Models for sequential sorting facility staff allocation

Hamish Thorburn, B.Sc.(Hons.), B.A., M.Res

Submitted for the degree of Doctor of Philosophy at Lancaster University.

July 2024

STOR-i
excellence with impact

Abstract

Sequential sorting facilities are a key step in the courier, express, and parcel delivery industry. In these facilities, staff are assigned to work areas (WAs) to sequentially process different commodities as they move through the facility. When setting the staff levels at these WAs, the shift manager needs to balance different objectives, such as the overall number of staff, the cost of unsorted mail, and how frequently the shift levels change. However, existing literature on staffing these facilities (particularly in the field of mail delivery) focuses on longer timescales, assumes simpler operational constraints, and generally assumes deterministic mail volumes. In this thesis, we develop novel deterministic and stochastic models to staff these facilities for a mail sorting centre. We also propose a framework for general problem-based scenario reduction to use with the stochastic model. The deterministic model is a time-expanded network design model, using staff numbers to increase throughput capacities between WAs. To account for the uncertainty of commodity volumes, we also propose a novel stochastic model. This model is a stochastic programming model where the workplan is the first stage decision, the mail volumes are stochastic, and how the mail is routed over time is the second-stage decision. To solve the stochastic model (and other similar models) more efficiently, we propose a framework to generalise several problem-based scenario reduction methods. We show the applicability of the framework by performing numerical tests using different combinations of candidate solutions and scenario reduction techniques on three different test problems, including the stochastic mail centre staffing problem.

Acknowledgements

Immediately after these acknowledgements, I will declare that I did all the work in this thesis on my own. While technically true, I feel this is grossly misleading.

Firstly, I would like to thank and acknowledge my third supervisor, John Boylan. John sadly passed away in July 2023. John provided excellent insight and guidance into the overall direction and scope of the project. He will be missed.

I cannot put into words how much I have to thank my main supervisors, Anna and Jamie. The amount of support and time I received from them was incredible - whether it was coding sessions, writing sessions, talking down when I'm having a meltdown in one of their offices, or just getting a coffee. I genuinely would not have completed it with any other supervisors.

I also need to thank Tim Flack and Marcos Charalambides, formerly of the unnamed UK-based mail company. Thank you for helping me to make sense of confusing data, rules, and priorities.

Thank you to the rest of STOR-i. The students, especially those in my year - Pete, Matt, Matt, Tamás, Ed, Eleanor, Libby, Tessa, Kes, and Aaditya - for being there for the MRes slog, the COVID weekly teams calls, the cheeky Nandos trips, the drunken walk home from Pete's wedding, and all other support. Specially mention must go to the Dorrington Road boys, for making a grimy, mouldy terrace house a home for a few years, and especially to Ed for always being there for when I need a vent about a relationship crisis, or advice about notation, or just a piece of gum. Thank you.

To the rest of the STOR-i students, those who were before and after me, thank you for making the office more than an office, but a community, a champion running team, and a group of people willing to have a dig. To the STOR-i staff who keep the place running (Kim, Wendy, Nicky and Kate) keep me on STORM (Dan Grose) and for leading the whole program (Jon, Idris, Anna, Rachel, and Kevin), thank you for giving me a reason to come all the way to North-West England, and to stay here for the full (nearly) 5 years.

To the Ligers, whether old (Ben, Pete, Joe, Luke, Jas, Fran, Jake, Taylor) new (Jack, Patrick, Wriggles, Barney, Meg, Izzy, all the Sams, and too many others to name), thank you for being there during the wins, losses, milk-cup games, Ligers report, and for giving me something other than maths to care about in life.

And finally to those from back home. The friends still there (Dave, Karen, Joe, Nat, John), thank you for trekking over to visit, taking phone calls at weird hours, and just checking that I'm still doing ok. To those over here now (Jack, Kate, Courtney, Dan), thank you for the couches to sleep on, the jaunts to the continent, and for picking me up when I was down. And lastly, to my family - Peter, Angela, Maddy, Kevin, Sadie, Angus, and Mikela. Thank you for your far away support, forgiving me for forgetting to call, and making my trips home special.

In short, thank you to everyone who's been part of the PhD. You've helped make this opposite corner of the world feel like home.

Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

Please note a version of the work in Chapter 3 has been published as Thorburn, H., Sachs, A.-L., Fairbrother, J., and Boylan, J. E. (2023). A time-expanded network design model for staff allocation in mail centres. *Journal of the Operational Research Society*, pages 1–16.

This thesis is approximately 44,400 words.

Hamish Thorburn

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

1.1.1 Staffing in sequential sorting facilities

Present in every industry, rostering the required staff and resources is a key management task (Van den Bergh et al., 2013). This is often a balance of assigning enough workers to complete all required tasks in a shift, and not wasting resources by allocating too many staff.

Setting the staff levels is particularly important in *sequential sorting facilities*. In these facilities, different types of material, called *streams* are sorted from each other. These streams arrive at the facility mixed, before passing through a sequential series of *work areas (WAs)* in order to be sorted by a given deadline. Material which is not sorted by this deadline is said to be *delayed*, and subject to a penalty. In each WA, the processing is done by a number of processing units. These can be either machines (staffed by workers) or workers themselves, and are the units that actually sort/process the materials. Each processing unit has a given *throughput*, giving the maximum items it can process in a given time period. If more items need to be sorted, then the shift manager can assign more processing units for that WA, at an additional cost per unit. It

is the job of the shift manager to determine a *workplan* outlining how many processing units need to be rostered in each WA at each time period over a shift. This workplan must balance the cost of the rostered processing units along with the cost for any delayed material at the end of a shift.

These facilities are a central part of the courier, express, and parcel (CEP) industry, which deals with the delivery of letters and parcels from one party to another. This is a vast industry globally, with over $415 billion USD (approximately £341 billion) in revenue worldwide in 2022 (PR Newswire, 2023). Within the UK, Royal Mail (one of the largest CEP companies in the UK) delivered over 10 billion letters and 1.2 billion parcels in the 2018-2019 financial year (Royal Mail, 2019). With these large volumes of items to be sorted, it is imperative that sorting in the sequential sorting facilities (often called sorting centres, distribution centres or mail centres) is done efficiently and in a cost-effective manner. The motivation for this thesis is to determine the optimal method of setting these workplans.

1.1.2 Mail centres

As an example application, we focus on mail centres of a large UK-based mail company (UKMC). The mail centres are the central point of the entire postal delivery system, and where the majority of sorting takes place. In this problem, the different mail types act as the different streams. Each stream generally describes a combination of the class (1st or 2nd class), product size (regular envelope, A4 envelope, parcel, etc), and destination (going to another mail centre, or going to nearby delivery offices).

A small amount of sorting is needed to separate the streams from each other, but the majority of required sorting is to sort items by destination. This is done at the various WAs in the mail centre. A small number of WAs separate the streams, and then the majority sort them into increasingly fine-grained areas of geographical segregation. At most WAs, the sorting is done by automatic sorting machines, each requiring a number

of staff to operate. However, items which cannot be mechanically sorted (for example, the address label is not clear and cannot be automatically read) are sorted by hand in different WAs.

These facilities are complicated by operational factors, including the tethering of WAs (i.e. WAs which cannot start processing material until another WA has completely finished processing all material), splitting of flows of streams, and variation in the amount of material that arrives.

Current literature on organising is somewhat limited. Firstly, existing models for staffing mail centres tend to focus on a longer timescale than is needed to solve this problem (years rather than months) and do not take all operational constraints (such as tethered WAs or split flows of mail) into account. Secondly, the literature assumes that all mail is known before the workplan is made. In reality, mail volumes can vary considerably day-to-day, and are only confirmed after workplans are set.

1.2 Thesis summary

The remaining chapters of the thesis are as follows.

Chapter 2 outlines necessary background information, and reviews the relevant literature. We cover the relevant definitions of network models, stochastic programs, and scenario set selection. We also review the current state of the literature on staff allocation, mail centre optimisation, network models, and scenario set selection.

Chapter 3 shows a staff allocation problem at a sequential sorting facility. In this facility, staff need to be assigned to WAs, through which streams flow sequentially to be processed. Assigning staff optimally involves a trade-off between a number of different objectives, such as minimising the overall number of workers, as well as having fewer changes in the staff levels over time. While optimising for these, many operational requirements need to be met, including minimum processing volumes, correct

ordering/processing of the streams, and not exceeding staff resource constraints.

We develop a deterministic time-expanded network design model to solve the staff allocation problem. The model addresses the problem at a more granular timescale and with more operational constraints than previously used models. We use a lexicographical approach to deal with the multiple objectives. To demonstrate the model's value, we apply it to a staff problem of a UK mail centre, showing that in the majority of cases, our model improves on current staffing practises on both objectives. We also show how the model performs in a number of different environments, including increasing total mail volumes, and changing the proportion of letters and parcels to be sorted.

We extend the deterministic model from Chapter 3 to allow for stochastic mail volumes in Chapter 4. This model is a stochastic programming model based on the deterministic network design problem from Chapter 3, with scenarios to represent potential volumes of streams on different days in the centre. We also examine different weightings of the first and second stage objectives, and allow more flexibility in the workplan in the second stage of the model.

We apply this model to the same mail centre dataset used in Chapter 3. We fit the stochastic programming model using this data, and compare it to using the deterministic model when assuming the mail volumes take increasingly higher multiples of expected demand. We find that when 20 or more scenarios are used for the stochastic programs, our the stochastic programs result in lower expected costs than the deterministic models. We also investigate the weightings between the first and second stage objectives of the stochastic programs, during which we found an appropriate balance between the two. Finally, we showed that allowing flexibility in the second stage of the model results in lower first and second stage costs.

One of the insights we gain from Chapter 4 is that the model could benefit from using more sophisticated scenario reduction techniques. Chapter 5 shows our work on improving these techniques. Finding representative scenario sets for stochastic pro-

gramming is key for finding good quality solutions within a reasonable time. Recent work advocates for the use of 'problem-based' scenario reduction techniques - that is, techniques which use information about the current problem to select scenario sets. However, these techniques are often tailored to specific problems.

We instead develop a framework to generalise a number of existing problem-based scenario reduction methods. The framework consists of creating a number of candidate solutions, testing each existing scenario on each candidate solution, to create a 'recourse matrix', and using the recourse matrix in an existing scenario reduction technique. We use this framework to test a number of combinations of candidate solutions with a number of existing scenario reduction methods on three different problems, showing some insights into which combinations prove better than others.

The conclusions of our work are discussed in Chapter 6. Here we outline how our work contributes to both staff allocation (Chapters 3 and Chapter 4) and scenario set selection (Chapter 5) fields. We also discuss future avenues for research which have been opened by our work.

Chapter 2

Theoretical background and relevant literature

2.1 Introduction

In this chapter, we introduce the theoretical background and foundation for the models developed in the later chapters. We then provide an overview of the literature on extensions of fundamental models to identify research gaps.

We identify three main streams of literature that are particularly relevant for this research, namely network design models (Section 2.2), stochastic programming (Section 2.3) and scenario set selection (Section 2.4). In Sections 2.2.4 and 2.3.4, we also consider literature on network design models and stochastic programming especially for the staffing problem in mail centres.

We start by defining networks, as well as extensions such as time-expanded and multi-commodity networks. We use these definitions to present the minimum cost flow problem, as well as the multi-commodity and network design extensions. We discuss common solution techniques to this problem, before reviewing how these problems relate to current literature on staffing in mail centres. Network design models serve as

theoretical basis both for Chapters 3 and 4.

In Section 2.3, we provide background on stochastic programming, which will be useful for Chapters 4 and 5. In addition to identifying basic properties and solution approaches, we also review literature on how stochastic programming compares to other approaches in accounting for randomness in mail volumes.

We conclude with Section 2.4, where we discuss scenario set selection in more detail when using the approaches from Section 2.3. We cover literature on evaluating scenario sets and review popular scenario set selection techniques. This provides important background for our work in Chapter 5.

2.2 Network flow models

2.2.1 Overview

In this section we provide an overview of network flow models, and how they can be used in staff allocation. Network flow models are concerned with sending units of flow optimally across a network. This could be, for example, sending units of a product across a distribution chain, or sending generated electricity through a power grid. For the purposes of this thesis, we define a network as follows, using a very similar definition as Ahuja et al. (1994):

Definition 2.2.1. *A network is a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ which consists of:*

- *A set of* nodes $\mathcal{N}$ *(indexed by n). Associated with each node is a demand b_n, representing the demand for the commodity at that node. We say that a node with $b_n > 0$ is a* source *node, a node with $b_n < 0$ a* sink *node, and nodes with $b_n = 0$ transshipment nodes.*

- *A set of directed* arcs $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$*, indexed by a. The direction of the arc indicates the direction that flow is allowed to move across the arc. Each arc will have a*

cost c_a, and a capacity u_a.

For convenience, we also define the sets $\delta^+(n)$ and $\delta^-(n)$ as the sets of arcs where the origin node and destination node is node n, respectively.

From Definition 2.2.1, we can define the Minimum Cost Flow Problem, or MCFP (Ahuja et al., 1994). Given a network $(\mathcal{N}, \mathcal{A})$, we can define the variable x_a giving the flow of the commodity to send along arc a. Using this, we define the MCFP as

$$\min \sum_{a \in \mathcal{A}} c_a x_a \tag{2.2.1}$$

$$\text{s.t.} \sum_{a \in \delta^+(n)} x_a - \sum_{a \in \delta^-(n)} x_a = b_n, \ \forall n \in \mathcal{N} \tag{2.2.2}$$

$$0 \leq x_a \leq u_a, \ \forall a \in \mathcal{A} \tag{2.2.3}$$

The premise of the MCFP is to minimise the total cost of flow along the arcs (2.2.1), subject to meeting demand at all nodes (2.2.2), as well as ensuring flows are non-negative, and do not exceed arc capacities (2.2.3).

From the general MCFP many other problems can be formulated as special cases. These include shortest path problems and maximum flow problems. The problem has many applications and extensions. See Ahuja et al. (1994) for a comprehensive overview of networks.

2.2.2 Extensions and variations

Time-expanded networks The MCFP deals with moving commodities around a network, as we need for modelling mail moving through a mail centre in our problem. However, we also need a way to incorporate deadlines in our problem. To do this, we need to consider the minimum cost flow over time problem, as stated by Skutella (2009): Given a network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with associated arc costs, capacities, and transit

times, a source node $s \in \mathcal{N}$ and a sink node $t \in \mathcal{N}$, and a time horizon $T \geq 0$, what is the flow that minimises the cost of sending the required flow from s to t within the time horizon T?

A popular method of dealing with this problem is to use a *time-expanded network* (also called a *dynamic* or *space-time* network). First defined by Ford and Fulkerson (1958), there are a number of similar definitions with subtle differences (Skutella, 2009; Fischer and Helmberg, 2014). Here we give the definition of Skutella (2009):

Definition 2.2.2. *Suppose we have:*

- *A given finite time horizon, represented as $T+1$ discrete time periods $t = 0, 1, ..., T$.*
- *A network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with:*
  - *Arc capacities u (for now, we will assume the lower bound on each arc is 0)*
  - *Arc costs c*
  - *Arc transition times θ_{ij}, which denote the number of time periods taken to move between nodes i and $j \in \mathcal{N}$, with $\theta_{ij} \in \mathbb{N}$, $\forall i, j \in \mathcal{N}$*

Then we define the time-expanded network as $\mathcal{G}^T = (\mathcal{N}^T, \mathcal{A}^T)$ with

$$\begin{aligned}\mathcal{N}^T :=& \{(i,t) : i \in \mathcal{N}, t = 0, 1, ..., T-1\} \\ \mathcal{A}^T :=& \{((i,t),(j,t+\theta_{ij})) : (i,j) \in \mathcal{A}, t = 0, 1, ..., T-1-\theta_{ij}\} \\ & \cup \{((i,t),(i,t+1)) : i \in \mathcal{N}, t = 0, 1, ..., T-2\}\end{aligned}$$

With arc costs $c_{((i,t),(j,t+\theta_{ij}))} = c_{ij}$ and $c_{((i,t),(i,t+1))} = 0$, and arc capacities $u_{((i,t),(j,t+\theta_{ij}))} = u_{ij}$ and $u_{((i,t),(i,t+1))} = \infty$

The idea of the time-expanded network is to take a base network $\mathcal{G}$ and make T copies of it, representing each of the time periods in $0, 1, ..., T-1$. The nodes in these

copies of $\mathcal{G}$ make up the nodes of the new time-expanded network $\mathcal{G}^T$. Each node in $\mathcal{N}^T$ now represents both a location from the base network $w \in \mathcal{N}$ and a time period $t \in 0, 1, .., T$ (the sink node is often associated with time T). The arcs of $\mathcal{G}^T$ represent flows that are possible in the network, both in terms of location in the base network, and time period. For example, if there exists an arc $(w_1, w_2) \in \mathcal{A}, w_1, w_2 \in \mathcal{N}$, and the travel time $\theta_{w_1 w_2} = \hat{t}$, then the arc $((w_1, 0), (w_2, \hat{t}))$ would exist in $\mathcal{A}^T$, as would $((w_1, 1), (w_2, \hat{t}+1))$, $((w_1, 2), (w_2, \hat{t}+2))$, and so on. Therefore, movement of flow across an arc $a = (w_1, t_1) \rightarrow (w_2, t_2)$ now represents flow moving from w_1 to w_2 over the time period $[t_1, t_2]$.

