks even after having begun the processing of an activity from a different block.

In practice, we carry this out by supplying instances of $I = \langle m, \langle B, A \rangle$ of the problem in an environment of $D = \{1, ..., m\}$ parallel duties. However, in addition to a set of blocks we provide a set of activities $A = \{1, ..., n\}$. The Historical instance utilised contained 3,285 activities that were allocated once again among a set of 183 duties.

Through supplying activities as the component to be scheduled, the model is now **pre-emptive** with respect to blocks, since the solver is allowed to move activities around within each duty hence, breaking the original structure of the blocks. Consequently, we have allowed the model to switch between blocks by allowing it to start processing an activity from one block and then switch to processing an activity from a different block. The activities $j \in A$ all have their processing time p_j as before.

The *MILP* is marginally changed. The biggest change is seen in the constraint: $\sum_{i=1}^{m} x_{i,j} = 1$ which is converted to $\sum_{i=1}^{m} x_{i,j} = p_j$ [3]. Moreover $x_{i,j}$ are no longer binary variables. Variables $x_{i,j}$ now represent the time each block spends on each machine compared to whether machine i executes job j as before.

As a result formulation (4.1) from Section 4.2 becomes:

$$
\begin{aligned}
\text{minimise} \quad & y \\
\text{subject to} \quad & y \geq \sum_{j=1}^{n} x_{i,j} && \forall\, i \in D \\
& \sum_{i=1}^{m} x_{i,j} = p_j && \forall\, j \in B \\
& y \geq \sum_{i=1}^{m} x_{i,j} && \forall\, j \in B \\
& y, x_{i,j} \geq 0
\end{aligned}
\tag{4.4}
$$

The problems objective is once again to minimise the **makespan** while obtaining a feasible schedule. The first constraint assigns y its definition of the **makespan**, since the makespan must be greater or equal to the completion time of the longest lasting machine. With the second constraint we ensure that each job is performed to completion on the various machines that process it. The final constraint ensures that no block is processed for more total time than the makespan itself.

Evaluation

To investigate how close we can approach to the theoretical limit of the maximum duty length, we attempt through this experiment to establish a practical lower-bound solution. As mentioned in the beginning of this section, we do not expect to achieve the `absolute theoretical limit`. But by investigating the effects a preemptive model would have on our results, we can obtain a realisable best possible solution that we can hope to achieve. The `absolute theoretical limit` is that of a maximum duty length equal to the average duty length of the instance, which in the case of the Historical instance is equal to 7 hours and 50 minutes.

The results of the preemptive model hints to a lower-bound for the maximum duty length of 8 hours and 5 minutes. Compared to the makespan of 8 hours and 25 minutes this is a significant improvement in the maximum duty length. In fact this is a **7%** improvement in the optimality gap with the `absolute theoretical limit`.

Evaluation - Optimising Redefined Instance

We repeat the process outlined above to calculate the effect of pre-emption on the Redefined instance. We ran the MILP (4.4) but with an instance containing 2,850 activities that were once again allocated among a set of 183 duties. The `absolute theoretical limit` in this case would be to obtain a schedule where L would be equal to the average duty length of the instance (i.e. 06:06).

Upon running the pre-emptive model on the instance we obtain a practical lower-bound of 6 hours and 32 minutes. Similar to the prior case of the Historical instance this is a significant improvement over the previously obtained makespan of 7 hours and 27 minutes. In quantitative terms this is a bigger improvement compared to the Historical instance's case, of magnitude **23%**.

Chapter 5

Dealing with Uncertainty

Following an initial analysis of the historical schedules and the discovery of significant room for optimisation, we now study the effects that the addition of an **uncertainty component** will have on our schedules.

The chapter begins by outlining the reasoning behind our wish to examine the concept of uncertainty. Section 5.1 begins the study by discussing the procedures followed to generate the uncertainty components. Section 5.2 continues that discussion by analysing the effect the uncertainty components have on our previously optimally generated schedules. It focuses on the robustness of those optimal schedules. Finally, Section 5.3 studies two new methodologies that will hopefully enhance the robustness of our proposed schedule, and compares every optimal schedule that we have generated to determine the most robust, with respect to the uncertainty components.

Our study on uncertainty begins by considering the optimal schedule from the Makespan Scheduling Section 4.2 of Chapter 4. In particular, henceforth we call the **Makespan optimised** schedule from the Historical dataset our **nominal** instance. In essence, it represents what we would like to take place at the MC for the majority of the time. However, for the purposes of this uncertainty study we want to investigate the degree to which this optimal schedule is **robust** with respect to various degrees of **uncertainty**. The purpose of studying this, is to ensure that the schedules generated from our optimisation models will withstand random perturbations that simulate what might occur on a given day at Royal Mail. We argue that robustness of a schedule is dependent upon two optimality criteria. When comparing two schedules, we can characterise as more robust the schedule that achieves the lowest makespan value [33] or the least amount of duties that end up overrunning a maximum duty threshold, defined below. In cases that one schedule performs better in one of the criteria and the other outperforms it in the other criterion, it is left to the scheduler's judgement to determine which schedule is more robust depending on the magnitude of the out-performance in the respective criterion.

5.1 Uncertainty Generation

We generate *disturbed* instances by using the nominal instance as our foundation, to which we then apply uncertainty components of various magnitudes. The uncertainty components are applied

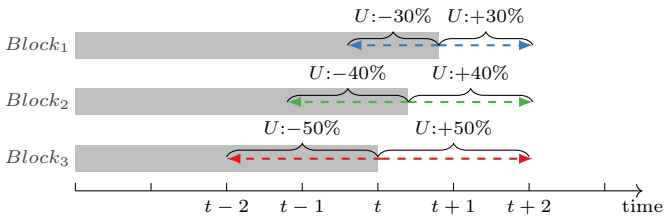


Figure 5.1: The effect of the application of box uncertainty sets of different magnitudes on the duration of blocks.

Instance	Total Time	Atomic Blocks (p_j)		
		`Average`	`Minimum`	`Maximum`
Nominal	1,435:22	03:05	00:40	08:25
U: ±30%	1,416:36	03:03	00:30	10:00
	1,446:14	03:07	00:29	10:12
U: ±40%	1,406:34	03:02	00:30	10:55
	1,439:28	03:06	00:32	11:26
U: ±50%	1,391:17	03:00	00:23	10:44
	1,454:18	03:08	00:20	10:52

Table 5.1: Table showing the overall labor time and other characteristic values of atomic blocks from the two **most disturbed** (`reduced`, `augmented`) instances for each uncertainty set in (HH:mm).

onto each schedule's blocks, which we subsequently will attempt to fit inside the duties. We begin by applying **box uncertainty sets** of magnitudes U : ±30%,±40%,±50% respectively. As seen for example on Figure 5.1, when we apply an uncertainty component of magnitude U: ±30%, each block's *processing time* p_j can randomly **shrink** or **expand** by up to a maximum of 30%, equivalently for uncertainty sets of other magnitudes.

In generating the first set of samples, we created ten disturbed instances per uncertainty set, a total of 30. We chose to showcase the **most disturbed** instances in Table 5.1. We measure the disturbance of the uncertainty set on the instance by examining the degree to which the `Total Time` of the instance has increased or decreased. Hence, for each uncertainty set we present the two instances for which we observed the largest increase, and decrease of the `Total Time` away from the nominal instance's `Total Time` benchmark. We henceforth refer to each group as `reduced` and `augmented`, for reduced and increased *Total time* respectively [21].

The thought process behind choosing the most disturbed instances per uncertainty set, is based on the idea that less disturbed instances will achieve levels of performance in between these two extremes when optimised, hence we can approximate their performance by using those two boundary cases for our analysis.

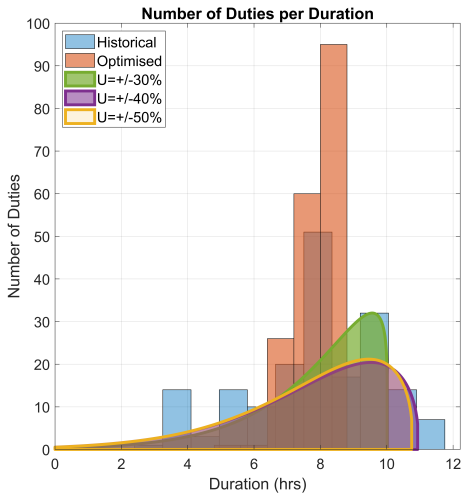
Analysis of the Generated Instances

It is evident in Table 5.1 that the application of the uncertainty components will likely decrease the optimal nature of the nominal schedule, with respect to the makespan. This can be deduced from the fact that the nominal instance contains around 1,435 labour hours whereas in some cases the perturbed instances end up having to schedule more labour hours. By definition the more time that requires scheduling means the tougher for the optimisation solver to fit all blocks in a schedule with a reduced makespan. Hence, this first observation already hints to schedules with generally increased makespans for the disturbed instances.