The advantage of a time-expanded network is that once it has been set up, it can be used with the MCFP just like a regular network. Therefore, any solution algorithms that work with the MCFP, and any extensions to it, are also applicable to MCFP with time-expanded networks. The drawback to these networks is that since the number of nodes is proportional to both $|\mathcal{N}|$ and T, the sizes of the MCFP using a time-expanded network grows rapidly. This can quickly become intractable, with Klinz and Woeginger (2004) showing that the MCF over time problem is weakly NP-hard.

By using a time-expanded network, we have incorporated time and deadlines into the interpretations of the flows in the network. By altering the construction of the network, we can enforce deadlines into the sorting. For example, if flow needs to be reach a certain node by time T, we simply connect the sink node to the node representing this destination node at time T. Therefore, for flow to reach the sink node (i.e. to have a feasible solution) it must pass through the required node at time T.

Multi-commodity networks The MCFP is designed to optimise a network with a single commodity, where all units of this commodity may be treated the same. However, in the mail centre, there are multiple streams which need to be sorted in the same WAs at the same time. This could mean that the streams need to follow different paths through the network, or have different throughputs, or different sorting deadlines.

Because of this, we need to use a *multi-commodity network*. This network is very similar to the networks 2.2.1 and 2.2.2. However, we now consider a set of commodities $\mathcal{K}$, each having a specified demand at each node b_n^k. With these additions, we can amend Problem (2.2.1) to be the multi-commodity minimum cost flow problem (MCMCFP):

$$\begin{aligned}
\min \quad & \sum_{a \in \mathcal{A}, k \in \mathcal{K}} c_a x_a^k \\
\text{s.t.} \quad & \sum_{a \in \delta^+(n)} x_a^k - \sum_{a \in \delta^-(n)} x_a^k = b_n^k, \ \forall n \in \mathcal{N}, \ \forall k \in \mathcal{K} \\
& 0 \leq x_a^k, \ \forall a \in \mathcal{A}, \ \forall k \in \mathcal{K} \\
& \sum_{k \in \mathcal{K}} x_a^k \leq u_a, \ \forall a \in \mathcal{A}
\end{aligned} \tag{2.2.4}$$

We see the MCMCFP is very similar to the MCFP. The key differences are an extension of constraints (2.2.2) to hold for all commodities, and the addition of the bundle constraint (2.2.4). The bundle constraint is a total capacity on the sum of the flow for all commodities on each arc. The addition of the bundle constraint is significant, as it prevents the problem being dis-aggregated into $\mathcal{K}$ smaller MCFPs.

Network design problems The other important extension to the MCFP is the *minimum cost network design problem* (which were will refer hereafter to as the network design problem). In the network design problem, the user has control to add/increase capacity to arcs in the network, as well as deciding where the flow goes. Practically, this is achieved by adding variables $y_a, a \in \mathcal{A}$, which can increase the capacity of arc a in the network. These variables can be defined as binary, indicating whether or not the arc is present in the network, or continuous/integer, which then can be used to change the capacity of the arc. Each arc will often have an associated cost m_a. The formulation of the network design problem is very similar to that of the MCFP. The only differences is that Constraint (2.2.3) is changed to link the presence of arcs to the

flow along the arcs:

$$0 \leq x_a \leq u_a y_a, \ \forall a \in \mathcal{A}$$

and that the cost of including arcs is added to the objective function:

$$\sum_{a \in \mathcal{A}} (c_a x_a + m_a y_a) \tag{2.2.5}$$

Note that the network design problem can build upon the MCMCFP to include multiple commodities.

2.2.3 Solution techniques

Initial algorithms for the MCFP include the cycle-cancelling algorithm (Klein, 1967), successive shortest-path algorithm (Busaker and Gowen, 1961) and the network simplex algorithm (Orlin et al., 1993). These algorithms have been extended and applied widely - see one of Kovács (2015), Vieira et al. (2019), or Cruz-Mejía and Letchford (2023) for a full review. However, these algorithms all are designed for single-commodity networks, and hence cannot be used for our setting. Instead, we consider the simplex algorithm for linear programs of Dantzig (1954). Given that the formulation of the MCFP (2.2.1) is linear in all constraints and variables, this algorithm can be applied to this problem. The advantage of the simplex algorithm is its versatility. For example, we see that the MCMCFP is also linear. Hence, the simplex algorithm can be applied to the MCMCFP as well. Furthermore, even when integral variables are introduced in the network design problem (2.2.5), the branch-and-bound algorithm (Land and Doig, 1960) and cutting plane (Gomory, 1958) extensions of the simplex algorithm mean that this problem can be solved as well.

We do note the relative disadvantage in speed of the simplex algorithm compared to the other MCFP algorithms (Vieira et al., 2019). However, work has been done

to speed up these models. These include the Lagrangian Relaxation (Fisher, 1985), Dantzig-Wolfe Decomposition (Dantzig and Wolfe, 1961), and Benders Decomposition (Benders, 1962) techniques. These techniques have all been shown to improve the solution time of linear/integer programs using the simplex method and associated extensions. We also note their appropriateness for this particular problem. Both Lagrangian Relaxation and Dantzig-Wolfe Decomposition are designed for problems where relaxing one of the constraints makes the problem significantly easier. In the MCMCFP, this is the case when the bundle constraint (2.2.4) is relaxed, as the problem decomposes into k independent MCFPs, which are much easier to solve (either by the simplex algorithm, or one of the earlier-mentioned algorithms). We also note that the difficulty in the network design problem comes from the arc variables y, and that if these were known, the problem would again by easier to solve. This can be dealt with using Benders Decomposition, which iteratively fixes the y variables, solves for the x variables, and uses the solution information to update the y variables. Finally, we also recognise the previous use of these decompositions (particularly Dantzig-Wolfe and Benders Decomposition) in solving MCMCFPs and network design problems in other applications. Dantzig Wolfe has been applied to MCMCFPs in telecommunications traffic (Mamer and McBride, 2000), airline scheduling (Lei et al., 2013), and school timetabling (Dorneles et al., 2017). Benders Decomposition has been employed in the design of supply chain networks (Easwaran and Uster, 2009; Pishvaee et al., 2014), refinery systems (Saharidis et al., 2011), wireless sensor networks (Lin and Uster, 2014), transit systems (Marin and Jaramillo, 2009).

2.2.4 Optimal staffing with network problems

Network models have been used for staff allocation in a number of different settings, including airport ground crews (Andreatta et al., 2014), and bus crews (Paias et al., 2021; Xie and Suhl, 2015). These papers use multicommodity network flow models

to schedule crews, treating crews as the flows, and duties that need to be performed as different nodes. We also see that network design problems have been used to both design and staff mail centres (Bard et al., 1993). In this case, the decision variables include not just staff allocation, but equipment purchasing and floor space allocation. Further extensions to this model include additional decisions to re-configure sorting machines (increasing throughput capacities) and out-sourcing some of the mail sorting (Jarrah et al., 1994a). We note that Jarrah et al. (1994a) also consider a secondary objective of the 'smoothness' of a workplan. That is, they also try to minimise how much the workplan for each WA changes between time periods. They deal with this objective using lexicographic optimisation. In discussion with our industry partner, we were also informed that we should consider this objective in our model as well.

Despite this, there are two important considerations which make the current network design models for mail centres insufficient for our setting. Firstly, network design models for mail centres have a slightly different focus than we need to for our problem. This includes both temporal - current models plan the mail centre for years (Bard et al., 1993; Jarrah et al., 1994b), rather than days - and operational - purchasing machines (Bard et al., 1993) and allocating floor space (Jarrah et al., 1994b), rather than just allocating staff. Current models also do not focus as much on the different objectives as we require, and are missing key details such as tethered WAs and split mail flows.

2.3 Stochastic programming

2.3.1 Two-stage stochastic programs with fixed recourse

The two-stage stochastic program with fixed recourse was first proposed by Dantzig (1955). In this problem, the decision maker has a number of *first-stage* decisions that need to be made before a realisation of a random process reveals itself. The decision maker then needs to make a number of *second stage decisions* or *recourse actions* in

the face of this, which have associated costs. The general form of a two-stage stochastic linear program is given by:

$$\begin{aligned} &\min c^T y + \mathbb{E}_{\mathcal{P}}[Q(y, \boldsymbol{\xi})] \\ &\text{s.t. } Ay \leq b \\ &\quad y \geq 0 \end{aligned} \tag{2.3.1}$$

with *recourse function* Q defined as:

$$\begin{aligned} Q(y, \xi) := \min\ & d_\xi^T x \\ & T_\xi x + W_\xi y \leq h_\xi \\ & x \geq 0 \end{aligned}$$

where

- y are the first-stage decision variables.
- x are the second-stage decision variables.
- A, b, c are deterministic matrices/vectors corresponding to the constraints/costs on the first-stage variables.
- $\boldsymbol{\xi} = (T_\xi, W_\xi, h_\xi, d_\xi)$ is a random vector, which we assume follows some probability distribution $\mathcal{P}$.
- $T_\xi, W_\xi, h_\xi, d_\xi$ are matrices/vectors corresponding to the constraints/costs on the second-stage variables for realised scenario ξ. Note that not all of these elements are necessarily random in every stochastic program.

Here we make an important remark on notation. The convention in the stochastic programming literature is that the first stage variables are represented with x, and

the second stage with y. However, we note that in network design problems, the arcs are denoted with y variables and the flows with x. Unfortunately, we see that when extending this to stochastic network design models, the arc placements behave as the first stage decisions, and the flows as the second stage decisions. This gives a contradiction in conventional notation. For consistency, we will refer to y variables as first stage decisions, and x variables as second stage decisions.

We demonstrate a two-stage stochastic program with a network design problem with uncertain demands. Consider the Network Design Problem (2.2.5). Previously, we assumed all node demands b_n^k were fixed when optimising this problem. Now, we assume that these demands are stochastic, and we need to select our arcs (the y variables) before the demands (the b_n parameters) are known. We assume that the demands b_n^k are random variables. Rather than enforcing that all demand is met, there is a cost associated with unmet demand, w. Therefore, this program is written as:

$$\begin{aligned} &\min_{y} \sum_{a \in \mathcal{A}} m_a y_a + \mathbb{E}_{\mathcal{P}}[Q(y, b)] \\ &\text{s.t. } y_a \in \{0, 1\}, \forall a \in \mathcal{A} \end{aligned}$$

where $Q(y, b)$ is the recourse function, given by the optimal value to the linear program:

$$\begin{aligned} &\min_{x,z} \sum_{k \in \mathcal{K}, n \in \mathcal{N}} w_k z_n^k \\ &\text{s.t. } \sum_{a \in \mathcal{O}_n} x_a^k - \sum_{a \in \mathcal{D}_n} x_a^k = b_n^k + z_n^k, \ \forall n \in \mathcal{N}, \ \forall k \in \mathcal{K} \\ &\quad 0 \leq x_a^k, \ \forall a \in \mathcal{A}, \ \forall k \in \mathcal{K} \\ &\quad \sum_{k \in \mathcal{K}} x_a^k \leq y_a u_a, \ \forall a \in \mathcal{A} \end{aligned} \tag{2.3.2}$$

where the new decision variable z_n^k is the *recourse action* for unmet demand for com-

modity k at node n. In a practical application, this could represent a number of different actions, such as accepting that the demand is unmet (usually with an accompanying penalty) or taking (expensive) stop-gap measures to meet the demand in the event of this realisation.

2.3.2 Performance measures for stochastic programs

Given we will be applying a stochastic program to the mail centre problem (and comparing it to previous deterministic methods) we want to be able to show the difference in performance of the two methods. In this section, we outline some ways to show the value of using a stochastic program over a deterministic solution. We define these measures adapting the terminology of Birge and Louveaux (2011).

To define these ideas, we first need to define some other concepts. In order to define these concepts, we define

$$\begin{aligned} z(y,\xi) &= c^T y + Q(y,\xi) \\ \text{s.t. } & Ay \leq b, y \geq 0 \end{aligned} \tag{2.3.3}$$

That is, $z(y,\xi)$ is the total cost of the Stochastic Program (2.3.1) for a given first stage decision y and realisation ξ. We define $\bar{y}(\xi)$ as the solution to (2.3.3). The first baseline solution is the *Wait and See* or *WS* solution. This is defined by Madansky (1960) as

$$\begin{aligned} WS =& \mathbb{E}_\xi \left[\min_y z(y,\xi) \right] \\ =& \mathbb{E}_\xi z(\bar{y}(\xi),\xi) \end{aligned}$$

the WS solution is the expectation of z when we can observe the variable ξ before planning for a decision. It is the best case (and in many cases, unrealistic) scenario, where all information is known before making a decision. While not realistic, it can be

used as a baseline measure.

By contrast, we define the *Recourse Problem* (RP) as

$$RP = \min_{y} \mathbb{E}_{\xi} z(y, \xi)$$

with optimal solution x^*. The recourse problem is just the stochastic program from (2.3.1).

With these, we can now define the *Expected Value of Perfect Information* ($EVPI$) as

$$EVPI = RP - WS$$

The $EVPI$ gives the difference between the objective of the recourse problem (where decisions are made before all information is known), and the expected value of the wait and see solution (where all information is observed before decisions are made). Therefore, the $EVPI$ tells how much value is lost by having to make decisions before knowing all information. It can also be interpreted as how much it would be worth to receive perfect forecasts of the uncertainty when making the decisions.

The $EVPI$ gives the value of perfect forecasts to the uncertainty, and is an important benchmark to compare to. However, given perfect forecasts are unavailable in real-world applications, a different measure can be used to evaluate a stochastic program, known as the *Value of the Stochastic Solution* or *VSS*.

Firstly, we define the *Expected Value* or EV problem. This problem is defined as

$$EV = \min_{y} z(y, \bar{\xi}) \tag{2.3.4}$$

where $\bar{\xi}$ is the expected value of the uncertain parameters in the stochastic program. This problem is the same as assuming the uncertain parameters take their expected

values, then solving a deterministic problem based on these values. We define $\bar{y}(\bar{\xi})$ as the solution to Problem (2.3.4). Using $\bar{y}(\bar{\xi})$ we can define the *Expected Value of the EV solution* or the EEV solution as

$$EEV = \mathbb{E}_{\xi}\left[z\left(\bar{y}(\bar{\xi}), \xi\right)\right]$$

that is, we take the expectation of using $\bar{y}(\bar{\xi})$ over all possible realisations of ξ. This gives what the expected long-term cost would be of using the EV solution.

Now we have defined the EEV, we can use it to define the *Value of the Stochastic Solution* or *VSS*. This is defined as

$$VSS = EEV - RP$$

with RP defined as above. The VSS is therefore the reduction in cost that we achieve by using a stochastic program rather than just assuming the expected values of the uncertain parameters. This is a useful measure, as it allows a comparison between deterministic and stochastic models.

There are a few basic properties of these solutions. Proofs for these properties are provided in Chapter 4 of Birge and Louveaux (2011).

Firstly, we have:

$$WS \leq RP \leq EEV$$

The WS solution will be superior, as we are assuming we have perfect information when making our decisions. The RP solution will be preferred to the EEV solution is it uses more information when making the decisions.

Secondly, we have the following

$$0 \leq EVPI \leq EEV - EV$$

$$0 \leq VVS \leq EEV - EV$$

The lower bounds hold for all stochastic programs. The upper bounds hold only when c, T, and W are fixed over all ξ. This shows that when $EEV = EV$, both the $EVPI$ and $VVS = 0$. This occurs when the optimal solution to $z(y, \xi)$ is independent of ξ. In these cases, the uncertain variables do not affect the optimal decision to the problem, hence there is no value from knowing this information, or incorporating the uncertainty in the decision making process. If this is not the case, then there is potential to either account for or use this information to improve solutions.

2.3.3 Deterministic equivalent and solution techniques

When $\mathcal{P}$ is continuous, the expectation in Problem 2.3.1 is often intractable, making the problem impossible to solve. This is often dealt with by approximating $\mathcal{P}$ with a discrete distribution $\mathcal{P}^*$. That is, $\mathcal{P}^*$ consists of discrete scenario set $\mathcal{S} = [\xi_1, ..., \xi_n]$ with associated probabilities $p_1, \ldots, p_n$. We discuss how this scenario set is chosen in Section 2.4.

When approximating $\mathcal{P}$ with $\mathcal{P}^*$ the expectation in (2.3.1) becomes

$$\sum_{s \in \mathcal{S}} p_s Q(y, \xi_s)$$

This allows us to alter Problem (2.3.1) to define the scenario-based stochastic program:

$$\min c^T y + \sum_{s \in \mathcal{S}} p_s d_{\xi_s}^T x_s \tag{2.3.5}$$

$$\text{s.t. } Ay \leq b$$

$$T_{\xi_s} x_s + W_{\xi_s} y \leq h_{\xi_s}, \ \forall s \in \mathcal{S} \tag{2.3.6}$$

$$x_s \geq 0, \ \forall x \in \mathcal{S}$$

$$y \geq 0$$

We see that Problem (2.3.5) now becomes a linear/integer program, depending on the nature of the x and y variables (in (2.3.5), they are defined as linear). Given this, the solution techniques previously described for linear/integer programs can again be applied here.

We again note the issues with tractability that can occur using the simplex method (for linear programs) and branch-and-bound algorithm (for integer programs). This is especially prevalent with stochastic programs as the size of the problem scales poorly with the number of scenarios, as an entire set of second stage decision variables (x_s) and second stage constraints (Constraint (2.3.6)) are needed for every scenario used. Fortunately, the decomposition techniques discussed in Section 2.2.3 can also be applied here. Benders Decomposition especially is very commonly used for stochastic programs, given their structure. That is, if the first-stage decisions are fixed, then the stochastic program decomposes into $|\mathcal{S}|$ smaller sub-problems (one for each scenario), which is ideal for Benders Decomposition. First used for stochastic programs by Van Slyke and Wets (1969), it is known in stochastic programming literature as the "L-shaped method". Multiple adjustments have been made to the method since its first use. Rahmaniani et al. (2017) provide a review of state-of-the-art techniques for it.

2.3.4 Accounting for randomness in mail centres

We see very little work accounting for randomness in daily mail volumes when staffing. Jarrah et al. (1994a) do note the fluctuations in the mail volumes. However, they plan for them by simply taking the expected value of the mail volumes, increasing the volume by 25%, and taking this to be the deterministic demand. While Zhang et al. (2009) consider disruptions to the workplans, they optimise making changes to a given workplan, given certain disruptions. That is, the model reacts to disruptions, rather than 'anticipates' or plans for them. Because of this, we will need to use more advanced techniques (such as stochastic programming) to plan for the uncertainty in the mail volumes.

2.4 Scenario set selection

Given the importance of scenario-based stochastic programs, a natural question is how to chose an appropriate scenario set.

Picking this set is a balance of two important, but contradictory, aspects. Firstly, we wish to pick a scenario set that accurately represents the uncertainty. While there are many ways to consider this, the most common method is to increase the size of the scenario set. This improves the representation of the uncertainty. However, keep in mind that for each scenario, Problem (2.3.2) will need an entire set of second stage variables and associated constraints. Therefore, as the scenario set size is increased, the size of the problem to solve vastly increases. This trade-off is shown in Figure 2.4.1.

Therefore, the challenge in scenario set selection is to try and balance these two aspects. That is, trying to pick a scenario set that is small enough such that Problem (2.3.2) is tractable, but also representative enough that the scenario-based problem yields a good approximation of the recourse function.

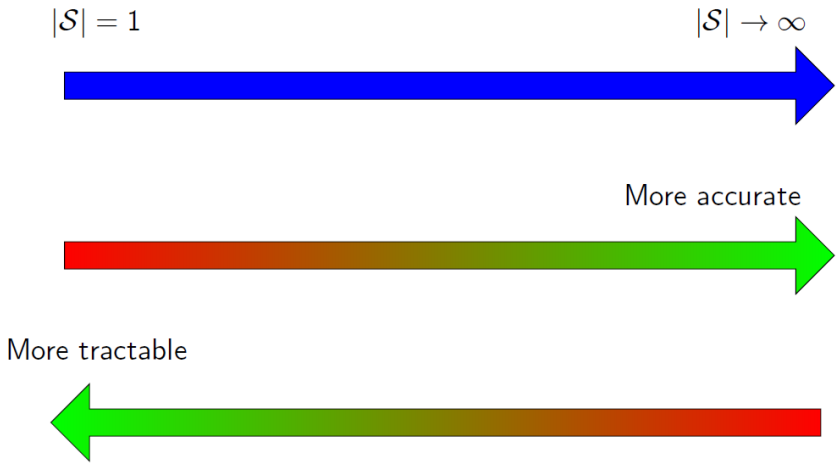


Figure 2.4.1: The contradiction in scenario set selection.

2.4.1 Performances measures for scenario sets

We need some measures to assess the quality of the sets themselves. We outline a number of them here. In this section, we are assuming that we are solving the two-stage stochastic program given by (2.3.1). We denote the objective of this problem evaluated for a given solution y under 'true' distribution $\mathcal{P}$ as $F_{\mathcal{P}}(y)$ for this section.

If $\mathcal{P}$ is continuous (or discrete, but large), $F_{\mathcal{P}}(y)$ cannot be evaluated directly. We need to determine an appropriate *scenario set* $\mathcal{S}$ to use as an approximation. Note that in some literature, this is referred to as a 'scenario tree'. The use of scenario 'tree' rather than 'set' comes from the multistage stochastic program setting (King and Wallace, 2012) We use 'set' as we are only dealing with two-stage problems. We assume that $\mathcal{S}$ has finite support $[\xi_1, ..., \xi_n]$, with associated probabilities $p_1, ..., p_n$.

Various concepts we outline require the optimal values or optimal solutions to $F_{\mathcal{P}}$. We define $F^*_{\mathcal{P}}$ and $y^*_{\mathcal{P}}$ as the optimal value and an optimal solution (respectively) to $F_{\mathcal{P}}$ under distribution $\mathcal{P}$. Note that when we solve the stochastic program over a scenario set $\mathcal{S}$, we will slightly abuse our notation and define the optimal value and (an) optimal solution as $F^*_{\mathcal{S}}$ and $y^*_{\mathcal{S}}$.

The most natural way to assess the quality of a scenario set is using the *approxima-*

tion error (Pflug, 2001) between the scenario set and the 'true' distribution.

Definition 2.4.1. *Suppose we want to approximate Problem $F_{\mathcal{P}}$ using scenario set $\mathcal{S}$ The approximation error from using $\mathcal{S}$ as an approximation to $\mathcal{P}$ is then defined as*

$$e(\mathcal{P}, \mathcal{S}) = F_{\mathcal{P}}(y^*_{\mathcal{S}}) - F^*_{\mathcal{P}}$$

While this is the measure that we are trying to minimise, we cannot use it in practise - if we could solve the stochastic program over $\mathcal{P}$ to obtain $F^*_{\mathcal{P}}$, then we would not need to use a scenario set.

Instead, we can often find the out of sample (OOS) value for a given scenario set $\mathcal{S}$.

Definition 2.4.2. *The OOS value for a scenario set $\mathcal{S}$ under true distribution $\mathcal{P}$ is given by:*

$$OOS_{\mathcal{S}} = F_{\mathcal{P}}(y^*_{\mathcal{S}})$$

That is, it is the first stage cost of $y^*_{\mathcal{S}}$ and the average recourse cost of the true distribution under solution $y^*_{\mathcal{S}}$. This is easy to do if $\mathcal{P}$ is discrete. If $\mathcal{P}$ is continuous, we instead may approximate it using a large sample.

The OOS values do not given any information about how close to optimal a given solution is. However, it does allow comparison between two scenario sets. In particular, if we have two scenario sets $\mathcal{S}_1$ and $\mathcal{S}_2$, we can compare these two sets by calculating $y^*_{\mathcal{S}_1}$ and $y^*_{\mathcal{S}_2}$ and comparing $OOS_{y^*_{\mathcal{S}_1}}$ and $OOS_{y^*_{\mathcal{S}_2}}$.

2.4.2 Scenario set selection methods

Taxonomy of Scenario set selection techniques

We now discuss different methods of selecting scenario sets for stochastic programs. Broadly speaking, there are two main ways that scenario set selection procedures can

be classified:

1. What information is used in the methods. That is, if the method is *distribution-based* or *problem-based*.

2. Whether the selected scenarios are newly generated, or (if the true distribution is discrete) are a subset of the existing true distribution. That is, if the method is a *scenario generation* method or a *scenario reduction* method.

The first classification refers to whether the information used to select scenarios is drawn from information about the true distribution only (distribution-based), or whether information specific to the problem we are solving is also incorporated (problem-based). The second classifies scenarios based on where the scenarios in the new set come from. That is, is the new set populated by new scenarios which are user-generated (including sampling scenarios from a continuous distribution), or from an existing scenario set which has been reduced in size.

At this point, we make an important note on terminology. In the literature, 'scenario generation' is often used to refer to any scenario set selection procedure. However, we use it to specifically refer to methods in which new scenarios are 'generated', as opposed to methods which start with a large scenario set and select a subset. Throughout the thesis, we will use the term 'scenario set selection' to refer to general methods of choosing a scenario set, and 'scenario generation' to specifically refer to methods which generate new scenarios.

Distribution-based scenario set selection

Traditionally, most methods have been distribution-based. This means that the scenario sets are selected such that the set best represents a desired distribution. This could be the true distribution, or a distribution with user-desired properties.

The most straight-forward method of distribution-based (and indeed all) scenario set selection methods is Monte Carlo sampling. In this method, the scenarios are

randomly drawn from some probability distribution. This can be from a continuous distribution (in which case the procedure is a scenario generation procedure), or can select members of a larger discrete distribution (which makes this procedure a scenario reduction method).

This is computationally very easy, and has been used by authors previously (April et al., 2003; Jobst and Zenios, 2003). When the scenario set is chosen with this, the solution to the stochastic program is known as the sample average approximation, or SAA. In order to improve upon basic Monte Carlo sampling, simple variance reduction techniques have also been used for scenario set selection. These include Latin hypercube or antithetic sampling (Higle, 1998). However, these techniques rely on the scenarios having independent marginal distributions.

More sophisticated methods try to pick representative sets more formally. These approaches (known as *optimal discretisation* approaches) try to pick a set which minimises some distance metric between some true distribution and the selected scenario set. A common metric used is the *Wasserstein distance* which we define here for the case where the true distribution is discrete, using the definition of Dupacova et al. (2003).

Definition 2.4.3. *The Wasserstein distance $\mathcal{D}_W(\mathcal{P}, \mathcal{S})$ between two discrete distributions $\mathcal{P} = [\zeta_1, ..., \zeta_m]$ and $\mathcal{S} = [\xi_1, ..., \xi_n]$ of size m and n with probabilities $p_1, ..., p_m$ and $q_1, ..., q_n$ respectively is defined as the square root of the objective value to the problem:*

$$\begin{aligned} \min_{\pi \in \mathbb{R}_+^{m \times n}} & \sum_{i=1}^{m} \sum_{j=1}^{n} \pi_{ij} \|\zeta_i - \xi_j\|^2 \\ s.t. & \sum_{j=1}^{n} \pi_{ij} = p_i, \forall i \in [1, ..., m] \\ & \sum_{i=1}^{m} \pi_{ij} = q_i, \forall j \in [1, ..., n] \end{aligned}$$

The Wasserstein distance can be interpreted as the solution to the optimal transportation problem from $\mathcal{P}$ to $\mathcal{S}$. The π_{ij} variables can be interpreted as the amount of probability mass being transported from ζ_i to ξ_j at cost $\|\zeta_i - \xi_j\|^2$. A more general definition of the Wasserstein distance is provided by Villani (2003).

The Wasserstein distance was first used as a scenario generation technique by Pflug (2001). This distance can also be generalised to the Fortet-Mourier metric (which has fewer assumptions on the objective function of the stochastic program) and used as well. This was done by Dupacova et al. (2003), who also developed scenario reduction algorithms to solve this for discrete true distributions. Further theoretical developments have been made to these methods (Heitsch and Römisch, 2003, 2007), as well as extensions of these methods to multi-stage scenario sets (Growe-Kuska et al., 2003; Heitsch and Römisch, 2009), and changes to increase computational efficiency (Kammammettu and Li, 2023). However, while there is a lot of theoretical grounding for these methods, they often involve solving highly non-convex problems, which can be difficult.

Given these difficulties, the optimal discretisation problem is often solved used clustering techniques. In clustering, the scenario set is grouped into $|\mathcal{S}|$ clusters. The selected scenario set is the set of the cluster centroids. Previously, the optimal discretisation method of Pflug (2001) was solved with a k-means type algorithm, and k-means has also been applied to multi-stage scenario set selection methods Sutien et al. (2010). However, other clustering techniques, such as k-medians (Kaufman, 1990) and spectral clustering (Von Luxburg, 2007) could also be used for scenario set selection. Whether clustering is regarded as a scenario generation or reduction method is specific to the type of clustering used. For example, if k-means clustering is used, the cluster centroids are new observations equal to the means of the clusters. Hence, these are scenario generation procedures. On the other hand, in k-medoids clustering the cluster centroids are the medoid observations in the cluster. Hence, this would be a scenario reduction method if used in scenario set selection.

Other methods try to construct a scenario set to have specific user-desired properties. In these methods, an initial set of randomly generated scenarios is then transformed to match pre-set target moments. With these methods, the user can set up to the first four moments (Høyland et al., 2003; Ponomareva et al., 2015), as well as matching given correlations between the scenarios as well (Høyland et al., 2003). Similar methods have also been used to control marginal distributions (Kaut and Lium, 2014), copulas (Kaut and Wallace, 2011) and empirical CDFs (Calfa et al., 2014).

Other authors have matched the moments using regression-type procedures (Høyland and Wallace, 2001). In these procedures, a non-linear optimisation problem is created and solved to generate the scenario set. For example, if we need the scenario set ξ to have the mean μ, we solve a problem with the constraint

$$\sum_{s \in \mathcal{S}} p_s \xi_s^i = \mu_i$$

where i is the index of the variable in the scenario. Further constraints can be added if specific variances and higher moments are needed. These constraints can be treated as soft constraints, and put into the objective function of the program to be minimised. Alternatively, they can be viewed as hard constraints, in which case, only a feasible solution needs to be found to give the scenario set.

Problem-based scenario set selection

More recent methods try to use information specific to the problem when selecting scenarios. These are known as *problem-based* methods. Many of these methods are based on or are analogous of existing distribution-based methods. These include sampling (Dantzig and Glynn, 1990; Infanger, 1992), clustering (Feng and Ryan, 2016; Narum, 2020; Hewitt et al., 2022), and (near) means-matching (Zhang et al., 2023).

Early work by Dantzig and Glynn (1990) and Infanger (1992) used importance sampling to select scenarios. In this work, scenarios from lower-probability regions of

the distribution were weighted more heavily in their sampling process. The process has quite restrictive requirements of the objective function to be minimised, but has still been used by more recent authors (Papavasiliou and Oren, 2013).

Other authors frame the problem as classifying scenarios by how much they will affect the stochastic program if included. This can be done in different ways for different problems, having been done for risk-adverse stochastic programs (Arpón et al., 2018), problems optimising tail-risk measures (Fairbrother et al., 2022), and two-stage programs (Prochazka and Wallace, 2018). The drawback of these methods is that they are often tailored to the specific problem to be solved, and are hence difficult to generalise.

A different approach is to create a number of candidate solutions, and test these solutions on different scenarios in the full set. This gives information that can again be used in scenario set selection methods. The specifics about how to create the pool of candidate solutions vary. These include heuristics (Prochazka and Wallace, 2020), solving single-scenario deterministic problems (Feng and Ryan, 2016; Sun et al., 2018; Bertsimas and Mundru, 2023; Hewitt et al., 2022; Keutchayan et al., 2023; Zhang et al., 2023), and solving smaller stochastic programs (Narum, 2020; Narum et al., 2024). These approaches seem promising, with different scenario reduction methods applied to different candidate solution pools. However, little work has been done to unify and generalise these approaches. We give a framework to draw these together in Chapter 5.

2.4.3 Stability

Given the different methods available to select scenario sets, we now discuss some ways of evaluating different methods. Furthermore, most methods have an element of inherent randomness. That is, the scenario set selected by the method will not be the same every time. Because of this, we wish to see that methods are consistent (produce sets with similar optimal values) and that the sets that they produce are of high quality (lower optimal values).

For this, we can examine the *Out-Of-Sample (OOS) stability* (Kaut and Wallace, 2007). When using OOS stability, we are evaluating if the overall cost with respect to the true distribution is the same for the optimal solution of different sets $\mathcal{S}_i, i \in 1, ..., N$ generated by a given scenario set selection method. That is, we want to see

$$OOS_{\mathcal{S}_i} \approx OOS_{\mathcal{S}_j}, \forall i, j, i \neq j \in 1, ..., N \tag{2.4.1}$$

Property (2.4.1) basically states that no matter the scenario set used to approximate the true distribution (generated by the method), the solution to the stochastic program under the set will give roughly the same objective value.

The OOS stability does not tell us much in-and-of itself. However, it is useful for comparing different scenario set generating procedures. What we ideally wish to see for the OOS stability is:

1. A narrower range of the OOS values. Since we are trying to approximate the expected recourse cost, it is ideal that the estimated recourse values are not too spread.

2. A lower average of the OOS values. Since we are trying to minimise a problem, we wish to see that the narrower range of recourse values means that better first stage solutions can be chosen.

To properly assess the OOS stability of a model, we need to check that Property (2.4.1) holds for multiple different solutions from multiple different sets. One way to assess this is through the following procedure:

For an arbitrary number of runs N:

1. Generate scenario sets $\mathcal{S}_i, i = 1, ..., N$

2. For each scenario set $\mathcal{S}_i$:

   (a) Evaluate $F_{\mathcal{S}_i}(y)$ to obtain $y^*_{\mathcal{S}_i}$

(b) Calculate $OOS_{\mathcal{S}_i}$.

This gives us N OOS values we can examine, based on the criteria above.

Sometimes the OOS stability cannot be calculated. This can occur if $\mathcal{P}$ is too large to calculate $F_{\mathcal{P}}(y)$, or $\mathcal{P}$ is continuous (and a large approximate distribution is not available). While we cannot test overall quality of the solutions, we can measure the *in-sample stability* (Kaut and Wallace, 2007). This is a measure of how effective a scenario set selection procedure is at creating sets that will give a similar solution every time. Suppose that (via some method) we have produced a set of N unique scenario sets $\mathcal{S}' = [\mathcal{S}_1, ..., \mathcal{S}_N]$. We say that our model has in-sample stability if

$$F^*_{\mathcal{S}_i} \approx F^*_{\mathcal{S}_j}, \forall \mathcal{S}_i, \mathcal{S}_j \in \mathcal{S}' \tag{2.4.2}$$

Note that King and Wallace (2012) assert that we only care that the objective values of the two SPs in (2.4.2) are similar, not the corresponding optimal solutions $y^*_{\mathcal{S}_i}$ and $y^*_{\mathcal{S}_j}$. They argue that since we are trying to solve a problem, we can regard two scenario sets as being similar if the objective values from the stochastic program are similar.