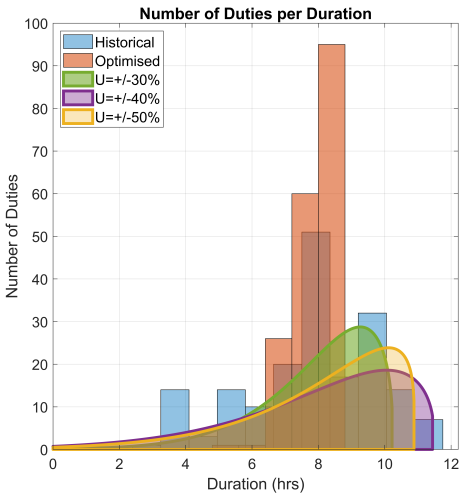
Moreover, as the magnitude of the uncertainty set is increased, we can see that the extremities as far as the durations of the blocks tend to increase accordingly. The increase in the duration of the longest lasting blocks will most likely result in a less robust schedule. Assuming our disturbed instance is **maximally disturbed** as those in Table 5.1, we can see that there exist blocks that have a *processing time* p_j that surpasses the maximum p_j observed in the nominal instance. As we saw in Section 4.2 the maximum p_j of 08:25 in the nominal instance also coincides with the nominal schedule's *makespan*. Hence, the disturbed blocks with `Maximum` (p_j)[1] have durations larger than the nominal instance's makespan. Consequently, it is inevitable, even if the solver encloses only those particular blocks by themselves in one duty, to expect an increased makespan for the disturbed instances over the nominal instance.

On the contrary, looking at the other end of the spectrum we can see that the `Minimum` (p_j) involves blocks with reduced *processing times*. The blocks with shorter lasting durations are potentially going to be helpful in ensuring the uniformity of the optimised schedule since the solver can utilise

[1] Observed in the final column of Table 5.1



(a) Maximally disturbed instances with a **decrease** in overall labor time (`reduced`).



(b) Maximally disturbed instances with an **increase** in overall labor time (`augmented`).

Figure 5.2: The histograms provide an overview of the effects of various levels of uncertainty set on *Duty* lengths.

them as a final buffer to be used in filling any non-utilised idle time at the end of a duty. In simple terms, the blocks with minimal processing times allow the solver greater flexibility in putting the duties together.

5.2 Evaluating the Effect of Uncertainty

Having generated this new set of instances of the problem, our goal is to detect the effects theses various **degrees of uncertainty** have on the optimality of the nominal instance. We hypothesise that there exists a maximum duty threshold L of 8 hours and 25 minutes, that should ideally not be exceeded by any duty under the application of uncertainty. Choosing the two most disturbed instances of Table 5.1 for each uncertainty set, we would like to measure the degree to which the optimised nominal schedule with makespan equal to L, is **disturbed** for each set. The degree of the disturbance is evaluated by measuring the number of duties that overrun this L plus a 15 minute interval, i.e. a **delay of a quarter of an hour**. We refer to this length of time as *L'*. The purpose of this experiment is to gain an insight into the level of robustness of the nominal schedule, with respect to uncertainty. We pass each instance D to the optimisation solver and by applying the **Makespan Scheduling** formulation of Section 4.2, we observe the effects in Table 5.2 and Figure 5.2.

In the figures we can see the Historical Schedule currently run by Royal Mail, as well as the nominal

Schedule	**Total Time**	**Duties (HH:mm)**		
		`Makespan`	`Minimum`	`Average`
Historical	1,435:22	11:45	02:50	07:50
Nominal	1,435:22	08:25	05:15	07:50
U: ±30%	1,416:36	10:00	01:15	07:44
	1,446:14	10:12	00:00	07:54
U: ±40%	1,406:34	10:44	00:00	07:40
	1,439:28	11:26	00:00	07:51
U: ±50%	1,391:17	10:44	00:00	07:36
	1,454:18	10:52	00:00	07:56

Table 5.2: Table showing the duty characteristics and overall time scheduled of the Historical, Nominal and Disturbed Schedules.

and disturbed instances with optimal Makespans. The effects of the application of the uncertainty components are somewhat unsurprising, given our expectations from the analysis of Section 5.1. The optimal schedules of the disturbed instances are **less optimal** than the nominal, with respect to both their **makespan** and their **uniformity**. The makespans of the disturbed schedules that are also listed in Table 5.2 clearly show that the disturbed instances have longer lasting makespans and hence perform sub-par compared to the nominal instance. As mentioned, in Section 5.1 this a completely expected result since the makespans for all disturbed schedules are equal to their longest lasting block, hence it is the `Maximum` (p_j) that is the determining factor for the makespan. As was predicted the degree of the disturbance to the makespan increases roughly as a function of the magnitude of the uncertainty set applied. Perhaps the most interesting observation, however, is that we can see that the schedules with a U: $\pm 40\%$ perturbation performs slightly worse than those with U: $\pm 50\%$ in both cases.

In a similar fashion, the uniformity of the schedules is worsened as we increase the magnitude of the uncertainty set applied. This can be seen qualitatively in Figure 5.2, since the bandwidths of the disturbed instances' distributions are considerably larger than both the historical and nominal's. As is evident the uniformity of the disturbed schedules is tied to the magnitude of uncertainty applied to them, once more noting that somewhat unexpectedly the U: $\pm 40\%$ schedule is even less balanced than the U: $\pm 50\%$, as determined by its larger bandwidth in both cases.

In terms of the effect the uncertainty components have on each duty, we take a look at the schedule on a micro level, and compare duty by duty to L' to measure the effect of the uncertainty. The reason why we are motivated to look at this detail, is because of the practical consequences of such a phenomenon. Were the duties of a disturbed schedule to significantly overrun the L' would mean that Royal Mail would have to incur additional compensation costs for each driver that is required to perform overtime.

Interestingly, we can observe that for these maximally disturbed instances, as we increase the magnitude of the uncertainty set, the number of duties that have surpassed L' does not exactly coincide in all cases. Starting with the `reduced` instances we can see that in the U: $\pm 50\%$ there are less overrun duties even compared to the U: $\pm 30\%$ case. As expected from our observations of the Figure 5.2 the U: $\pm 50\%$ schedules performs better than the U: $\pm 40\%$ with respect to number of duties that overrun. In the `augmented` instances, we can see that the least number of overrun duties are observed in the least disturbed schedule of U: $\pm 30\%$

Schedule	Number of Overrun duties
U: $\pm 30\%$	71
	73
U: $\pm 40\%$	79
	89
U: $\pm 50\%$	68
	82

5.3 Robust Scheduling

To evaluate the robustness of our nominal schedule, we apply the uncertainty instances that generated our disturbed schedules on the nominal schedule. We call this the **recovered** version of the nominal schedule. The motivation behind this study is to determine how well our nominal schedule can perform for the same perturbed instance. To carry out this operation we maintain the assignment of blocks to duties of the nominal schedule and crudely apply the uncertainty sets on each of its blocks, with no regard for the duration of the resulting duty. As one can easily determine this disturbed version of the nominal schedule is going to be far from optimal, since no optimisation principle was applied to it prior to the application of the uncertainty component. It relies on the optimal assignment of blocks to duties at the time that uncertainty had not been applied yet. Having generated these final non-optimised disturbed versions of the nominal schedule we then compare it with the Nominal and Disturbed schedules that have been optimised with the Makespan Formulation. The results are observed in Table 5.3.