2.5 Summary

Tying back to staff rostering in sequential sorting facilities, we established at the start of this chapter that a promising area to explore was network design models. However, in keeping with both our problem statement, and the gaps in the literature, we would need to consider a new method to account for the randomness in the mail volumes.

We reviewed different extensions and techniques for solving network flow problems. Crucially, we saw that there are extensions to the base MCFP available to account for time deadlines, multiple commodities, and network design - all key features in our mail centre problem. Within the literature, extensions have been made to improve solution

algorithms for the models. These new advancements will be useful for our new work on mail centre staffing. We show our work in this area in Chapter 3.

In terms of stochastic models, we saw that stochastic programming would be well suited for this type of problem. This allows us to further the field in mail centre staffing, by properly accounting for randomness in mail volumes for the first time (to the best of our knowledge). This work is presented in Chapter 4.

Looking deeper into the stochastic programming field, we examined the area of scenario set selection. We noted that there appeared to be a gap in the area of problem-based scenario reduction, in that there was no general framework to make existing methods problem-based. We develop a framework to generalise existing problem-based scenario reduction methods in Chapter 5.

Chapter 3

A time-expanded network design model for staff allocation in mail centres

3.1 Introduction

Running any large facility requires rostering staff and resources (Van den Bergh et al., 2013). We consider facilities which process large quantities of different *streams* of materials, which have to pass through a series of different *work areas* (WAs) to be processed. Operational workforce planning is difficult because changes in staff levels and processing rates at upstream WAs will have large effects on downstream WAs. This may be further complicated by additional factors such as some WAs not being able to operate simultaneously, flows of materials being split between several areas, arrivals of streams at the facility at different times, and deadlines on processing.

The shift manager often has multiple objectives to balance when setting staff levels. Firstly, staff and resources are costly, so the manager wants to minimise costs of resources used. Secondly, the decision maker may need to consider the impact of shift

patterns on staff. In particular, it is preferable for a worker to not change WAs too often during a shift. Hence, the decision maker could aim to minimize the number of changes in the roster for a WA to achieve a smooth shift pattern.

As an example of such a facility, we consider mail centres. These are facilities where mail is processed (i.e. sorted) by both type and destination as part of the delivery process. This application is particularly important as due to increases in electronic communication, there are increased pressures on the mail industry, which requires them to plan their workforce operations as efficiently as possible.

In a mail centre, the streams consist of different combinations of mail type (e.g. letters and parcels of different sizes) and priorities (e.g. first class, second class, tracked/untracked, etc). There are many tasks required to sort mail, such as:

1. Receiving mail as it arrives at the mail centre.
2. Within letters and parcels, segregating items:
    - Sorting letters by size (e.g., regular-sized envelopes, flats (larger envelopes), and other sizes).
    - Sorting parcels of different dimensions for automatic sorting machines.
    - Separate non-machineable items to be sorted manually (e.g., address not recognizable by machine, damaged items, parcels of irregular shapes (e.g., tubes) or exceeding standard dimensions).
3. Arranging and stacking items to be fed into the automatic sorting machines.
4. Sorting items by destination. This can involve:
    - Sorting into outward (mail that needs to go to another mail centre in the country) or inward (mail that is addressed to a destination nearby, needing to go to a nearby delivery office).

- Sorting inward mail into postcodes/destinations. This can involve multiple stages of sorting, with mail being disaggregated into finer regions at each stage.
- Sequencing mail within postcodes, so they are in order of delivery and ready for post officers to deliver them.

5. Sending mail onto onward destinations.

These tasks are performed sequentially by different WAs. For the majority of mail, these tasks are carried out automatically by machines. Mail that cannot be sorted by machines (e.g. because the address cannot be read) needs to be processed manually at other WAs.

As a universal service, there are strict service levels that a postal system has to meet (Office of Communications, 2012). Meeting these quality of service targets depends heavily on both the number of letters/parcels that need to be sorted and the number of staff rostered to sort them. Additional complications which arise in this context are that there are different mail types with different sorting requirements, and operational requirements given the start and finish times of the different WAs.

The appropriate number and placement of staff need to be determined for each day the centre operates. Existing approaches either do not consider sorting deadlines, they are set at a much longer time scale than required (multiple years, as opposed to a single day), or do not adequately model the splitting of flow between different WAs - a key aspect of our setting.

In this paper, we contribute a new staff scheduling model to address optimal staffing in sequential sorting facilities with high volumes of heterogeneous products. We assess the benefits of the new model by applying it to a mail centre in the United Kingdom. We model this as a design problem on a multi-commodity network (Ahuja et al., 1994), and consider minimising both the maximum number of staff required in the mail centre over a shift, as well as the number of changes in staff levels between the different time

periods. This approach builds upon previous models in the literature, but with a more fine-grained time scale (as opposed to the strategic long-term focus of previous work), emphasis on smoothness of shifts, and new operational constraints, such as split flows and tethered WAs.

The outline of the paper is as follows. Section 3.2 provides a review of the relevant literature. Section 3.3 defines the problem and gives an overview of our specific application of a mail centre. Section 3.4 explains our mathematical model in detail. Section 3.5 presents computational results of our model for a real-world case study. Finally, Section 3.6 discusses these results in context, as well as limitations and further research.

3.2 Literature review

Staff scheduling is important in almost every industry. It has been studied in fields including healthcare (Devesse et al., 2022), transport (Tello et al., 2019), telecommunications (Louly, 2013), education (Kabiru et al., 2017), and security (Restrepo et al., 2012). Extensions of the problem have been made to account for complexities in the available workforce. These include variation in employee skills and effectiveness (De Bruecker et al., 2015), contract type and duration (Bard et al., 2003), job precedence (Zhang and Bard, 2005; Zhang et al., 2009), and shift time flexibility (Ni and Abeledo, 2007; Brunner et al., 2009).

Staff scheduling can cover a number of different aspects (Ernst et al., 2004). These include:

- Demand modelling - determining the number of staff needed to complete all required tasks on a shift. This has been used in airport security screening (Li et al., 2018) and bus crew scheduling (Paias et al., 2021).
- Days off scheduling - determining the number of rest days required for staff between work days, also covered by Paias et al. (2021).

- Shift scheduling - assigning which staff from a pool of employees to tasks that need to be completed, with examples in physician scheduling (Devesse et al., 2022; Tapak et al., 2023).

- Line of work construction - creating repeating cyclic rosters for staff. An example of this is given by Xie and Suhl (2015), to create rosters for bus crews.

- Task or staff assignment - the assignment of tasks to different staff members, or staff to different lines of work. Andreatta et al. (2014) use this in scheduling airport crews.

The large variety of problems also requires different techniques for solving these models. Devesse et al. (2022) use a mixed-integer-programming formulation to solve the physician scheduling problem in emergency rooms. Li et al. (2018) design different networks of security screening for airports, and use simulation to test which networks are the most effective. Tapak et al. (2023) used column generation and constraint programming to devise crew schedules for a railway system. Network models are another technique used to organise scheduling. Andreatta et al. (2014) staff airport ground crews by visualising tasks as a graph in which different nodes represent different aspects of a task (staff member to complete it or equipment to be used), and the flows represent different tasks that use these resources. Xie and Suhl (2015); Paias et al. (2021) look at scheduling bus crews using network flows. In their example, the flows represent workers to complete various duties (nodes).

Within the mail industry, there are a wide variety of approaches to optimal staff scheduling. Jarrah et al. (1994b) and Bard et al. (2003) both calculate how many staff are required for each shift, and determine who works from a pool of available workers. However, this ignores mail sorting deadlines, as well as job precedence constraints.

Other approaches do incorporate sorting deadlines and job precedence. Zhang and Bard (2005), Qi and Bard (2006), and Zhang et al. (2009) consider both staff rostering

and equipment scheduling. In these studies, which machines to schedule and how many staff members to roster on are treated as different decision variables while taking job precedence into account. In particular, Zhang and Bard (2005) and Zhang et al. (2009) examine the scheduling of both equipment and workers, characterising it as a multi-level lot sizing problem. In this problem, they schedule when mail is sent between different WAs, to optimise when equipment is scheduled to run, and when to allocate staff. However, the main focus of these problems is when to set-up and run machines. The question of when to allocate staff is only dealt with as an auxiliary problem, after the equipment schedules have been decided.

Another approach is by Bard et al. (1993) and Jarrah et al. (1994a), who minimise the equipment and staffing costs of setting up a USPS General Mail Facility (GMF). They model the GMF as a network design problem, with mail passing between WAs representing the flows in the model. Furthermore, they use a time-expanded network to include deadline constraints in sorting this mail. However, the authors consider fewer WAs and streams than needed to deal with present-day mail centres. They also omit more specific operational constraints, such as split flows of commodities, and focus on more strategic planning time-horizons (i.e. years, rather than hours or days).

An important aspect is to consider the smoothness of a shift. Here, 'smoothness' refers to how often and how much the number of rostered workers changes during a shift. A smoother shift has fewer and smaller changes in the number of staff working between the time periods. This is desirable for the staff, as they do not have to change WAs unnecessarily during a shift. Furthermore, it has been shown that task-switching can lead to higher error rates in multiple fields (Skaugset et al., 2016). Shift smoothness is common in scheduling/rostering problems (Brucker and Knust, 2012), but generally is not considered when using network design problems to determine staff levels. Specifically to mail sorting, Jarrah et al. (1994a) is the only previous study to look at smoothness, but only as a secondary priority compared to other objectives.

Looking at the literature, there is little research on using networks to solve staffing problems for sequential sorting facilities - particularly considering multiple objectives. Furthermore, our problem can be thought of as a demand modelling problem - determining the number of staff required at various times to account for 'demand' (in this case, materials to be sorted). Network design models also do not appear to have been used for this type of shift scheduling. To bridge this gap, we analyse the effect of different objectives and priorities. Closest to ours is the work by Jarrah et al. (1994a), who use a network design to organise a mail centre, but they do not evaluate the trade-offs between competing objectives and their priorities. Also, they take a long-term strategic view, rather than a day-to-day operational view. Another novelty of our model is that we consider indirect streams and tethering, which are important aspects for the sorting facilities we analyse.

3.3 Problem description

3.3.1 General problem overview

We consider a facility which processes large quantities of different types of items which we call *streams* and may have different properties and requirements. Each stream must pass through several different WAs where items are processed, and subsequently forwarded to the next one until they have been entirely processed.

Mappings Each stream has a known path(s) through the work areas of the facility. The stages of these paths are known as *mappings*. Mappings are pairs of work areas giving where the stream is coming from and where it is going. There are two types of mappings. *Direct mappings* mean that the whole stream passes between the two work areas. *Indirect mappings* mean that only a certain proportion of the stream passes between the two work areas. The remainder goes to other work areas, given by

different indirect mappings.

Work plans Workers are needed to run the work areas. This could be to either operate machines which sort the materials, or to sort the materials by hand themselves. The facility manager needs to decide on a *work plan*, showing how many staff to assign to each WA over the course of a day. Staff levels are linked to the processing as the more staff assigned to a WA, the more items that can be processed in a given time interval.

Our aim is to create a work plan for a whole day. Note that the work day is divided into *shifts*. If a worker is required for part of a shift, then they must be paid for the whole shift. Under the assumption that a worker can be reassigned to any other work area over the course of a shift, the number of workers required for a shift is the maximum number of workers assigned to work areas over the course of the shift.

Priorities The time by which an item is processed is important as companies are often subject to contracts that require them to meet deadlines. In our specific example, these deadlines are enforced through assigned *priorities* at each work area. The priority gives the volume of flow that the work area has to process in a given shift. In this problem, we have 3 types of priority:

1. **All** - that is, all flow entering this work area needs to be processed this shift.

2. **0** - that is, no flow is to be processed, which means that this work area is not open during this shift, and will not be staffed.

3. A given mail volume $r > 0$ - this means that at least r units of flow must be processed and leave the work area during the shift.

Operational constraints The number of staff that can work on a WA during a time interval is subject to operational constraints. WAs can have different *start* or *finish*

times for different shifts. Often these are given as set times. However, some work areas cannot start until another one has finished. These work areas are referred to as being *tethered*, which is a common occurrence in scheduling problems (Brucker and Knust, 2012). Another general constraint could be a maximum number of staff that can be rostered in a work area at a given time - for example, due to a limited amount of floor space.

The main aim when creating a work plan is to minimize the costs of workers while meeting all required deadlines. Also important is the *shift pattern*, that is, how the number of workers at a WA changes over time. A shift pattern with many large and frequent changes will be less practical for the workers compared to a shift pattern with fewer changes.

3.3.2 Mail sorting centre example

In Figure 3.3.1, we show a simplified version of a mail centre as an illustrative example. In reality, mail centres are much larger with many more types of mail than shown here. This is a closed system, where all mail that enters the network passes through WAs and then leaves the network. There are four mail types in the illustrative example, which need to move between eight WAs in order to be sorted. WA 1 would sort letters from parcels, and remove letters/parcels that cannot be mechanically sorted. The mechanical sort letters and parcels would then pass to WAs 2 and 4 respectively. These would be automated sorting machines, and would perform the majority of destination sorting. A small proportion of the mechanically sorted letters/parcels would be unable to be read by a machine and so would be passed to WAs 5 and 6 where they would join the other manually sorting letters/parcels. WA 3 would take the letters/parcels which cannot be processed by a machine, and sort them into letters vs parcels. At WAs 5 and 6, the letters/parcels that need to be manually sorted (including those rejected by the automatic sorting machines) will be manually sorted by destination. WA 7 acts as a

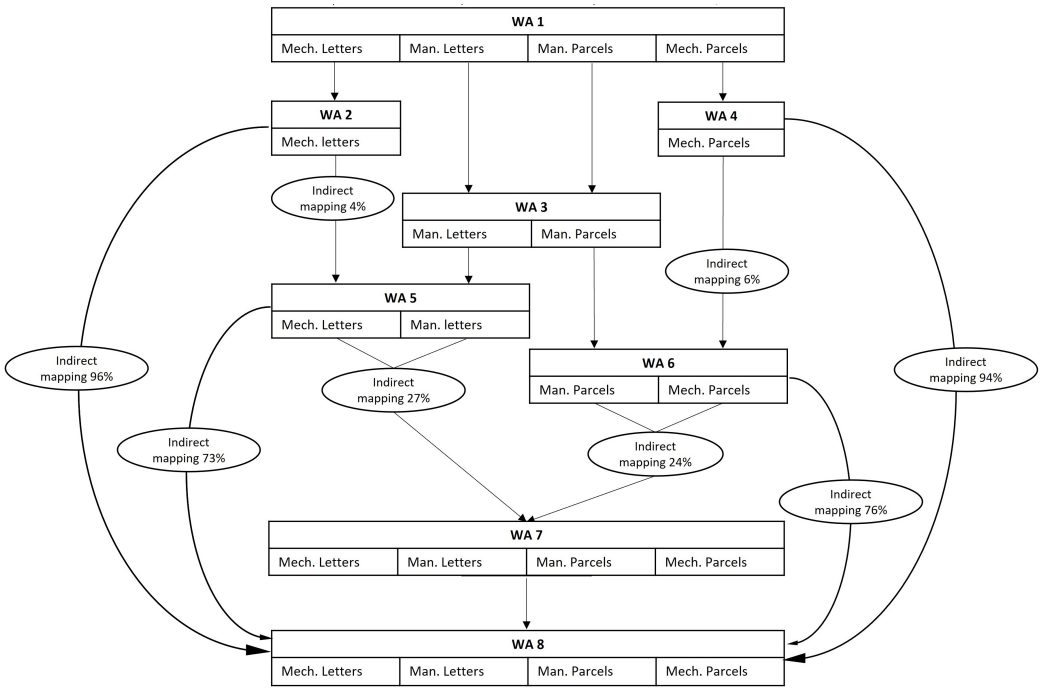


Figure 3.3.1: Diagram of a small example mail centre. (Abbreviations: Mech.: mechanically sorted, Man.: manually sorted)

sequencing WA. Finally, all sorted mail would move to WA 8, which is a consolidation WA, where the mail is prepared to be sent to its next destination.

The arrows show the mappings of the commodities between the different WAs. The majority of mappings are direct mappings, indicating all flow of a commodity passes between the two WAs - such as the mapping for mechanically sorted letters (Mech Letters) from WA 1 to WA 2. There are a number of indirect mappings which are labelled on the relevant commodities. For example, there is an indirect mapping between WA 5 and WA 7, indicating that 27% of all commodities that leave WA 5 need to go to WA 7. The remaining 73% will go straight to WA 8.

The different WAs need staff to operate them. Either to operate the automatic sorting machines (in WAs 2, 4, and 7), sort by hand (WAs 5 and 6) or send onwards (WA 8). The shift manager needs to decide how many workers are needed in each WA at each time.

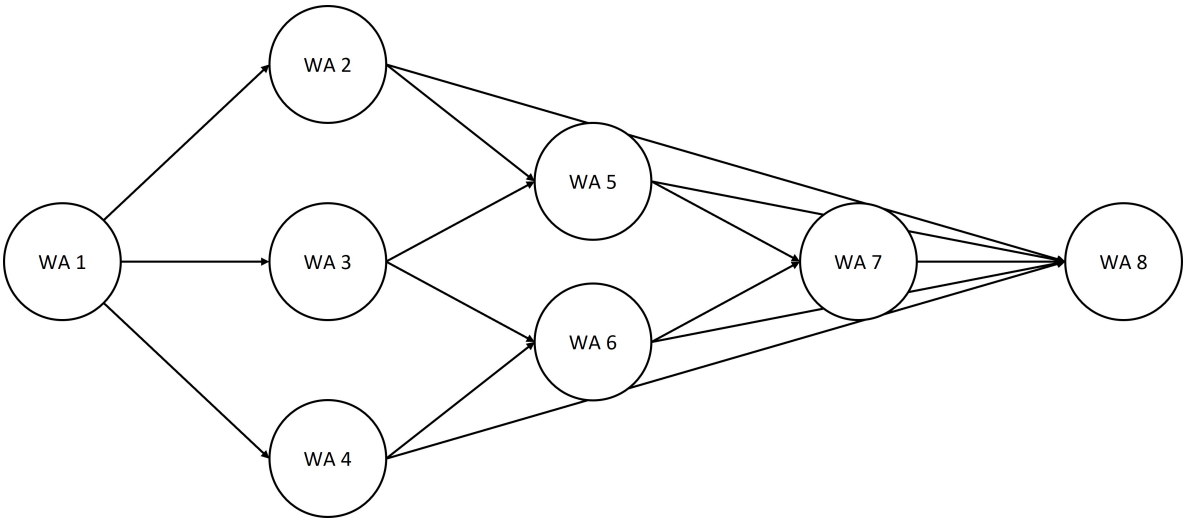


Figure 3.4.1: Base network of a small example of a mail centre

3.4 Mathematical model

3.4.1 Overview of network design problem

We model the sequential processing facility as a network design problem by first constructing a **base network** describing the mappings $\mathcal{E}$ between the different WAs $w \in \mathcal{W}$ in the facility. The base network for the small example of the mail centre shown in Figure 3.3.1 is given in Figure 3.4.1.

The base network does not include any concept of time. However, there are strict deadlines on processing all streams. Therefore, we construct a **time-expanded network** to incorporate time. Let $\mathcal{T} = \{1, \ldots, T\}$ represent a discretisation of time over the course of a day. Each $t \in \mathcal{T}$ corresponds to a short time interval, e.g., 10 minutes. In a time-expanded network, we have a node (w, t) for each WA $w \in \mathcal{W}$ and $t \in \mathcal{T}$, as well as *source* and *sink* nodes to represent points where streams enter and leave the system. Flows in this network will represent the movement of a commodity stream through the WAs in the facility over time.

The arcs in the time-expanded network consist of the following:

- $((w, t), (w, t+1))$ for $w \in \mathcal{W}, t \in [1, T-1]$: Flow travelling along these arc

represents material that is held in WA w from time t to time $t+1$.

- $((w_1, t), (w_2, t+1))$ where (w_1, w_2) is an arc in the base network and $t \in [1, T-1]$: Flow travelling along these arcs represent material that is processed at w_1 in time t and then sent on to w_2.

- $(s, (w, t))$ where s is a source node, and $w \in \mathcal{W},\ t \in \mathcal{T}$: flow along these arcs represent points at which streams enter the system.

- $((w, t), e))$ where $w \in \mathcal{W},\ t \in \mathcal{T}$ and e is a sink node: flow along these arcs represent streams leaving the system.

The time-expanded version of the base network from Figure 3.4.1 is shown in Figure 3.4.2. For this example, we have only one source node linked to WA 1 at time period 1 which means in this case that all streams to be processed arrive there at the beginning of the day. For convenience, we also add a dummy 'completion' WA. This WA signifies flow which has been completely processed and is waiting to be shipped out of the facility. The sink node is then linked to the completion WA in the final time period. Adding the completion WA is useful, as we can easily keep track of which flow has been processed (as it arrives in the sink via this completion area) and which has been delayed (arrives at the sink via different arcs). This helps adhere to WA priorities, mentioned earlier. We therefore add "Completion" as a WA in the network, and add it to the set $\mathcal{W}$. There is also only one sink node which is linked to every WA at the final time period. In doing this, we allow for the case where not all of a stream is processed over the course of a day. This will mean our optimization model below will remain feasible even if it is not possible to process everything in the given time limit. Although we only have one source and sink in this example, it is possible to have multiple. Additional sources would represent additional times and WAs where streams arrive, and additional sink nodes may be used to represent different deadlines for processing.

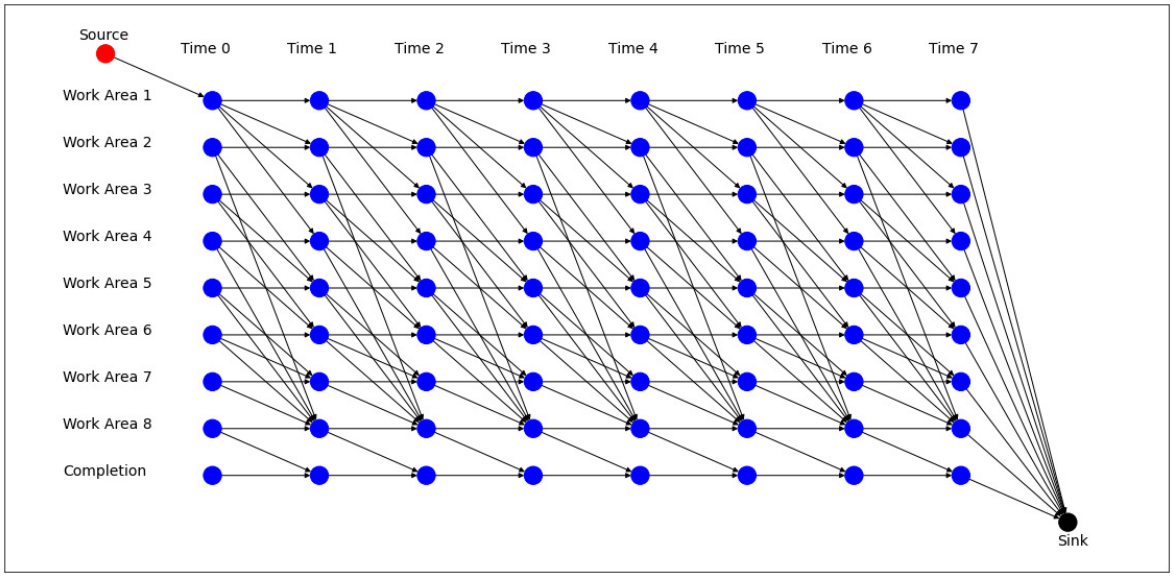


Figure 3.4.2: Time-expanded version of the base network from Figure 3.4.1 for 8 time periods.

3.4.2 Mathematical formulation

The problem of staffing at a sequential sorting facility can be formulated as network design problem on time-expanded network presented above. Flow on this network represents the movement of material as it is processed over the course of a day. The design element comes from the fact that we must decide the number of workers to allocate to each WA over time, determining the flow capacity on arcs between WAs. Note that we present the formulation here minimising the number of workers in the mail centre at any point in time over each shift as the objective. We discuss different choice for different objectives in Section 3.4.3.

We use the following notation to describe our model.

Sets and indices:

- $\mathcal{W}$: set of WAs indexed by w.
- $\mathcal{E}$: set of mappings between WAs.
- $\mathcal{I}$: set of indirect mappings.

- $\mathcal{H}$: set of tethered WAs.
- $\mathcal{J}$: set of shifts, indexed by j.
- $\mathcal{T}$: set of time periods, indexed by t.
- $\mathcal{T}_j$: set of time periods associated with shift j. Note, $\mathcal{T}_j \subset \mathcal{T}$.
- $\mathcal{N}$: set of nodes in the time expanded network, indexed by i.
- $\mathcal{A}$: set of arcs in the time expanded network, indexed by a.
- $\delta^+(i)$ and $\delta^-(i)$: sets of outgoing and incoming arcs for node i, respectively.
- $\mathcal{K}$: set of commodities, indexed by k.
- $w_O(a)$ and $w_D(a)$: origin and destination WAs associated with arc a.
- $t_O(a)$ and $t_D(a)$: origin and destination times associated with arc a.
- $ID^+(w, w', t) := \{a | w_O(a) = w, w_D(a) = w', t_O(a) > t, t_D(a) = t_O(a) + 1\}$: a set of all the arcs originating at WA w and finishing at w', originating at a time later than t. This is used in defining constraints to enforce the indirect streams.
- $ID^-(w) := \{a | w_O(a) \neq w, w_D(a) = w\}$: a set of all the arcs originating at different WAs, and finishing at WA w. This is also used to define the indirect stream constraints.

Parameters:

- c_w: number of staff members required to operate one processing unit in WA w.
- b_i^k: demand for commodity i and node k. For the majority of nodes and commodities, this will be 0 - that is, flow simply passing through here. At the source nodes, this will be positive, as this is where flow enters the network. For sink nodes, this will be negative, indicating where flow leaves the network.

- v_a^k: commodity specific capacity for arc a and commodity k. These are used to enforce that the commodities follow the correct mappings.

- $\rho_{w,w'}$: proportion of total flow passing from WA w to WA w' from indirect mapping $(w, w') \in \mathcal{I}$.

- u_w: total WA processing capacity per time period per processing unit for WA w.

- C_{wt}: processing unit capacity for WA w in time t.

Decision variables

- x_a^k: amount of commodity k sent along arc a.

- y_{wt}: number of processing units rostered in WA w for time t. Here, we use the term 'processing unit' to describe one either person or machine that is used to sort the commodity in a WA.

- g_j: auxiliary continuous variable, giving the number of workers required for shift j.

- S_{wt} and F_{wt}: auxiliary binary variables indicating if WA w has started and finished in or before time t, respectively.

Formulation

$$\min \sum_{j \in \mathcal{J}} g_j \tag{3.4.1}$$

$$\text{s.t. } g_j \geq \sum_{w \in \mathcal{W}} c_w y_{wt}, \ \forall t \in \mathcal{T}_j, \forall j \in \mathcal{J} \tag{3.4.2}$$

$$\sum_{a \in \delta^+(i)} x_a^k - \sum_{a \in \delta^-(i)} x_a^k = b_i^k, \ \forall k \in \mathcal{K}, i \in \mathcal{N} \tag{3.4.3}$$

$$x_a^k \leq v_a^k, \forall a \in \mathcal{A}, \ \forall k \in \mathcal{K} \tag{3.4.4}$$

$$\sum_{k \in \mathcal{K}} \sum_{a | (w, W_d(a)) \in \mathcal{E}} x_a^k \leq y_{wt} u_w, \forall w \in \mathcal{W}, \ \forall t \in 1, ..., T \tag{3.4.5}$$

$$\sum_{a \in ID^+(w,w',t)} x_a^k = \rho_{w_1 w_2} \sum_{a \in ID^-(w)} x_a^k, \ \forall k \in \mathcal{K}, \ \forall (w, w') \in \mathcal{I}, \ \forall t \in \mathcal{T} \tag{3.4.6}$$

$$y_{wt} \leq C_{wt}, \ \forall w \in \mathcal{W}, t \in \mathcal{T} \tag{3.4.7}$$

$$y_{wt} \leq C_{wt} S_{wt}, \ \forall w \text{ s.t. } \exists \, w' \in \mathcal{W} \mid (w', w) \in \mathcal{H} \tag{3.4.8}$$

$$y_{wt} \leq C_{wt}(1 - F_{wt}), \ \forall w \text{ s.t. } \exists \, w' \in \mathcal{W} \mid (w, w') \in \mathcal{H} \tag{3.4.9}$$

$$S_{wt} \leq S_{w,t+1}, \ \forall t \in 2, ..., T, \tag{3.4.10}$$

$$F_{wt} \leq F_{w,t+1}, \ \forall t \in 2, ..., T \tag{3.4.11}$$

$$F_{wt} \geq S_{w't+1}, \ \forall t \in 2, ..., T, \ (w, w') \in \mathcal{H} \tag{3.4.12}$$

$$x_a^k \geq 0, \ \forall a \in \mathcal{A}, k \in \mathcal{K} \tag{3.4.13}$$

$$y_{wt} \in \mathbb{N}_0, \ \forall w \in \mathcal{W}, t \in \mathcal{T}$$

$$S_{wt}, F_{wt} \in \{0, 1\}, \ \forall w \in \mathcal{W}, t \in \mathcal{T} \tag{3.4.14}$$

The objective (3.4.1) is to minimise the total number of workers scheduled over a given day. When we are counting the number of workers in the facility at a time, we are counting the number of processing units in the manual WAs, plus the number of processing units in the machine WAs multiplied by the number of workers needed to work one machine in this WA.

Constraint (3.4.2) forces the g variables to take the value of the maximum number of workers in the WAs over the shifts.

The mass balance constraint (3.4.3) ensures that all mail that enters the network leaves the network, and that the inflow equals the outflow in all nodes (except for the source and sink nodes).

Constraints (3.4.4)- (3.4.6) enforce limits on the amount of flow travelling along arcs, but with slightly different purposes. (3.4.4) limits the commodity specific flow along each arc. This is used to ensure flow stays along the correct mappings. By contrast, Constraint (3.4.5) ensures that the flow leaving a node in a time period is limited by the number of processing units assigned to that WA in that time period. This links our staff rostering levels with our processing capacity. Constraint (3.4.6) ensures that the correct proportion of flow follows the indirect mappings.

The number of processing units assigned to each WA in each time period is limited to its capacity by Constraint (3.4.7).

Constraints (3.4.8)-(3.4.12) are used to enforce tethering. Constraints (3.4.8) and (3.4.9) ensure that no processing units are assigned in WAs before they start or after they finish. Constraints (3.4.10) and (3.4.11) are used to ensure that once a WA has started or finished for the day, it does not start or finish again. Constraint (3.4.12) ensures that if WAs w and w' are tethered, then w must finish before w' starts.

Finally, we define the types of decision variables used (including non-negativity) in Constraints (3.4.13)-(3.4.14).

3.4.3 Alternative objectives

Another objective is to ensure that shifts are as smooth as possible, which we achieve by minimising the total number of changes in workers between consecutive time periods in each WA. Let the decision variable $h_{wt}, w \in \mathcal{W}, t \in 2, ..., T$ define the change in processing units between time periods $t-1$ and t at WA w. To calculate h_{wt}, we then

add the following constraints to the model:

$$y_{w,t} - y_{w,t-1} \leq h_{wt},\ \forall w \in \mathcal{W}, \forall t \in 2, ..., T, \tag{3.4.15}$$

$$-(y_{w,t} - y_{w,t-1}) \leq h_{wt},\ \forall w \in \mathcal{W}, \forall t \in 2, ..., T. \tag{3.4.16}$$

Constraints (3.4.15) and (3.4.16) force h_{wt} to be greater than or equal to the change in processing units in WA w between the time periods t and $t-1$.

The objective to minimise the total number of changes in processing unit numbers between consecutive time periods (not counting changes between shifts) is therefore:

$$\min \sum_{w \in \mathcal{W}} \sum_{t \in \mathcal{T}} c_w h_{wt} \tag{3.4.17}$$

$$\min \sum_{w \in \mathcal{W}} \sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}'_j} c_w h_{wt}$$

where $\mathcal{T}'_j$ is a set of all time periods in $\mathcal{T}_j$ except for the first time period in $\mathcal{T}_j$. This is so the changes in staff numbers at each WA between shift changes is not counted. We also deal with meeting priorities as a soft constraint here. From the priorities listed previously, there are two types of priorities we need to consider - "All" and r priorities. For any WA with "All" as priority in shift j, we penalise any flow along the arc linking this WA during the last time in shift j to the node of this WA in the first time in shift $j+1$ (or to the sink, if j is the final shift). Mathematically, we write this as

$$\sum_{a \in \mathcal{A}_{All}} \sum_{k \in \mathcal{K}} x_a^k$$

where $\mathcal{A}_{All}$ is the set of all arcs which start at a WA with "All" priority in shift j at

the final time in shift j and finish at either the same WA in the next time period (if j is not the final shift) or the sink node (if j is the final shift).

The r priority WAs are more complex. For these WAs, we need to calculate the volume of flow processed over a shift, and then penalise any shortfall. To do this, we first introduce a variable $d_w \geq 0$ to represent the delayed flow in WA $w \in W$. This is defined through the constraint:

$$\sum_{t<T} \sum_{a \in \mathcal{A}_{wt}} x_a^k + d_w \geq r_w, w \in \mathcal{W}_R$$

where $\mathcal{A}_{wt}$ is the set of all arcs originating at node (w, t) and terminating at $(w', t+1)$ where $(w, w') \in \mathcal{E}$, $\mathcal{W}_R$ is defined as the set of WAs with an r priority, and r_w is the target value for WA $w \in \mathcal{W}_R$.

Priorities can then be enforced by minimizing the total delayed flow:

$$\sum_{a \in \mathcal{A}_{All}} \sum_{k \in \mathcal{K}} x_a^k + \sum_{w \in \mathcal{W}_R} d_w \tag{3.4.18}$$

There are various approaches to balancing multiple objective functions in optimisation. In the case where there is a strict order of importance for the objectives, a natural approach to use is lexicographic optimisation. In lexicographical optimisation, the model is first solved with one objective. Then, it is solved with the second objective, subject to keeping the first objective at its best value, and so on with other objectives. This is the approach we shall use in our numerical tests, in which, in particular, we will experiment with the order in which we optimise the maximum workers and smoothness objectives.