Schedule		Optimised	Total Time	Duties (HH:mm)			Overrun Duties
				Makespan	Minimum	Average	
Undisturbed							
Nominal		✓	1,435:22	08:25	05:15	07:50	0
Disturbed							
Instance							
U: ±30%	reduced	✓	1,416:36	10:00	01:15	07:44	71
		✗	1,395:50	10:46	04:35	07:37	44
	augmented	✓	1,446:14	10:12	00:00	07:54	73
		✗	1,452:38	10:46	04:16	07:56	60
U: ±40%	reduced	✓	1,406:34	10:44	00:00	07:40	79
		✗	1,416.47	11:29	04:00	07:44	66
	augmented	✓	1,439:28	11:26	00:00	07:51	89
		✗	1,442:05	11:41	04:34	07:52	68
U: ±50%	reduced	✓	1,391:17	10:44	00:00	07:36	68
		✗	1,386.40	12:15	03:20	07:34	58
	augmented	✓	1,454:18	10:52	00:00	07:56	82
		✗	1,445:31	12:19	03:41	07:53	79

Table 5.3: Table compares optimised[a] and recovered[b] disturbed versions of the Nominal schedule.

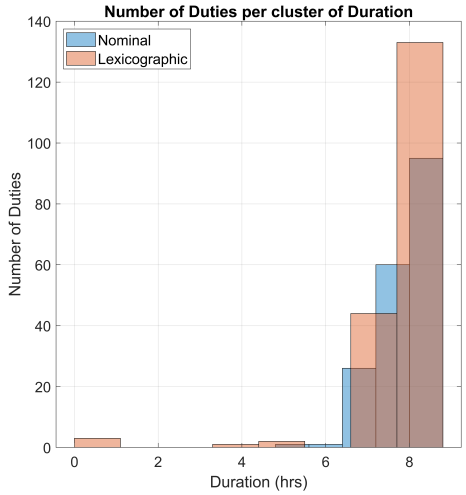
[a]Represented with (✓)
[b]Represented with (✗)

By analysing Table 5.3 we can see that in all of the cases the optimised disturbed schedule outperforms the disturbed nominal schedule with respect to the makespan, as the non-optimised nominal schedule has a longer lasting makespan than the optimised one under uncertainty. However, we can see that on the contrary, the non-optimised nominal schedule has considerably less duties that end up lasting more than L' compared to the optimised schedule under every case of uncertainty. This idea is corroborated also in Figure C.2 of Appendix C.2, where it can be seen that the distributions of the disturbed nominal schedules always have a smaller mean (μ) value compared to the corresponding disturbed but optimised schedule. Consequently, we could argue that our nominal schedule is not very robust with respect to maintaining a low makespan, however, is fairly robust with respect to making sure that not that many duties end up overrunning, which could cost the company dearly in overtime payments.

Having established in the start of this section that our nominal schedule, is not particularly robust with respect to its makespan, we would like to attempt to generate schedules that are significantly more robust to uncertainty. We start by examining two methodologies, that we hope will not have their makespan worsened significantly once uncertainty is applied. We take the nominal instance, that was previously used to generate the nominal optimised schedule, and apply those two new methodologies on it. Effectively, this will act as a re-arrangement of the blocks to duties in a way that respects the protocol of each method. At the end of each section we compare the results of each methodology to the nominal with respect to the two criteria. Their makespan, and the amount of duties that last longer than threshold L'. Our objective is to determine which one of those two methods is more robust once various degrees of uncertainty are applied to it.

Minimisation of the LexOpt Weighted Sum

We start by studying the Lexicographic optimal Scheduling method combined with the *weighting* method with respect to the protocol that generates the schedule. The weighting method, is utilised as a method to implement Lexicographic optimisation which is defined in a more abstract sense, that does not allow the provision of schedules. Hence, by using the weighing method we are able to apply the following formulation to our nominal instance and receive a lexicographically optimised schedule:



(a) Histogram showing the comparison of distributions of block per duty length of the two schedules.

Schedule	Characteristics	
	`Makespan`	`Minimum`
Nominal	08:25	05:15
Lexicographic	08:25	00:00

(b) Table comparing the characteristics of the two schedules.

Figure 5.3: Comparison of the Lexicographic and recovered-(i.e. nominal) schedules.

$$
\begin{aligned}
\text{minimise} \quad & \sum_{i=1}^{m} w_i y_i & & \\
\text{subject to} \quad & y_i \geq y_{i+1} & \forall\, i \in D & \\
& y_i \geq \frac{1}{m-i+1}\Big(\sum_{j=1}^{n} p_j - \sum_{q=1}^{i} -1 y_q\Big) & \forall\, i \in D & \\
& y_i \geq \sum_{j=1}^{n} p_j * x_{i,j} & \forall\, i \in D & \\
& \sum_{i=1}^{m} x_{i,j} = 1 & \forall\, j \in B & \\
& y_i \geq 0 & & \\
& x_{i,j} \in \{0,1\} & \forall\, j \in B,\ i \in D & \\
& \text{where } w_i = 2^{m-i} & &
\end{aligned}
\tag{5.1}
$$

The objective is to get the **minimal** possible weighted sum of completion times. The weights are determined by selecting a value of 2 for parameter M such that we ensure the completion of the algorithm within a maximum time limit of 10^3 seconds [21]. The first constraint applies the lexicographic assignment policy according to which the duties are ordered in a non-increasing order of completion times. With the second constraint we establish a boundary for the completion time of a duty according to the lexicographic optimal scheduling policy [21]. The third and the fourth constraints are carried over from the Makespan scheduling formulation (4.1) used to enforce the feasibility of the resulting schedule. Finally, the fifth and sixth constraints enforce the continuous and integrality nature to the y, x variables respectively, rendering this a *MILP* problem.

We first apply this formulation on our previously nominal schedule, to receive its lexicographically optimised equivalent. Our expectation is that once we apply an uncertainty component on the lexicographically optimised schedule the disturbance will have a smaller effect on it due to its lexicographic nature. We start by generating the lexicographic schedule based on our previously nominal instance and comparing it with it.

As we can see, the lexicographical schedule maintains the same makespan as the nominal, however is quite more tightly concentrated on duties lasting between 6-7 hours, and also has duties lasting

Schedule		LexOpt	Total Time	Duties (HH:mm)			Overrun Duties
				Makespan	Minimum	Average	
Instance							
U: ±30%	reduced	✓	1,395:54	10:49	00:00	07:37	43
		✗	1,395:50	10:46	04:35	07:37	44
	augmented	✓	1,456:10	10:53	00:00	07:57	58
		✗	1,452:38	10:46	04:16	07:56	60
U: ±40%	reduced	✓	1,427:18	12:16	03:38	07:47	64
		✗	1,416.47	11:29	04:00	07:44	66
	augmented	✓	1,444:57	11:45	00:00	07:53	68
		✗	1,442:05	11:41	04:34	07:52	68
U: ±50%	reduced	✓	1,387:40	12:15	00:00	07:34	57
		✗	1,386.40	12:15	03:20	07:34	58
	augmented	✓	1,445:08	12:22	00:00	07:53	79
		✗	1,445:31	12:19	03:41	07:53	79

Table 5.4: Table compares the Lexicographic[a] and recovered[b] disturbed versions of the Nominal schedule.

[a]Represented with (✓)
[b]Represented with (✗)

considerably less, even down to duties of zero length.

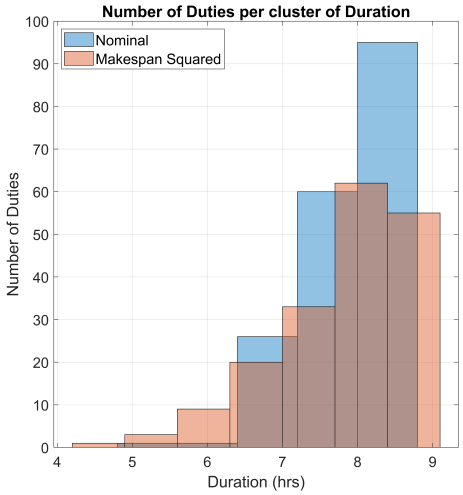
To determine the robustness of the lexicographic schedule to uncertainty, we apply the `reduced` and `augmented` instances for each of the U : ±30%,±40%,±50%, and observe the effects each has on this newly generated schedule, in comparison to the previously observed effects on the nominal schedule.

As we can see from Table 5.4, the Lexicographic schedule performs only marginally better than the nominal schedule in one of the two evaluation criteria. Namely, it performs marginally better in terms of the number of duties overrun since as observed on the final column of the table there are consistently less duties overrun in the lexicographic schedule compared to the nominal. However, it is marginally outperformed in terms of the makespan by the disturbed nominal schedule.