3.5 Results and discussion

To test our model, we applied it to data collected from a mail centre based in the United Kingdom (UKMC). We solved this model using four approaches, balancing the objectives differently in each. We compared our results to the actual workplans the UKMC would use, given the same forecasts of mail volume. We also use our model to investigate the effect of different mail scenarios has on staffing levels, as well as investigating the effect of the time granularity in our model has on outcomes.

3.5.1 Data and setting

We collected data from the mail centre for a three month period (June 29 to September 30, 2020). This data contained multiple data sets consisting of the following:

- Arrival volumes of each stream for each day.
- Origin WA, destination WA and stream for each mapping in the mail centre. For indirect mappings, the ratio of mail passing to that destination WA was also given.
- Capacity for the number of workers for each WA.
- Throughput for each WA.
- Sorting priority for each WA on each shift on each day.
- Number of workers in each WA for each 10-minute interval on each day suggested by the UKMC.

We used this data to build a network of the mail centre, then time-expanded this network out with 144 different time periods (a 24 hour day split into 10 minute intervals, starting at 6:00 am), and added the required sources and sinks. This new network

contains 8,127 nodes, connected by 45,163 edges. Note that in later experiments, we change the number of time periods we use to time-expand the network.

3.5.2 Experimental set-up

The shift manager faces a number of objectives - the throughput of delayed mail, the number of workers, and the number of changes. In this experiment, we will assume that the number of delayed mail is an absolute priorty. Therefore, we will minimise Objective (3.4.18) first, and lexicographically constraint the model to meet this minimum before meeting any other objectives. Since this is always the first priority, and we will always be meeting this before optimising with respect to any other objective, we will think of this objective as a constraint, and references to "objectives" in the remainder of this paper will refer to either maximum workers or changes.

For the maximum number of workers and smoothness objectives in (3.4.1) and (3.4.17), we test four approaches:

1. Minimising the maximum number of workers only (**MMWT**).
2. Minimising the total changes in staff levels at WAs between time periods (**MC**) only.
3. Lexicographically first minimising the maximum number of workers, then minimising the total changes (**Lex 1**).
4. Lexicographically first minimising the total changes, then minimising the maximum number of workers (**Lex 2**).

We also performed some experiments where we optimised the primary objective (either workers or changes), with varying bounds (not at the optimal value) on the secondary objective. From these experiments, this was deemed to not be a suitable approach to this problem. See Appendix 3.D.

We implement and solve all optimisation models using the Python interface to the Gurobi solver (version 9.0.0) (Gurobi Optimization, LLC, 2022) on a High Performance Computing Facility. For each day, we ran the model using 1 core and 3 hours per objective. For a full breakdown of the computational performance of the model, see Appendix 3.A. For the lexicographical approaches, if the model does not converge in this time limit, we take the current best objective value found, and use this as RHS for the lexicographical constraint.

3.5.3 Comparison between approaches, and with the UKMC

Firstly, we compare how the different approaches perform under each objective by solving the model for every day (excluding Sundays) for the three month period we consider. These were compared to each other, and to staff levels suggested by the UK mail company (**UKMC**). The UKMC sets their workplans with a basic algorithm. The details of this algorithm are given in Appendix 3.B.

Figures 3.5.1 and 3.5.2 show the maximum workers and changes (respectively) required for each day for the different approaches, as well as those recommended by the UKMC. Across both metrics, the broad picture is the same. Firstly, the approaches designed for a given metric (MMWT and Lex 1 for max. workers, MC and Lex 2 for changes) perform much better than those that do not. What is more noteworthy is how much better MC and Lex 2 perform regarding max. workers than MMWT and Lex 1 do regarding changes. This seems to be because there are already some constraints (e.g. Constraint (3.4.7)) to limit how many workers are in WAs at different times. However, there are no constraints limiting changes. Therefore, if left unchecked, there is more scope for a model to schedule many changes than to schedule many workers. Secondly, there are many instances of our approaches outperforming the UKMC. While this occurs for any approach designed for a specific objective (e.g. MMWT or Lex 1 for max. workers), we see that Lex 2 outperforms UKMC on both metrics, showing

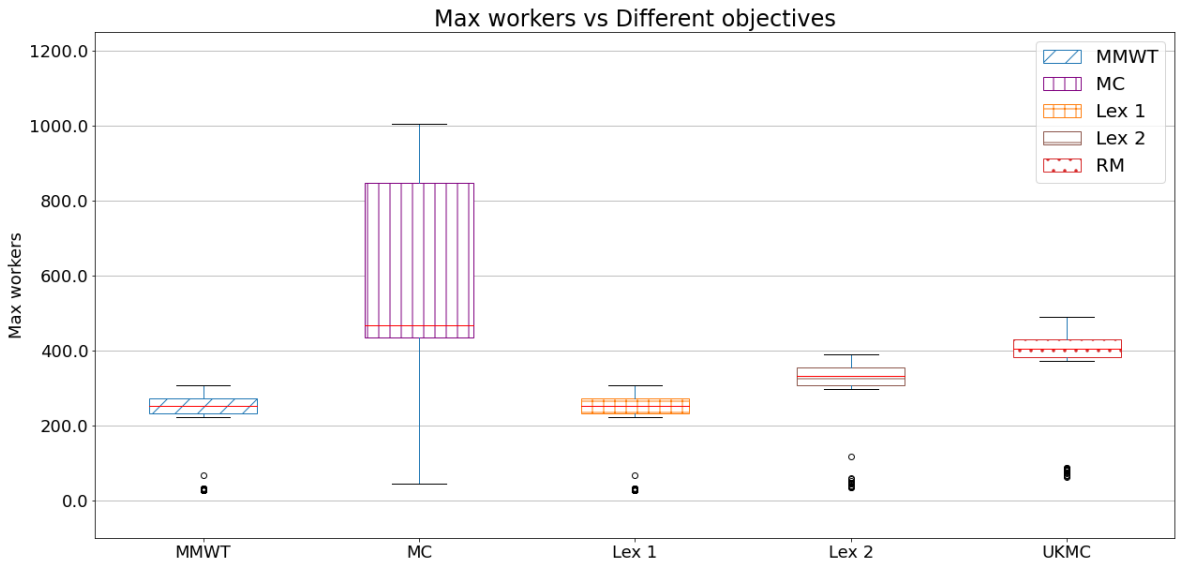


Figure 3.5.1: Boxplots of maximum workers for each day for the different approaches, as well as those recommended by the UKMC.

the value of this approach. Note that to see the differences in the changes between the approaches more clearly, we re-create Figure 3.5.2 without MMWT or Lex 1. This is shown in Figure 3.C.1 in Appendix 3.C.

3.5.4 Increasing mail volumes

As well as producing work plans, the proposed optimisation model can be used to investigate the effect of different scenarios. One scenario is a potential increase in total mail traffic, which happened, for example, during the COVID-19 pandemic. In this experiment, we look at the effect of increasing mail volumes by 10% and 20% using our optimisation approach.

The effect of increasing mail volumes on max. workers and changes is shown in Figures 3.5.3 and 3.5.4, respectively. Broadly, both max. workers and changes increase as the mail volumes increase. However, the proportional increases are less than those of the mail volume increases. For example, Looking at MMWT and Lex 1 with respect to max. workers, the median max workers increase from 253 to 276.5 and 296.5 for 10% and 20% increases respectively. This is a 9% and a 17% increase – showing that

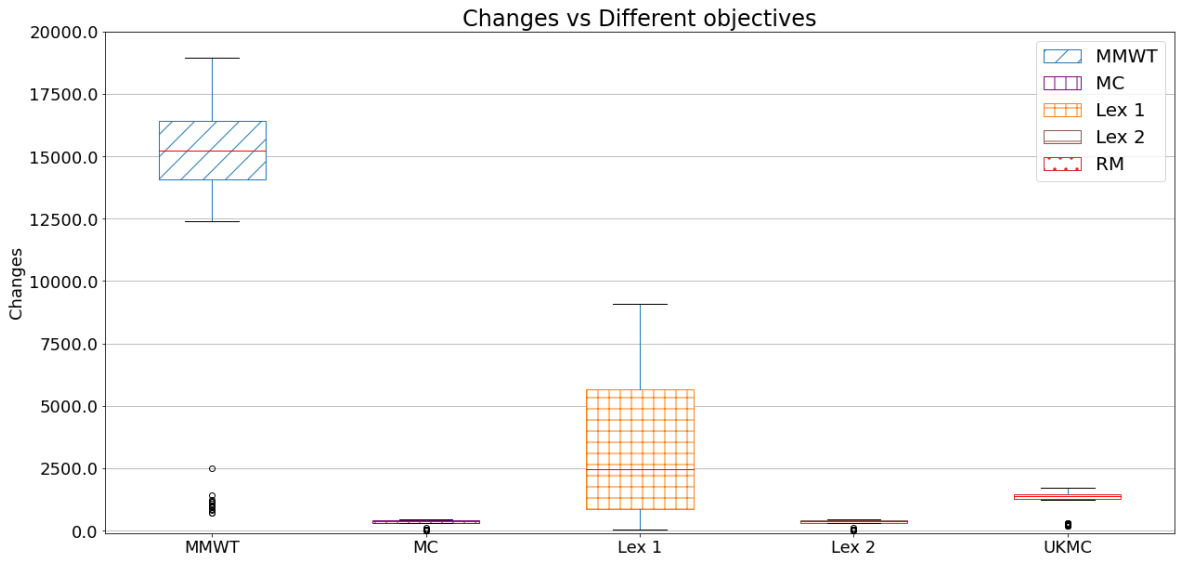


Figure 3.5.2: Boxplots of changes required for each day for the different approaches, as well as those recommended by the UKMC.

the workers become more efficient as the mail volumes are increased. We also see this for MC (approximately 9% and 12% increases) and Lex 2 (approximately 7% and 12% increases). Changes show a similar pattern. We see a (approximately) 6% and 11% increase in changes for MC and Lex 2, and an 8% and 14% increase for MMWT for the respective 10% and 20% increases in mail volumes. An exception to this pattern is the result for Lex 1 with respect to changes. In this case, the median changes seem to decrease as mail volumes are increased, albeit with increasing spread. For the other three approaches however, the pattern is clear.

3.5.5 Changing proportion of letters vs parcels

Another scenario is a reduction in the proportion of the mail volumes which are letters, rather than parcels. This scenario mimics the current long-term decline in the number of letters sent, and an increase in the number of parcels. In our sample, an average day consists of roughly 80% letters and 20% parcels. In this experiment we explore what happened if (while keeping the total volumes the same), we reduced to proportion of letters to 50% and 20%, along with the corresponding increase in parcels.

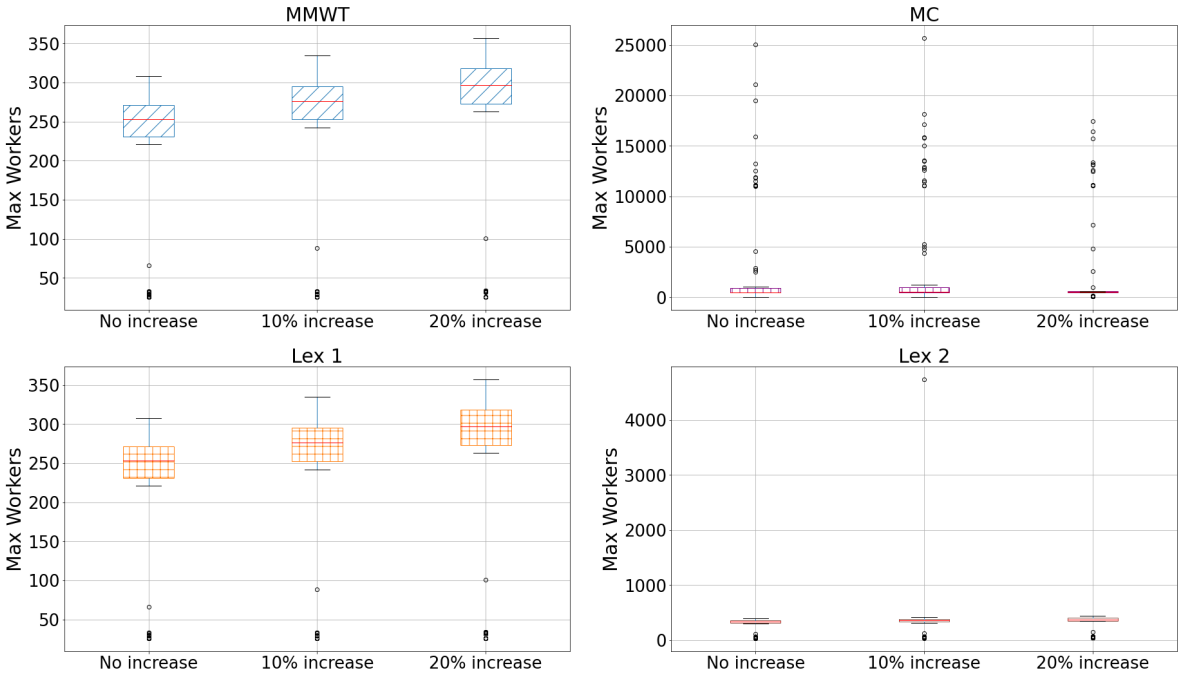


Figure 3.5.3: Max workers vs mail volume increase for different approaches.

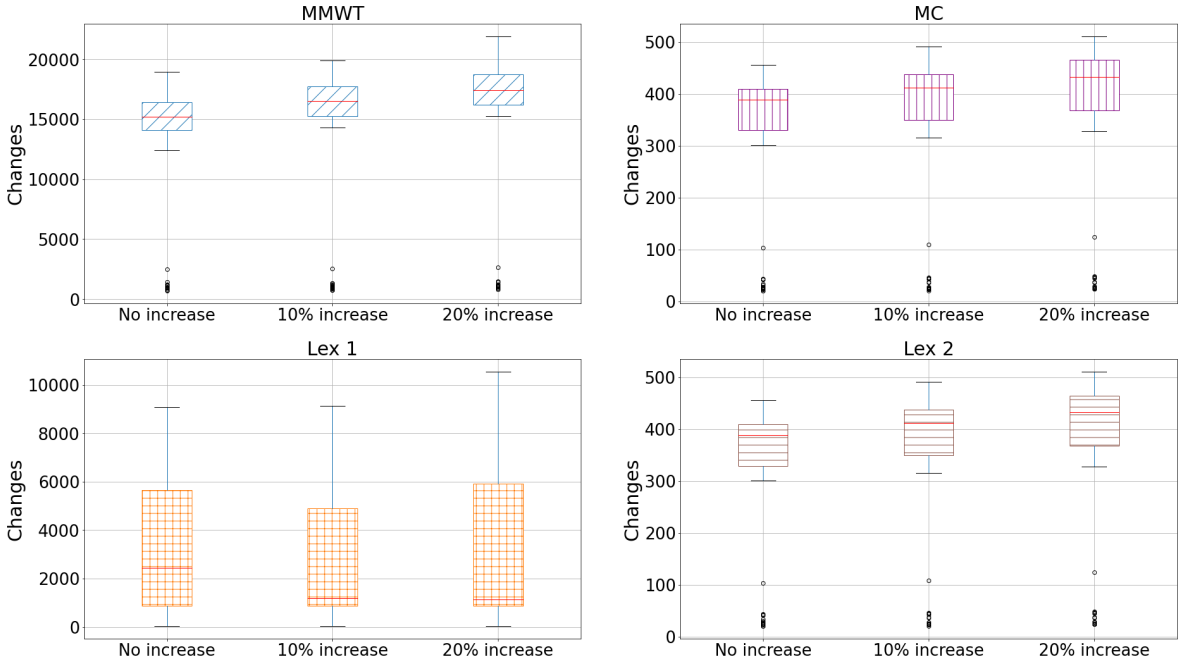


Figure 3.5.4: Changes vs mail volume increase for the different approaches

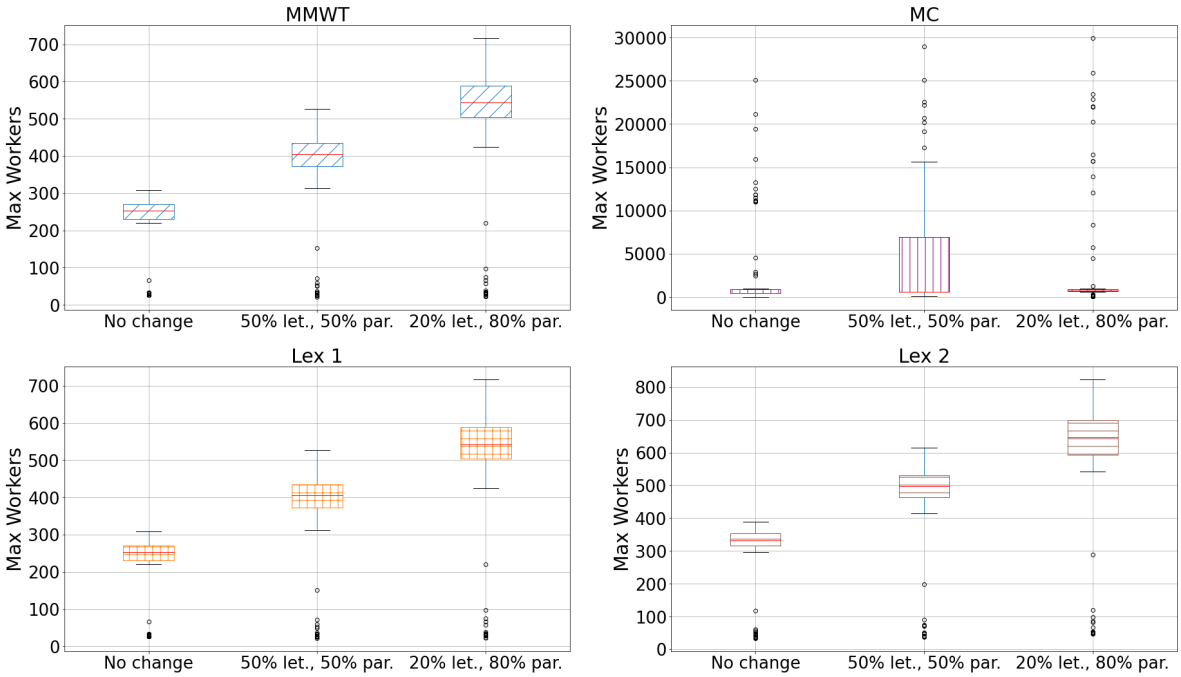


Figure 3.5.5: Max workers vs proportion of letters. 80% letters is the current baseline

For MMWT, Lex 1, and Lex 2, there is a clear increase in the maximum workers required when the proportion of letters decreases (see Figure 3.5.5). This suggests that parcels are more labour-intensive, and that workforces need to take this into account going forward. The trend for MC is harder to see from Figure 3.5.5, but the median max workers also increases with an increase in parcels (464, 642, and 775 workers for 80% letters, 50% letters, and 20% letters respectively).

The relationship between changes and proportion of letters/parcels is shown in Figure 3.5.6. Again, we see an unusual pattern in the Lex 1 results, in that the median seems to decrease while the spread increases. However, for the other three approaches, the pattern is clear. The changes increase as the number of parcels is increased. Given that the max. workers increase with more parcels, this means that there is more scope to have large changes in numbers.

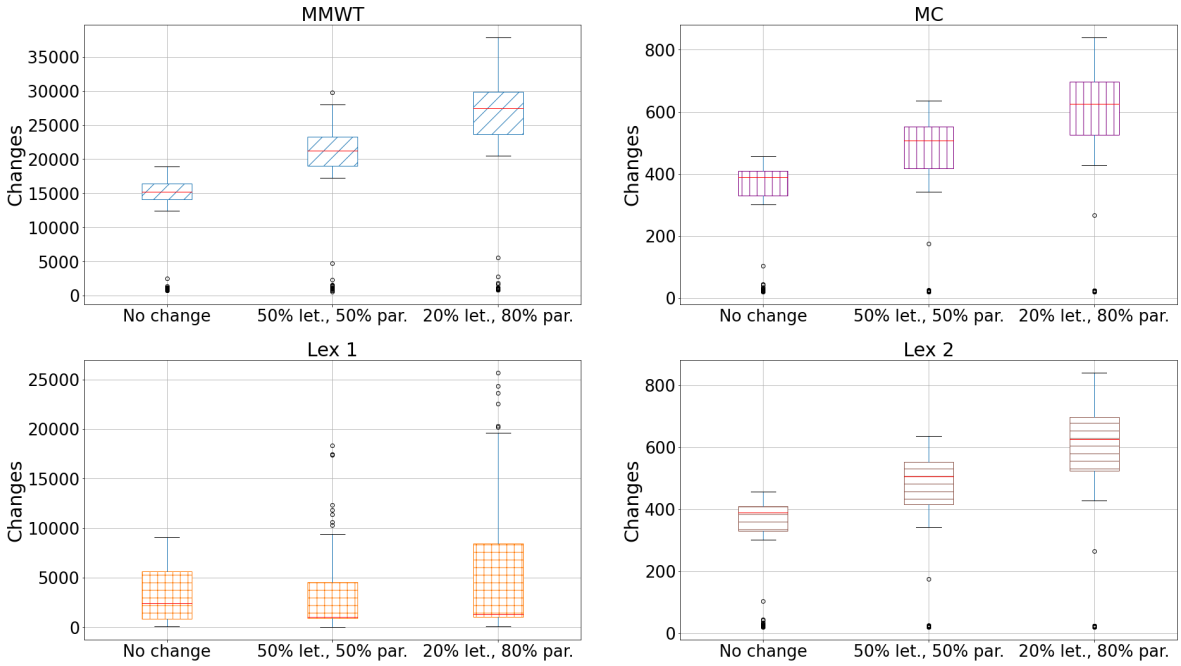


Figure 3.5.6: Changes vs proportion of letters. 80% letters is the current baseline

3.5.6 Reducing the time granularity

Given the complexity of the model, there were some computational challenges. In order to ease these, we also examined how the model behaved when using only 48 or 24 time periods (half hourly and hourly time intervals, respectively) when time-expanding the model.

The results are shown in Figures 3.5.7 and 3.5.8. There are two important aspects to note. Firstly, the number of changes strongly reduces as the length of the time periods increases, across all approaches. Going from 10 minute intervals (144 time periods) to 30 minute intervals (48 time periods) gives reductions in changes ranging from 34% (MC and Lex 2) to 71% (MMWT). Going from 10 minute to 1 hour intervals (24 time periods) results in reductions ranging from 67% (MC and Lex 2) and 88% (MMWT). If there are fewer distinct time periods, there are fewer chances to change worker numbers.

However, this comes at a cost of increasing the max. workers. For example, the max. workers required for MC increased from a median of 464 to 555.5 (20% increase), when decreasing from 144 to 24 time periods. We also observe an 11% increase for Lex

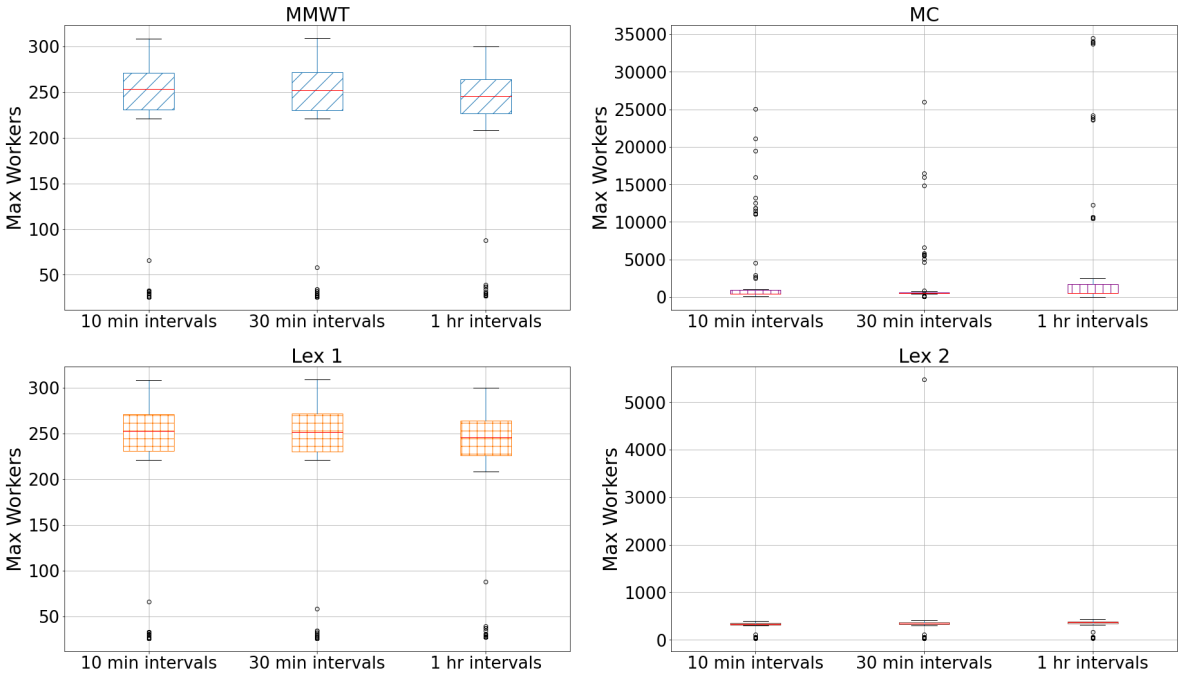


Figure 3.5.7: Max workers vs different levels of time discretisation, split by approach

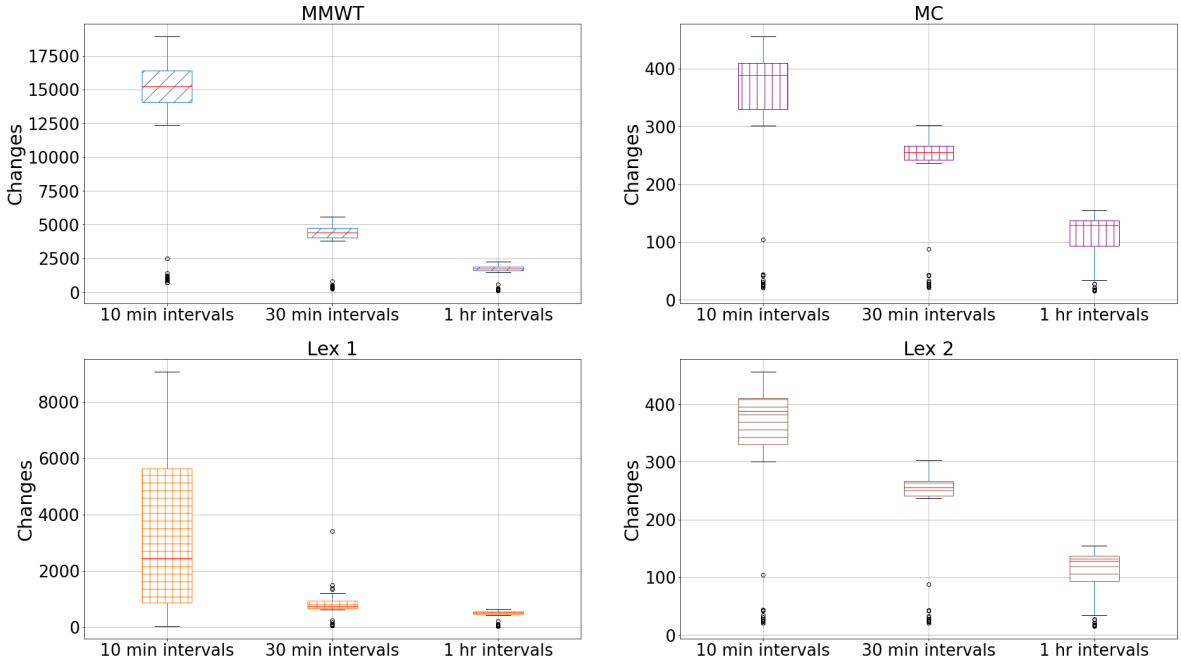


Figure 3.5.8: Changes vs different levels of time discretisation, split by approach

2 (333 workers to 370), and a 3% decrease for MMWT and Lex 1. This is important - by considering longer time intervals, we reduce changes, but this requires more max. workers (though proportionally less) when considering MC and Lex 2 and sometimes even decreasing maximum workers required (for MMWT and Lex 1) because there is a more balanced spread of workers across the WAs due to the longer time intervals (i.e., a worker stays on for longer at the same WA).

3.6 Conclusion

Staffing in sequential sorting facilities is important, as upstream delays will propagate through the facility. We present a new model to solve this problem for several objectives that are important in practice. This is based on a network flow model, but incorporates additional complexity, such as indirect streams and tethered WAs. We also account for time-dependency in sorting deadlines using a time-expanded network, at a very fine-grained timescale. This poses additional challenges as it increases the size of the network, and hence the model complexity.

We apply this model to data collected at a mail sorting facility over a time period of 3 months. The data includes daily mail volumes for different streams of mail (e.g. first class letters, second class letters), suggested staff workplans provided by the company, and details on sorting constraints for each stream. We compared the different approaches with each other, as well as against staff levels suggested by the UKMC. We also examined how the model performs under different scenarios. Namely, when mail volumes are increased, when the split between letters and parcels changes, and when the level of time discretisation is decreased.

When comparing all objectives, we naturally found that MMWT and Lex 1 were the best approaches regarding maximum workers, and MC and Lex 2 were the best regarding changes. However, Lex 2 consistently outperformed the UKMC on both

considered objectives.

Increasing mail volumes resulted in more workers required, but the proportional increase in workers was less than that of the increase in mail. Changes also showed increases as the mail volumes increased.

The proportion of parcels vs letters also had an interesting effect on the results. Both max workers and changes increased as the proportion of parcels increased. This suggest that parcels are more labour-intensive to sort. Given the long term increasing trend in parcel volumes (and decreasing trend in letter volumes), this is important information for practitioners.

Finally, when changing the time granularity, reducing the number of time periods has differing effects, depending on the approach used. For Lex 2 and MC, there is a trade-off when reducing the time granularity, between reducing the number of changes, but increasing the max. workers. For Lex 1 and MMWT, the number of changes reduces, but there is also a slight reduction in the max. workers. This shows that there can be benefits to having a reduced time discretisation, but it is approach-dependent.

While the results are positive, there are some limitations with this work. We assume that the mail volumes are known before staff allocation. This is often unrealistic in practise. An avenue for future research would be to relax this assumption. For example, this could be formulated as a stochastic problem, with mail volumes treated as random variables, and including probabilistic constraints. Such an approach would need greater computational power to solve the model. Using or developing more sophisticated optimisation techniques to solve this is another area for future research.

We contribute to the staff rostering literature by showing the value from using network models to set short-term operational staff levels for sequential sorting facilities. We also highlight the importance of examining the priorities of differing objectives.

From a managerial perspective, our model allows decision makers to determine work plans balancing several objectives for their day to day operations. By analysing several

scenarios and potential future trends, such as decreasing letter and increasing parcel volumes, decision makers can assess whether their current setup is appropriate to meet future challenges or whether adaptations are required.

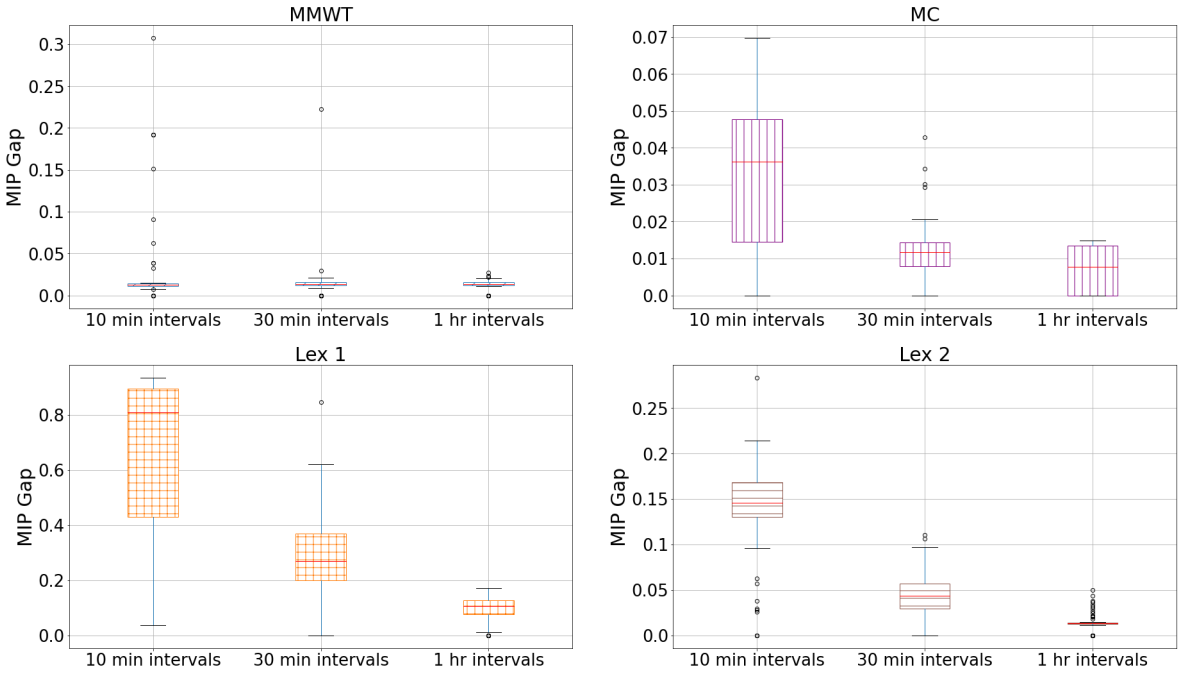


Figure 3.A.1: Optimality gap of the models vs time granularity, split by objective.

3.A Computational performance

To examine the computational performance of the model, we report the optimality gap for each objective for each day. Given the effect that the level of time discretisation should have on these gaps, we plot them against time granularity, split by objective in Figure 3.A.1.

Note that the gaps for 10-minute intervals should be representative of the gaps for all other experiments. We see that the gaps are very large for the Lex 1 objective, and (while smaller) are also still large for Lex 2. The gaps for MMWT and MC are much smaller, with MMWT having a median gap of 1.2%.

We see significant improvement in the gaps for MC, Lex 1, and Lex 2 as the level of time granularity is decreased (MMWT does not improve, primarily given how well it performs already).