Taking into consideration the above set of results, one can argue that it is questionable to call the Lexicographic schedule **more robust** than the nominal one. However, in reality it depends on the scheduler to decide which of the two criteria is of more significance for Royal Mail. Namely, is it more important for Royal Mail, for drivers to have a shorter maximum duty length L or would they rather have less duties running longer than the pre-defined threshold. Arguably, a pragmatic approach would conclude that it is considerably more important to have a schedule that ensures that the least amount of duties possible end up crossing the L' threshold. That is since the costs that could be incurred from overtime compensation due to a substantial amount of duties overruning are most likely going to be substantially higher compared to a handful of minutes increase in the length of the longest lasting duty (i.e. makespan). Hence, it is our opinion that the Lexicographic schedule can be characterised as more robust due to its lower allowance of duties running very long, despite its longer lasting makespans.

Minimisation of the Squared Completion Time

The secondary method that we would like to check, in our quest to enhance the robustness of our nominal schedule is that of the *Minimisation of the Squared Completion Time*. This is a slightly transformed formulation that stems from the Makespan Scheduling formulation of Section 4.2 of Chapter 4. The reasoning behind the choice of this method is two-fold. Firstly, a makespan scheduling based formulation loosely guarantees a well-balanced schedule. This would most likely translate to a schedule where a minimal amount of duties end up overrunning. Moreover, by slightly tweaking the objective function, and instead of minimising the makespan we refocus our formulation towards minimising its square, we will target the longer-lasting duties significantly more. That is because by minimising its square we effectively penalise significantly more the



(a) Histogram showing the comparison of distributions of block per duty length of the two schedules.

Schedule	Characteristics	
	Makespan	Minimum
Nominal	08:25	05:15
Makespan Squared	09:04	04:34

(b) Table comparing the characteristics of the two schedules.

Figure 5.4: Comparison of the Makespan Squared and Makespan Optimised (nominal) schedules.

longer lasting duties. This idea stems from the philosophy followed in linear regression. The transformed formulation that we will be applying can be seen below:

$$
\begin{aligned}
\text{minimise} \quad & y^2 \\
\text{subject to} \quad & y \geq \sum_{j=1}^{n} x_{i,j} p_j && \forall\, i \in D \\
& \sum_{i=1}^{m} x_{i,j} = 1 && \forall\, j \in B \\
& y \geq 0 \\
& x_{i,j} \in \{0,1\} && \forall\, j \in B,\ i \in D
\end{aligned}
\tag{5.2}
$$

The formulation is identical to equation 4.1 of Chapter 4, except for the objective function. The objective is to get the **minimal** possible squared of **makespan** as opposed to previously simply minimising the makespan.

As one can determine from Figure 5.4 the two schedules as expected resemble each other quite a lot, given that they stem from a principally similar formulation. However, we can notice that Makespan Squared model has a slightly longer bandwidth that also contributes to each longer lasting makespan. We subsequently, apply the disturbed instances on it to see its robustness against those uncertainty components, compared to the nominal schedule.

Through a thorough analysis of Table 5.5, one can determine that the Makespan Squared formulation performs substantially sub-par when compared to the nominal Schedule in both criterion. The makespan is consistently higher, and there are generally more duties that end up overrunning compared to the nominal schedule.

As a result, overall we can conclude that the lexicographically generated schedule is the most robust schedule. That is determined by the results of our experiments and the comparison of the results that show that the recovered version of the LexOpt schedule that makes the least amount of compromise with respect to the minimisation of its Makespan, while achieving the robustness quality since it consistently has the least number of overrun duties.

Schedule		Squared	Total Time	Duties (HH:mm)			Overrun Duties
				Makespan	Minimum	Average	
Instance							
U: ±30%	reduced	✓	1,405:43	11:19	03:57	07:40	83
		✗	1,395:50	10:46	04:35	07:37	44
	augmented	✓	1,457:31	11:36	04:49	07:57	73
		✗	1,452:38	10:46	04:16	07:56	60
U: ±40%	reduced	✓	1,427:18	12:16	03:38	07:47	101
		✗	1,416.47	11:29	04:00	07:44	66
	augmented	✓	1,456:17	12:41	03:31	07:57	97
		✗	1,442:05	11:41	04:34	07:52	68
U: ±50%	reduced	✓	1,392:27	12:54	03:23	07:36	99
		✗	1,386.40	12:15	03:20	07:34	58
	augmented	✓	1,446:46	13:05	03:37	07:54	88
		✗	1,445:31	12:19	03:41	07:53	79

Table 5.5: Table compares the Makespan Squared[a] and recovered[b] disturbed versions of the Nominal schedule.

[a]Represented with (✓)
[b]Represented with (✗)

Chapter 6

Evaluation

This chapter contains a discussion that aims to evaluate the contributions of this report and provide a critical appraisal of the experiments and techniques used.

This dissertation is a continuation of the Red October project which refers to the partnership of the Royal Mail Data Science Group with Imperial's Computational Optimisation Group. The project as the title suggest mostly deals with data analysis processes and concludes with an examination of uncertainty. The original direction of the project focused more on the study of uncertainty. However, after a meeting with our Industrial Liaison at the start of the year both parties realised that a thorough examination of the dataset was going to be a substantially bigger task than originally through. That is because this body of work is the first attempt at analysing the dataset described in Chapter 3. This project's purpose was hence slightly amended, and its new objective became to serve as a stepping stone for future methodological explanations based on the findings of our analysis of the dataset. The high-level objectives of the project resembled part of those of a previous PhD thesis [34], which also included the first exploration of a different dataset provided by Royal Mail.

Those high-level objectives of extracting the maximum amount of insights possible from this dataset were explored in Chapters 3,4. We evaluated the schedules that are currently run by Royal Mail and used them as the basis to develop more efficient schedules. The thought process behind this task was to generate a finding for our industrial liaison that illustrates the room for optimisation that exists. Although our findings are not concrete proof that our schedules would work for Royal Mail since they are based on theoretical assumptions they provide an insight into what is possible in regards to efficiency. Having obtained a lower bound that shows what schedules we are capable of developing, we can then use this insight to gradually create evermore practically realisable schedules by shifting our assumptions towards more life-resembling postulates.

The uncertainty portion of the dissertation attempts to cover new ground with respect to findings that compare the performance of various methodologies. More specifically in Chapter 5 we compare a total of three methodologies with respect to the efficiency and robustness to uncertainty of the schedules they generate. Those are the non-pre-emptive Makespan Scheduling formulation of Section 4.2 as well as schedules from the Lexicographic Optimisation methodology and the Squared Completion time method.

The uncertainty components are based on *box* uncertainty sets of various sizes. Our initial goal was to also explore the effects of the ellipsoidal uncertainty sets[1] however, the study of uncertainty was added as an additional topic towards the end of the project timeline. As a result, although an initial attempt to incorporate ellipsoidal uncertainty sets was made we decided to abandon further pursuing it and instead propose it as one of the future direction that we deemed worthwhile investigating in Chapter 7. However, when providing an appraisal of Chapter 5 we are obliged to mention that this results in a potentially non-complete study of the disturbance effects. It is critical for the uncertainty study to provide meaningful insights to also explore the effects that the ellipsoidal uncertainty sets will have on our schedules, since they could be use approximate the

[1] Mentioned in Section 2.3 of Chapter 2

effects of more complex uncertainty sets as discussed in [35].

Classification of Redundant Activities

As mentioned in Section 3.5 of Chapter 3 we created a series of additional instances that we labelled **Redefined Historical Schedules**. The generation of those instances was performed by deleting a series of redundant activities from the dataset. More specifically, we classified a collection of duties as *non-useful* and proceeded to erase them from within their blocks. The classification of the activities as *useful* or not was not performed in consultation with our Industrial Liaison. Instead we used as a criterion the frequency of occurrence of each activity to determine its level of importance. Using our self-determined classification we conducted the experiments involving the Redefined instances and presented them to our industrial partners in an interim presentation.

During the preparation of the presentation, it was noticed that the classification of the activities might be ill-advised. This observation stems from Table 3.2 in Section 3.5. As we can see we mistakenly classified the activities `Park Vehicle`, `Check`, `Clean` as *non-useful*. However, we later determined that all three of those activities require the presence and control of the HGV driver. As a result, they cannot be characterised as *non-useful*. It is only the `Distribution/Processing` activity that can be regarded as *non-useful*.

Although this observation was uncovered in too late a stage, further research showed that the effect on our results is minimal.

Optimality Gap in Redefined Optimised Schedules

In Section 4.2 of Chapter 4, we conclude our analysis of the Makespan Scheduling formulation by applying formulation (4.1) on the Redefined instances. In observing the results from that experiment we observed that the generated schedules had not achieved their full potential with respect to the minimisation of their makespan. More specifically we observed that the makespan achieved was greater than the longest lasting block observed in the instance that it was based on.

The reasoning behind this under-performance was founded in the fact that during the generation of the schedule the timer upper of 10^3 *seconds* was triggered. Hence, naturally we did not obtain the optimal solution for the case and as a result an optimality gap existed between the true optimal schedule and our result. We decided not to amend that optimality gap by changing the timer trigger because that would render the comparison of all our results inconsistent. However, we could claim that for the purposes of assisting Royal Mail's efforts it might have been advisable to close that optimality gap and obtain the true optimum solution.

Impracticable Pre-emptive Schedules

In Section 4.5 of Chapter 4 we transform our previously utilised formulation of Makespan Scheduling into an equivalent pre-emptive formulation. The purpose of this transformation of our model is to use the transformed model to obtain a an approximation of the **theoretical optimal schedule**.

We were indeed able to reduce the optimality gap to the theoretical limit by applying this pre-emptive formulation. As mentioned in Chapter Chapter 4 however, this formulation violated one of the assumptions outline in Section 4.1. The practical implications of generating a schedule that violates this postulate are quite significant. The structure of such a schedule render it impossible to implement from the perspective of Royal Mail. A pre-emptive schedule is not realistic in terms of its implementation because it would require the HGV drivers to carry out activities in a chaotic sequence.

More specifically, the blocks of an instance that we schedule with each formulation have certain semantics attached to them. As outlined in Section 4.1 we can afford to neglect them as long as the non-pre-emptive nature of our final schedule is preserved. In more detail, the blocks contain certain activities within the spectrum of a round-trip. Within a block those activities are ordered according to a logical sequence. For instance, as we had discussed in Section 3.3 of Chapter 3, we

expect an `unload` activity to occur in direct succession to a `travel` activity. By maintaining a non-preemptive formulation and through obeying assumption (1) of Section 4.1 that logical sequence is preserved. As a result, the practical utility of such a schedule is also preserved. Such a schedule has true value for Royal Mail since it can be implemented on the floor of a MC with little effort, as it is fundamentally based on a re-arrangement of the Historical schedule.

In contrast, the schedules generated from a pre-emptive formulation that consequently violate assumption (1), although more efficient with respect to the objective, have no practical utility for Royal Mail. That observation is based on two pitfalls of the pre-emptive schedule.

Firstly, it would take a tremendous effort of a re-definition of Royal Mail's company policies for such schedules to be allowed. That is because implementing those schedules in practice would require a fundamental re-organisation of the routes followed by drivers. That is since the pre-emptive schedules pay no regard to the reasoning behind the crafting of routes but are only concerned with most efficiently fitting activities into duties. Moreover, since we neglect information regarding distance between the various external locations[2] we could have successive travel legs schedule that are at the complete opposite ends of the area for which the MC is responsible for.

Secondly, the pre-emptive schedules could never be realised in their true form because they tend to schedule futile sequences of activities. More specifically since we allow the pre-emptive schedules to breakdown the pre-defined structure of blocks when observing the sequence of activities schedule by the solver we notice the occurrence of sequence of activities that are impossible. For instance, we observed the successive scheduling of `unload` activities one after the other. As one can easily determine that sequence of activities is completely futile as the second `unload` activity will be scheduled completely in vain. Once again, this phenomenon is observed because we allow our formulation to disobey assumption (1) and hence the semantics associated with each block.

All in all, the approximation for the theoretical limit for the makespan that we obtained from the pre-emptive formulation can only be characterised as a practically obtainable lower bound. It can only be characterised as practical because it can actually be obtained from the output end of a solver but not because it can be practically implemented by our Industrial Liaison.

[2] As mentioned in Section 3.4 of Chapter 3

Chapter 7

Concluding Remarks

The goal of this final chapter is to conclude this report by reiterating the main contributions of this dissertation, and state a collection of promising ideas for future exploration that stem from the work enclosed in this report.

7.1 Conclusion

In this dissertation, we provide an initial exploration of the opportunity for optimisation of schedules based on the dataset supplied by Royal Mail, as well as an initial step towards the development of an efficient but also robust schedule against uncertainty. Our focus with respect to locating the room for improvement that exists in the historical schedule has been focused on the solution of various *Mixed-Integer Optimisation Problems*, motivated in part by the fact that such formulations enable us to optimised various aspects of the historical schedules. Our goal in optimising those aspects of the historical schedules is to provide solutions for improvements inside a sphere of different factors ranging from employee quality of life, to cost-savings for the company.

In Section 4.2 of Chapter 4, the solution of the Makespan Scheduling formulation provides us with a schedule that featured a minimised makespan. This translates to a more balanced schedule that will prevent employees working overtimes hence improving their general quality of life at work. The following model involved a formulation that attempted to minimise the number of duties required to execute the workload at hand. As seen in Sections 4.3-4.4 the solutions provided, outlined a series of schedules that improved on the current schedule both on the employee quality of life and company cost-savings fronts. Namely, the schedules generated by formulation (4.2) minimised the number of duties required. Consequently, the majority of employees could afford to work marginally less, since there were less duties to be completed, and concurrently, Royal Mail could have cost-savings from the reduction in duties since it might possible to place some willing employees to a part-time work schedule. Similarly, formulation (4.3) maximised the number of blocks that can be processed as we decrease the maximum duty length of the schedule. Our findings indicated that we can generally afford a reduction in the maximum duty length without impacting our processing capacity significantly. Hence, provided Royal Mail is able to implement such schedules with a reduction in the maximum duty length we will have generated an improvement in the quality of life of employees as well as possibility for cost-saving by placing employees on part-time duties.

Finally in Chapter 5 we make the first steps towards the formulation of a schedule that is concurrently efficient with respect to the makespan criterion but also remains robust against uncertainty. We start by applying uncertainty components of various sizes to the schedule generated through formulation (4.1) of Section 4.2. We then optimise those schedules with the Makespan Scheduling formulation. We then compare the optimised disturbed schedules with the recovered version of the nominal. We repeat the same process for schedules generated from two new formulations, the lexicographic and completion time squared models. In the end, we compare all three schedules and determine that the schedule that stems from the lexicographic optimisation is the most robust, and makes the smallest compromise in its makespan efficiency.

7.2 Future Directions

We conclude this final chapter by enclosing directions for future exploration that were deemed to be worthwhile for further research, upon conversation with our *industrial liaison*, Royal Mail. These ideas are largely an extrapolation of the work presented in this dissertation in the direction of a more practical application of the ideas developed in this report. The fulfilling of the directions outlined below would render our solutions more realistically applicable from the perspective of Royal Mail, since they involve the addition of practical constraints faced by Royal Mail. In contrast the work of this dissertation is largely theoretical and is hopes to discover the theoretical boundaries for optimisation that exist in the context of the problem that we have studied.

Scheduling Meal-Reliefs

The first direction for future investigation concerns the provision of schedules that comply with the Meal-Relief constraint enforced by EU regulations around **driving** and **working time directives**, as seen in Appendix D.1. Meal-Reliefs are one of the activities observed in the data, as seen in Chapter 3. However, for the purposes of the Meal-Relief constraint, they are considered a special kind of activity since their location inside a schedule dictates whether the overall schedule is compliant with the EU regulations. Consequently, the decision regarding the time that these activities are scheduled involves the addition of a new type of scheduling constraint.

In practice, this involves the addition of two new scheduling constraints to our models, one for the driving time and one for the working time directives. These two constraints will render certain schedules infeasible if they do not satisfy both directives.

Even though it is extremely interesting to explore from an algorithmic point of view, after consulting with our industrial liaison, we determined that it was out of scope for the purposes of this dissertation. Our goal through this project is to provide schedules that explore the boundaries of what is possible in the context of this problem. In contrast, choosing to implement the Meal-Relief constraint would allow us to search for efficient schedules from within a rather narrow subset of the overall feasible set, hence preventing us from approaching the boundaries of global optimally. Moreover, as we were informed by the Royal Mail Scheduling staff, enforcing the Meal-Relief constraint, is largely an *ad-hoc* operation that they tend to perform at the last stage of the schedules' design, which also takes into account the various peculiarities related to each MC's fleet of drivers. For instance, drivers at each MC may be accustomed to taking their Meal-Relief breaks only at specific locations and at specific times within their duty. Hence, enforcing a high-level version of the Meal-Relief constraint that does not take into account those minor but important details would not prove effective in reality. As a result, focusing on further exploring this aspect of the problem without considering such human factors that are not captured by the dataset, and can only be examined within Royal Mail's infrastructure was determined to be not worthwhile pursuing at this stage of the project.

Provided, that we can gain access to this information and consult with each MC's stakeholders to take into account their requests, and implement them in the form of various optimisation constraints, the enforcement of the Meal-Relief constraint can be reduced down to a mere addition of a constraint to our model. The **working time directive** is to be respected by making sure that the **duration** of a *duty* is compliant with the regulation. On the contrary the **driving time directive** is examined at the **block** level, and has to be respected between adjacent blocks. The collection of travel legs featured within each block are those that determine the feasibility component, regarding the **driving time** regulation.

Driving Time Directive

Ideally, it should take place at the end of a shift, at which point the *driving time* limit has been exhausted. However, in the case that prior blocks have consumed a large part of the 4.5 hour driving buffer such that the following block would violate the regulation, the meal relief has to be brought forward, and scheduled between the blocks.

Working Time Directive

Technically, `Meal-Relief` activities, are at the moment actually scheduled inside the blocks featured in our proposed schedules since as explained in the Data Cleaning section (3.4) the `Meal Relief` activity was one of the **useful** activities that we preserved insider our blocks during the Data Cleaning procedures we followed. Hence, as far the implementation of the Meal Relief Constraint is concerned, rendering our *proposed schedules* legal with respect to this constraint is simply a matter of moving the `Meal Relief` activities in the right place within the space of a duty.

Moreover, we anticipate that the addition of these constrains to our models, will make the process of finding an optimal solution less complicated, as opposed to without them since we will be essentially *tightening our feasible space*, hence it will relatively more straightforward to obtain an optimal solution.

Upon conversation with our industrial liaison (Royal Mail), we decided to forego the desire to explore the **Meal Relief** constraint, as it was deemed merely a practical contribution that would not particularly expand the theoretical frontier for what is possible.

Vehicle Routing Considerations

As was discussed in the introductory chapters we do not particularly focus on the *Vehicle Routing* component of the problem. By additionally implementing the *routing* component, we wish to extend the aspects of the problem captured in our model, a critical sub-problem in the majority of distribution network problems.

We briefly investigated a glimpse of the routing aspect of the problem in section 4.5, where the implementation of the pre-emptive philosophy did in fact result into a *re-routing* of the routes followed by drivers. However, numerous experiments that are specifically focused on the solution of the *routing* problem are required to acquire an accurate and insightful understanding of the problem through this point of view.

As explained, in the Data Cleaning section (3.4), we eliminated certain attributes from each data entry of the dataset that were not helpful when looking at the problem from the *scheduling perspective.* As seen in Appendix B.1 one out of those eliminated attributes, consisted of information regarding the distance between locations visited by the HGVs. In solving this problem, one could use this information and form the problem through the concept of a **graph**. *Edges*, of this graph would represent the *time* and *distance* between locations and **nodes** would represent the various external locations themselves. Plotting these graphs, one could then rearrange the **routing** aspect of the problem to make efficiency gains.

Incorporating Deadline Constraints

During the initial data cleaning stages of the project we observed the existence of certain blocks that contained trips that were bound with a deadline with respect to the time of their completion. Namely, such trips involve the delivery of time-sensitive packages. As explained in Section 3.4 of Chapter 3 this packages often refer to trips to the airport, which are by nature time-constrained.

However, as explained in Section 3.4 we decided to ignore such data that is related to **time-constrained packages** and determine it us out of scope for the purposes of our project. The rationale behind this decision is founded in the fact that to satisfy our objective of finding the available room for optimisation we demand total freedom as far as our ability to move the blocks from duty to duty.

Nevertheless, the exploration of such time-constrained trips would be of significant interest both for researchers but also for Royal Mail. For researchers such a task poses great methodological potential. Namely, these time-sensitive round-trips would place certain blocks rigidly inside the spectrum of a duty such that the solver cannot relocate them. In practice, it would be implemented as an additional optimisation constraint. As our result, this would further shrink the feasible set allowing the scheduler less room for the exploration of the optimisation bounds. From the

perspective of Royal Mail, the incorporation such constraints would be highly valuable since it would make our proposed schedules more realistic. Moreover, those schedules would be easier to implement since Royal Mail's operators would not have to heuristically add those constraints and re-order the duties' sequences so that those blocks are placed rigidly in the right place.

Incorporating Ellipsoid Uncertainty Sets

As mentioned in the evaluation chapter (6) due to the fact the study of uncertainty was a late addition to the topics explored we decided not to proceed with the addition of ellipsoid uncertainty set-based components. However, we believe that the study of the effects the application of an ellipsoid-based uncertainty components is a direction for future research that should definitely be tapped.

In greater detail, we believe that the ellipsoidal nature of such uncertainty components would introduce an additional methodological interest to our problem. In greater detail, the addition of ellipsoidal uncertainty sets will transform the study of the robustness of the target model[1] to the process of solving conic quadratic problems [36]. Moreover, the analysis of the effects of ellipsoidal uncertainty sets can be used as a further stepping stone for the approximation of the effects of more complex uncertainty sets[35].

[1]The model to which the uncertainty component is applied

Appendix A

Glossary

These terms are used interchangeably throughout the dissertation, to not tire the reader by using the same term repeatedly.

- Timetable = Schedule = Itinerary
- Duty = Shift
- Break = Meal-Relief
- HGV = Heavy Goods Vehicles = 7.5 tonne lorries
- HGV Driver = Employee
- Model = Formulation
- Maximum Difference [32] = Measure of the difference in the distribution of load between the heaviest and least loaded duties.

Unless stated otherwise, all **time** values are presented in (HH:mm) format.

Appendix B

Dataset Findings

In this section of the appendix, we mention some interesting facts that were observed during the study of the historical schedules that we not deemed important enough to display in the main part of the port. However, we believe that they are useful for reference purposes which is why we outline them below.

B.1 Attributes Featured in the Dataset

In the following section we outline a detailed list of the attributes observed in the dataset as well as a description of the information they provide. As mentioned in Section 3.4 of Chapter 3 we preserve information from only a handful of them and the rest are taken into account for the purposes of this project.

Activity	Description
`Operator`	Indicates the ID of the operator that structured each duty.
`Sort_Order`	Attaches a unique ID to every activity.
`Duty_ID`	Provides each duty with a unique code for identification purposes.
`Date_Amended`	Mentions the date that the particular duty was last modified.
`Commencement Time`	Start time of each activity.
`Ending Time`	End time of each activity.
`Element Type`	Mentions the type of each activity.
`Element Time`	Contains the duration of each activity.
`Due to Convey`	Mentions the purpose of each `travel` activity.
`Vehicle Type`	Mentions the type of HGV vehicle utilised for each travel leg.
`From_Site`	Contains the start location at which each activity occurs.
`To_Site`	Contains the end location at which each activity is completed.
`Driver_Grade`	Mentions the qualification of the driver undertaking each activity.
`Leg_Mileage`	Contains information about the distance (in miles) of each travel leg.

Table B.1: List of the types of attributes featured in the dataset.

B.2 Activities Featured in the Dataset

The activities seen in Table B.2 were those that were observed in the original form of the dataset as was provided to us by Royal Mail. Upon the implementation of the Data Cleaning procedures as seen in Section 3.4 of Chapter 3, the list of activities was transformed to its **Finalised Dataset** form as seen in Table 3.1 of Section 3.4.

Activity	Description
`Start`	Indicates the *beginning* of a duty.
`End`	Indicates the *end* of a duty.
`Travel`	The *travel leg* from one location to the next.
`Load`	The *loading* of mail units before leaving a location.
`Unload`	The *offloading* of mail units after arriving at a designated location.
`Meal-Relief`	The *meal allowance* break to meet EU *driving time* regulations.
`Distribution`	Non-essential administrative tasks.
`Processing`	Non-essential administrative tasks.
`Park Vehicle`	*Parking* of HGV at end of duty.
`Check`	Scheduled *servicing* of HGV.
`Clean`	Scheduled *cleaning* of HGV.

Table B.2: List of the types of activities, as featured in the dataset.

B.3 Starting Times of Duties

As was mentioned in the Data Exploration section 3.3, the duties of each driver tend to start in *clusters*, internally referred to as **waves**. This is clearly observed in Figure B.1 where we have plotted the *starting times* of duties, with the duties sorted in an increasing order.

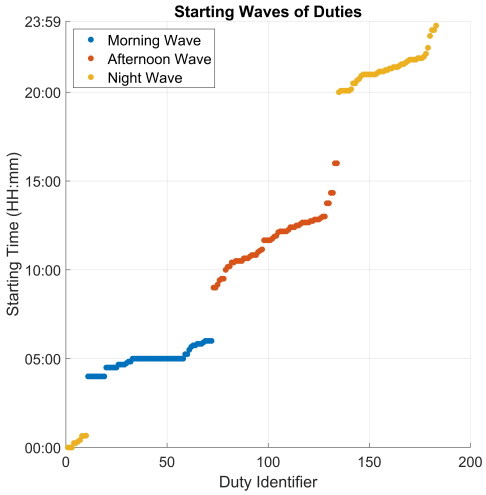


Figure B.1: Plot of the starting times of duties, illustrating the wave like fashion of the starts of duties.

Using Figure B.1 as our guide, we manually characterised each cluster of starting times of duties as **wave instances**, splitting our overall dataset into the **three wave sub-instances** of Table B.3 as outlined in Section 3.6.

As we can see in Table B.3 the waves have practically the same amount of duties, however they differ considerably in their timespan. The `morning` wave is around half the length of the `night` wave, and the `afternoon` is almost equal to the sum of the `morning, night` waves.

B.4 Number of Daily Duties

We thought, it would be an interesting fact to see how many duties occurred each day of the week. This information was deduced from our dataset which contained one week's worth of duties. We plotted the number of duties that occurred each day on Figure B.2. The findings from this plot

Wave	**Characteristics**			
	Start time	*End time*	*Timespan*	*Number of Duties*
`morning`	4:00 AM	6:00 AM	2 hours	60
`afternoon`	9:00 AM	4:00 PM	7 hours	60
`night`	8:00 PM	0:40 PM	4 hours, 40 minutes	63

Table B.3: Table outlining the Starting time, End time and Number of duties of each wave. Time values are stated in (HH:mm) AM/PM units.

would consequently, be indicative of the number of HGV drivers required to carry out those duties per week. Hence, we can infer what Royal Mail's overall fleet of drivers looks like on a given week, since each driver performs one duty per day.

We can observe from the figure that Royal Mail, requires on average around 60 drivers to be active each day of the week. Understandably considerably less drivers are required to carry out the weekend shifts since there is not much activity over the weekend.

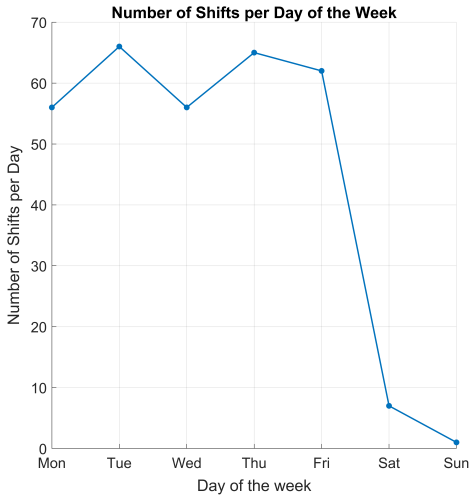
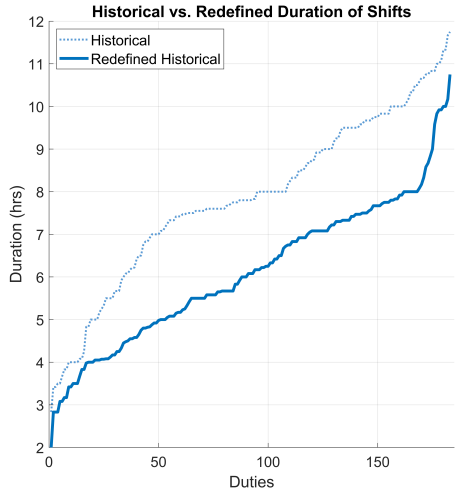


Figure B.2: Plot of the number of shifts occurring each day.

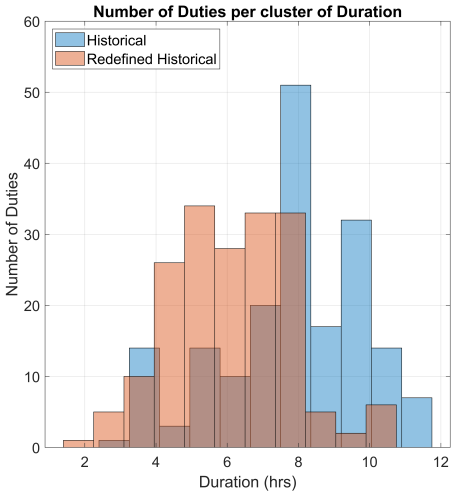
Appendix C

Operations on Historical Schedules

C.1 Redefining Historical Schedules



(a) *Duty lengths* sorted in increasing order, showing the effect of deleting the **non-useful** activities.



(b) Histogram showing the effect of Redefining the dataset.

Figure C.1: Figures illustrating the effects of the Redefining the dataset by deleting the **non-useful** activities from the Historical Schedule.

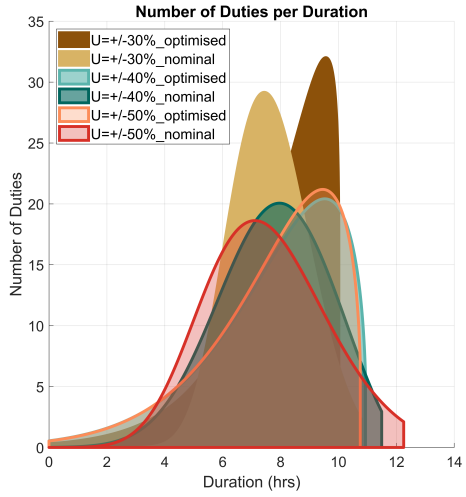
In this section we discuss the effects of the operation carried out in Section 3.5 regarding the deletion of **non-useful** activities from the original dataset. As one can see Figure C.1(a) there is a step change deletion of overall time to be schedule from the Historical dataset, once we delete the non-useful activities. Subsequently, Figure C.1(b) shows the same effect as observed in a histogram graph. The deletion of the non-useful time is observed as a horizontal shift to of the histogram to the left, signifying the fact that less overall hours are not contained in the schedule.

The operation of deleting the activities has a direct impact on the structure of the blocks. Namely, blocks that contain such **non-useful** activities will see their duration decreased. This is seen more practically in the following table:

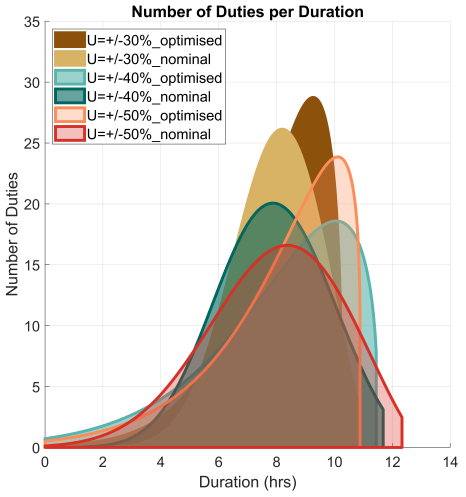
Instance	**Blocks (HH:mm)**		
	`Average`	`Minimum`	`Maximum`
Historical	03:05	00:40	08:25
Morning	03:28	00:50	08:25
Afternoon	03:05	01:10	07:20
Night	02:52	00:40	07:30

C.2 Comparison of Disturbed Nominal and Optimised Schedules

In Figure C.2 our goal is to determine the robustness to uncertainty of the nominal schedule. To identify its level of robustness we compare, for of the three uncertainty sets, the disturbed instance optimised under uncertainty with the nominal schedule disturbed with the same instance. There are two such cases for each uncertainty set, one that involves instances that have been `reduced`, with respect to the overall time scheduled, after the application of uncertainty, and those that have been `augmented` respectively. The former are presented in Figure C.2(a) while the latter in Figure C.2(b).



(a) Disturbed instances with a **decrease** in overall labor time (`reduced`).



(b) Disturbed instances with an **increase** in overall labor time (`augmented`).

Figure C.2: The histograms provide an overview of the effects of various levels of uncertainty set on *Duty* lengths.

Appendix D

Supporting Notes

Tables showing the characteristics of Schedules throughout the report provide measurements of length of time in (HH:mm) units, unless otherwise stated.

D.1 EU Directives for HGV Drivers

This section makes reference to the European Union (EU) rules on drivers' hours and working time as dictated by the Department for Transport (DfT). This is an important real-life aspect of our problem, that is mentioned and referred to, at various points in the report.

(1) Driving-time Directive:

i Time Limit:

- 9 hours daily driving limit.
- Maximum of 56 hours weekly driving limit.
- Maximum of 90 hours fortnightly driving limit.

ii Break:

- 45 minutes break after 4.5 hours driving

(2) Working-time Directive:

i Time Limit:

- Working time must not exceed average of 48 hours a week.
- Maximum working time of 60 hours in one week.
- Maximum working time of 10 hours if night work performed.

ii Break:

- Cannot work for more than 6 hours without a break. A break should be at least 15 minutes long
- 30 minute break if working between 6 and 9 hours in total
- 45 minute break if working more than 9 hours in total

D.2 Relaxation of a Mathematical Program

In order to utilise the efficacy of the simplex algorithm, and apply to solve MILPs we require to obtain the relaxation of an integer-linear program into a LP. A relaxed version of an integer program is defined as below.

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & f(x) \\ \text{subject to} \quad & h_i(x) = 0 \\ & g_j(x) \leq 0 \\ \text{where } x \in S_{original} \end{aligned} \tag{D.1}$$

with the corresponding **relaxed** version of the problem,

$$\begin{aligned} \underset{x}{\text{minimise}} \quad & f(x) \\ \text{subject to} \quad & h_i(x) = 0 \\ & g_j(x) \leq 0 \\ \text{where } x \in S_{relaxed} \text{ and } S_{original} \subseteq S_{relaxed} \end{aligned} \tag{D.2}$$

The relaxation of a problem is usually obtained by the removal of one or more constraints of the original formulation. For example, when obtaining the relaxation of an integer program, we usually refer to the process of neglecting the integrality constraint on the integer program's decision variable(s) to transform our problem into a standard LP.

Bibliography

[1] Luenberger DG, Ye Y. Linear and Nonlinear Programming. vol. 228. Springer Publishing Company, Incorporated; 2015.

[2] Taha HA. In: Operations Research: An Introduction (8th Edition). vol. 8. Pearson Prentice-Hall, Inc.; 2006. p. 81–151.

[3] Pinedo ML. In: Scheduling: Theory, Algorithms, and Systems. 5th ed. Springer Publishing Company, Incorporated; 2016. p. 1–28.

[4] Karmakar N. A new polynomial-time algorithm for linear programming. Combinatorica. 1984;4:373–395.

[5] Faulin J, et al. In: Sustainable Transportation and Smart Logistics. vol. 1. Elsevier; 2018. p. 266–268.

[6] Wedelin D. An algorithm for large scale 0–1 integer programming with application to airline crew scheduling. Annals of Operations Research. 1995 Dec;57(1):283–301.

[7] Wang C, Luh PB, Gribik P, Zhang L, Peng T. The subgradient-simplex based cutting plane method for convex hull pricing. In: IEEE PES General Meeting; 2010. p. 1–8.

[8] Gomory RE. Outline of an algorithm for integer solutions to linear programs. Bull Amer Math Soc. 1958 09;64(5):275–278.

[9] Cornuéjols G. Revival of the Gomory cuts in the 1990's. Annals of Operations Research. 2007 Feb;149(1):63–66.

[10] Balas E, Ceria S, Cornuéjols G, Natraj N. Gomory cuts revisited. Operations Research Letters. 1996;19(1):1 – 9.

[11] Winston WL. In: Operations Research Applications and Algorithms. vol. 4. Thomson Brooks/Cole; 2004. p. 695–706.

[12] Khalilpourazari S, Khalilpourazary S. A lexicographic weighted Tchebycheff approach for multi-constrained multi-objective optimization of the surface grinding process. Engineering Optimization. 2016 08;.

[13] Taheri J, Zomaya AY, Siegel HJ, Tari Z. Pareto frontier for job execution and data transfer time in hybrid clouds. Future Gener Comput Syst. 2014;37:321–334.

[14] Bertsimas D, Brown DB, Caramanis C. Theory and Applications of Robust Optimization. SIAM Review. 2011;53:464–501.

[15] Dokka T, Goerigk M, Roy R. Mixed Uncertainty Sets for Robust Combinatorial Optimization. ArXiv. 2018;abs/1812.04895.

[16] Ben-Tal A, Ghaoui L, Nemirovski A. In: Robust Optimization. Princeton University Press; 2009. .

[17] Li Z, Floudas CA. Robust Counterpart Optimization : Uncertainty Sets , Formulations and Probabilistic Guarantees. Princeton University Press. 2011;p. 1–5.

[18] Gorissen BL, İhsan Yanıkoğlu, [den Hertog] D. A practical guide to robust optimization. Omega. 2015;53:124 – 137.

[19] Bertsimas D, Brown D. Constructing Uncertainty Sets for Robust Linear Optimization. Operations Research. 2009 12;57:1483–1495.

[20] G S. A practical Analysis of Optimisation and Recovery Under Uncertainty. Imperial College London MEng Thesis. 2019;.

[21] Letsios D, Misener R. Exact Lexicographic Scheduling and Approximate Rescheduling. CoRR. 2018;abs/1805.03437.

[22] Liebchen C, Lübbecke M, Möhring R, Stiller S. In: The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications. vol. 5868; 2009. p. 1–27.

[23] Balas E, Fischetti M, Zanette A. A hard integer program made easy by lexicography. Mathematical Programming. 2010 01;135:1–6.

[24] Gupte A. Convex hulls of superincreasing knapsacks and lexicographic orderings. Discrete Applied Mathematics. 2016;201:150 – 163.

[25] Irnich S. A Unified Modeling and Solution Framework for Vehicle Routing and Local Search-Based Metaheuristics. INFORMS Journal on Computing. 2008 05;20:270–287.

[26] Solomon MM, Desrosiers J. Survey Paper—Time Window Constrained Routing and Scheduling Problems. Transportation Science. 1988;22(1):1–13.

[27] Laporte G. The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research. 1992;59(3):345 – 358.

[28] Arkin EM, Hassin R, Levin A. Approximations for minimum and min-max vehicle routing problems. Journal of Algorithms. 2006;59(1):1 – 18.

[29] Coffman EG, Garey MR, Johnson DS. An Application of Bin-Packing to Multiprocessor Scheduling. SIAM J Comput. 1978;7:1–17.

[30] Brucker P. In: Scheduling Algorithms. 3rd ed. Berlin, Heidelberg: Springer-Verlag; 2001. p. 1–178.

[31] Munkres J. Algorithms for the Assignment and Transportation Problems. Society for Industrial and Applied Mathematics; 1957.

[32] Stein C, Zhong M. Scheduling When You Don't Know the Number of Machines. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '18. USA: Society for Industrial and Applied Mathematics; 2018. p. 1261–1273.

[33] Bradley JT, Letsios D, Misener R, Page N. Approximating Bounded Job Start Scheduling with Application in Royal Mail Deliveries Under Uncertainty. In: Li Y, Cardei M, Huang Y, editors. Combinatorial Optimization and Applications - 13th International Conference, COCOA 2019, Xiamen, China, December 13-15, 2019, Proceedings. vol. 11949 of Lecture Notes in Computer Science. Springer; 2019. p. 69–81.

[34] Page N. Vehicle Optimisation in Royal Mail Delivery Offices. Imperial College London PhD Thesis. 2018;.

[35] Bertsimas D, Sim M. The Price of Robustness. Oper Res. 2004 Jan;52(1):35–53.

[36] Ben-Tal A, Nemirovski A. Robust solutions of Linear Programming problems contaminated with uncertain data. Mathematical Programming. 2000 Sep;88(3):411–424.