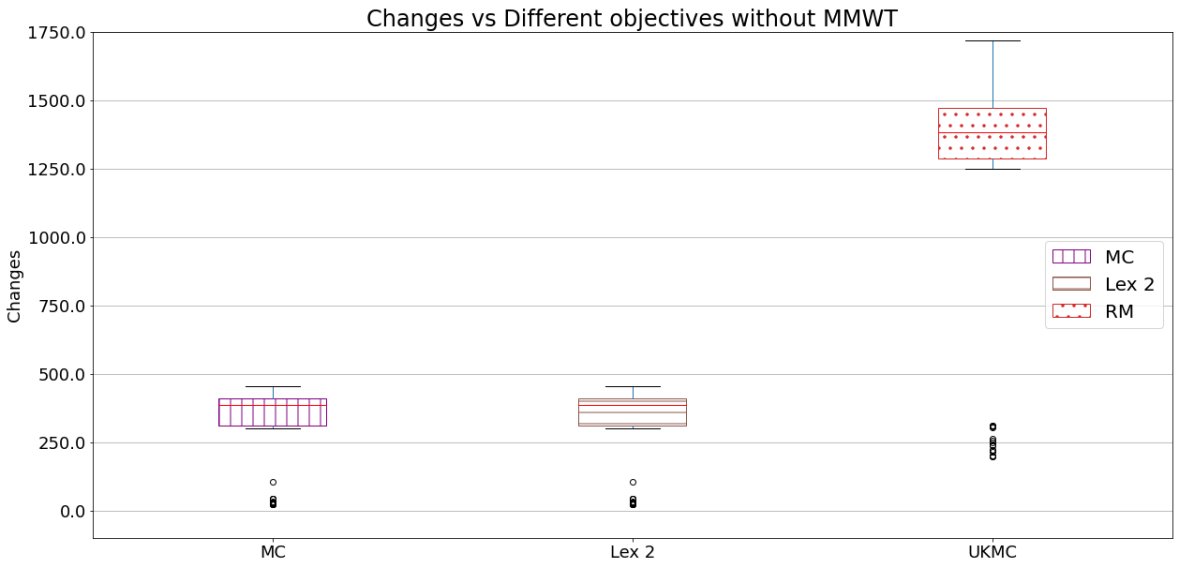


Figure 3.C.1: Boxplot of changes for the different objectives with MMWT and Lex 1 removed, for easier comparison.

3.B UKMC algorithm for setting staff levels

The UKMC sets their workplans with an algorithm which:

1. Smooths the arrival of mail evenly between all time periods between its arrival time and the arrival time of the next delivery of mail.

2. Calculates the amount of mail going through each work area at each time period.

3. Divides the calculated mail traffic by the given throughput per processing unit (in this case, sorting machines) to obtain the number of machines.

4. Multiplies the number of machines by the number of workers required to work one machine to calculate the final number of workers in each time period in each work area.

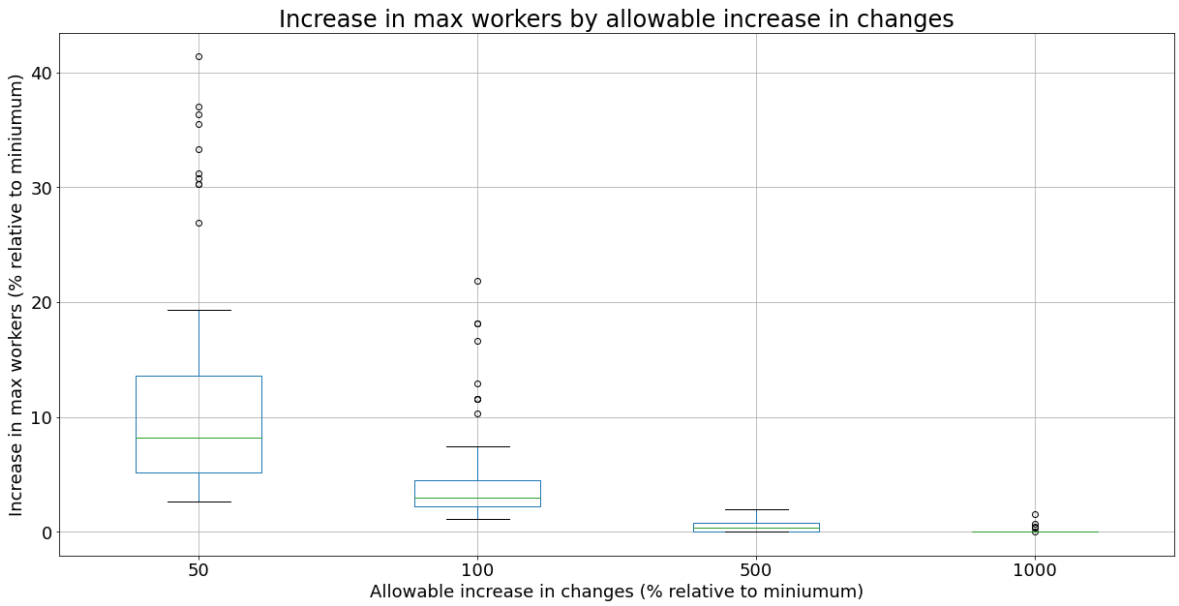


Figure 3.D.1: Increase in max workers by allowable increase in changes

3.C Results with omissions

Figure 3.C.1 shows the total changes for the MC and Lex 2 objectives, along with those suggested by the UKMC. It is now clear that MC and Lex 2 show lower changes than the UKMC.

3.D Constraining the secondary objective

We examine the effect of applying a bound on the secondary objective at various levels when minimising the first objective. This prevents the model from finding solutions which, while performing well on one objective, are extremely poor regarding the other. We set the constraints to be at most 50%, 100%, 500%, and 1000% worse than the optimal value for the secondary objective. We then look at how much proportionally worse the primary objective becomes relative to its optimal value (which we found previously) when imposing this constraint.

As can be expected, tightening the bound on changes causes the minimum maximum number of workers to become worse (see Figure 3.D.1). This shows that while we can

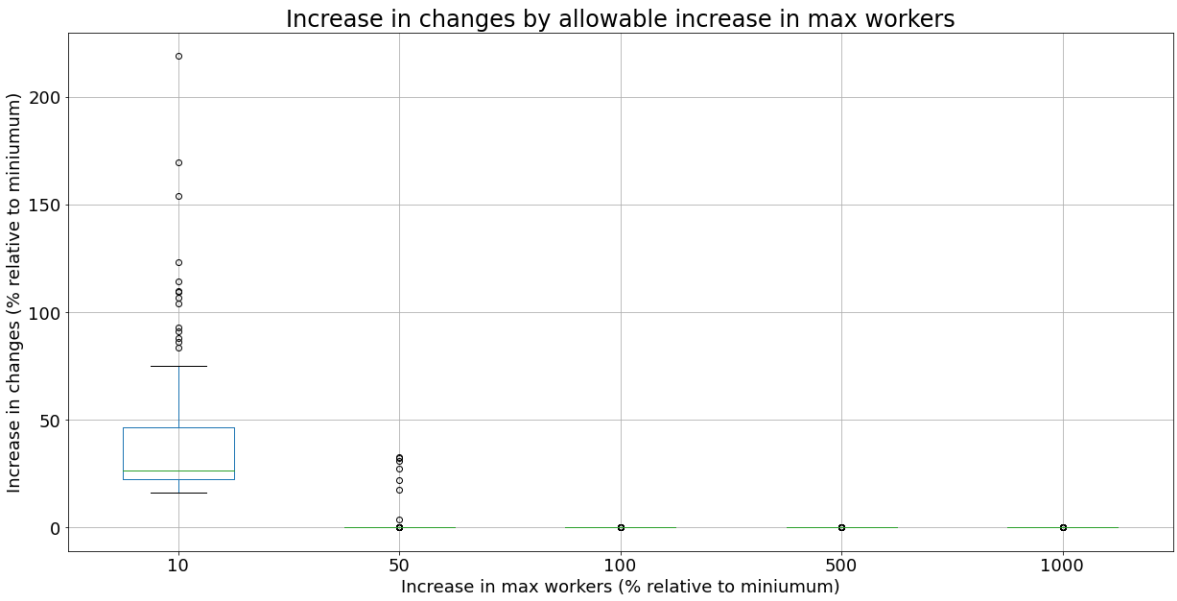


Figure 3.D.2: Increase in changes by allowable increase in max workers

limit how many changes the model sets, this does come at a cost. The median increase in the max workers is 0.3% when the changes have to be within 500% of the minimum, 3% if the changes have to be within 100%, and 8% when the changes have to be within 50%. We see similar results when changes is the primary objective. The main difference is that the bound on max workers needs to be much tighter to start affecting the number of changes (Figure 3.D.2). For 100%, 500%, and 1000% bounds, the median increase in the optimal number of changes is 0%. This shows that the minimum number of changes is less sensitive to these constraints on the max workers. We also included a bound of a 10% increase in the maximum workers, to see when the bound was tight enough to start affecting the workers. With the 10% increase, the median changes were 26% above their lowest possible values, showing that at this point, the bound is tight enough to start affecting the changes. While this can be useful in practice, an important question is which value for a bound is appropriate to avoid setting it arbitrarily. For this decision to be made, the whole optimisation model would need to be run first. In contrast, lexicographical optimisation ensures that (given the priorities of the objectives) we are achieving the lowest possible on one objective, and conditionally the lowest possible

value on the second objective.

Chapter 4

Staff allocation in mail centres under uncertain mail volumes

4.1 Introduction

In staff rostering, a shift manager needs to determine how many staff to roster onto a shift to complete all required tasks. Rostering decisions are often based on the amount of work that needs to be done. However, this information is often unknown before the staff workplans need to be set. This leaves the shift manager in a delicate position, needing to determine the staff levels before having all the information they need to make the optimal choice.

This is especially important in sequential sorting facilities. In these facilities, *streams* of materials need to be sorted at different *work areas* or *WAs* in sequence in order to be sorted/processed. Given the sequential nature of the required sorting within these facilities, understaffed WAs early in the process can cause delays and inefficiencies which will propagate through the system. In addition, the optimal staff levels will heavily depend on the amount of material to be processed. Variation in the volume of this material will have consequences in staffing these facilities. Furthermore, the rostering

decisions often have to be made before the exact amount of material to process is known.

Mail centres are one example of such facilities. These are the facilities in the post delivery system where the majority of sorting occurs. Within each mail centre, there are many steps to the process:

1. Accepting mail when it arrives.
2. Within the letters and parcels (these arrive already segregated), separate mail based on:
    - Size. For example, sorting regular-sized envelopes from larger sized envelopes (flats), or sorting parcels of different dimensions.
    - Machinable vs non-machinable. Machinable items refer to those that can be automatically read and sorted by sorting machines. Non-machinable items cannot be sorted automatically, and instead must be sorted by hand. For example, parcels of irregular dimensions and shapes, or letters with hard-to-read addresses.
3. 'Facing' letters and parcels. That is, stacking them in such a way that they can be fed into the automatic sorting machines.
4. Sorting items by destination. This can be any of:
    - Segregating 'outward' vs 'inward' mail. Outward mail is mail that needs to be sent to another mail centre closer to its destination, opposed to inward mail, which is addressed nearby.
    - Sorting mail into destinations and postcodes. This can have multiple steps, sorting the mail into finer geographical areas at each step.
    - 'Sequencing' the inward mail. This is a process where the individual letters within postcodes are sorted into delivery order based on the post officer's delivery route. After this stage, mail is ready to be delivered.

What makes this more complicated is that there are multiple different streams of mail. These streams differentiate mail on a combination of type (e.g. regular-sized envelops vs flats) and priority (e.g. first vs second class). The different streams all need to pass through different areas in a specific order to be sorted. Furthermore, there are specific time deadlines and service requirements for the different streams. For example, 93% of all first class letters need to be delivered the next working day. Additional operational constraints make this problem even more complex. These include the splitting of flows, where the total volume of material sent to a WA needs to be split by given proportions to subsequent WAs; and tethering of WAs, where certain WAs cannot start processing mail until specific previous WAs have processed all their mail and closed for the shift.

However, there is considerable variation in daily mail volumes. Figure 4.1.1 shows the daily mail volumes for each day of the week (excluding Sundays) in one year period (April 2020 to March 2021) in a mail centre for a UK-based mail company (UKMC). This shows the considerable difference in volumes over the year, with some days showing a range of more than 2 million items between their heaviest and lightest volume days. Furthermore, the staff levels need to be decided *before* the exact mail volumes are known. This means that we need a method to determine the staff levels that takes account of the uncertainty of mail volumes.

In this chapter, we use a stochastic programming model to account for the uncertainty in mail volumes. First proposed by Dantzig (1955), stochastic programs are models in which some decisions are made before some uncertain parameters are known (first stage decisions) and some are made in response to observing the uncertain parameters (second stage decisions). This fits a shift manager having to set a workplan for a mail centre before knowing all mail volumes, and having to decide how to respond to this.

The main contribution of this chapter is to provide a novel modeling approach

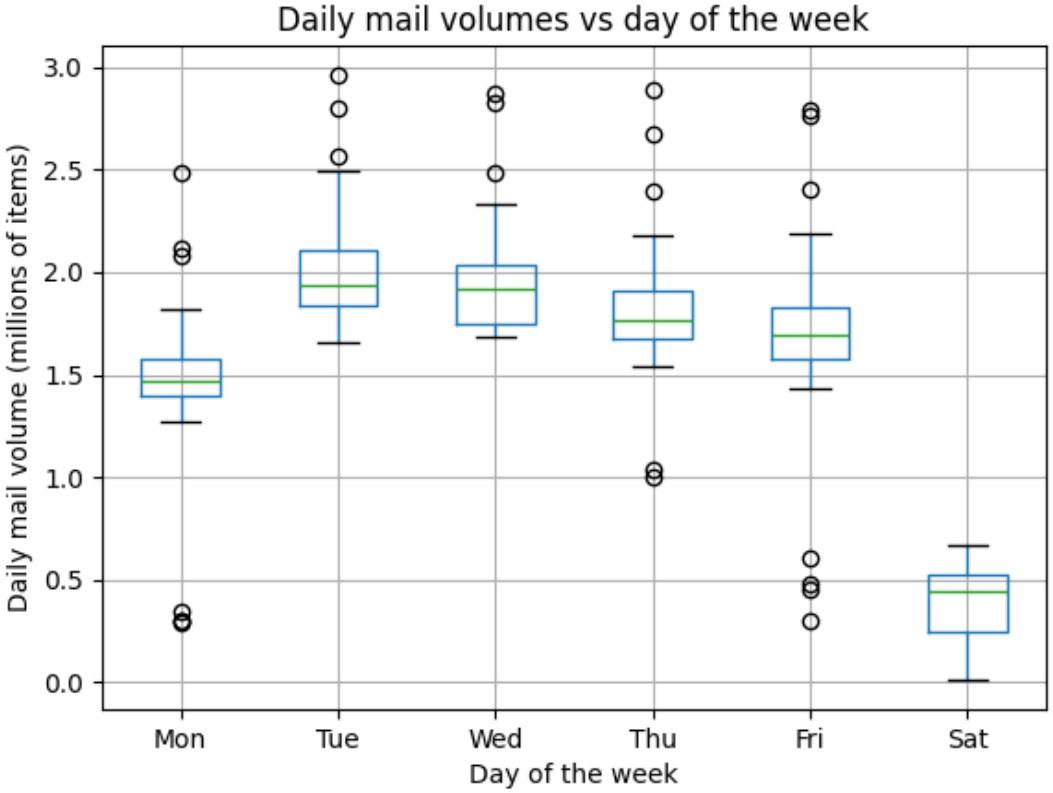


Figure 4.1.1: Boxplots showing the variation in daily mail volumes, split by day of the week. Data covers the 12 month period 30 March 2020 to 26 March 2021, taken from a large UK-based mail company.

for staffing in a sequential sorting facility that accounts for uncertainty in the stream volumes. Here we show the improvement that can be made when modelling this facility using varying numbers of discrete scenarios. Also, we relax the model slightly, and allow changes to the workplan in the second stage, and show the improvement that this can offer. We do this to demonstrate the benefits that can be gained by allowing more flexibility in the model. We compare the results from this model with the initial model where the workplan is set across all scenarios.

The outline of this chapter is as follows. In Section 4.2, we review current techniques and literature on this problem. Section 4.3 shows how we adapt current deterministic models to account for random mail volumes. We present our computational results in Section 4.4. Conclusions and further work is discussed in Section 4.5.

4.2 Literature review

Previous work has been done optimising multiple parts of the mail industry. This includes a variety of areas, from designing the optimal queueing system in a post office (Balachandran, 1977) up to designing the whole postal network (Song et al., 2009; Jarrah et al., 2016). Specifically to mail centres, there have been previous attempts to determine optimal staffing. Zhang and Bard (2005), Qi and Bard (2006), and Zhang et al. (2009) all attempt to determine optimal staff levels, along with scheduling the equipment to be used. These authors attempt to model the problem using job-shop and lot-sizing problems, solving it with LP and MIP-based methods. Zhang and Bard (2005) were the first to use this approach. Qi and Bard (2006) extended it, by using the LP model to solve the equipment scheduling problem, and a simulation model to solve the staffing problem. Zhang et al. (2009) then extend this further to adjust given schedules for disruptions.

As outlined in Chapter 3, another common approach is to use a network design

model, similarly used by Bard et al. (1993) and Jarrah et al. (1994a). These approaches model a mail centre as a network, and use a network design formulation to assign staff levels. Bard et al. (1993) and Jarrah et al. (1994a) take a much longer-term view of the mail centre, including variables to allow the purchasing of additional equipment, and optimising the cost of running the mail centre over a longer time horizon (years, rather than shifts). In Chapter 3 we focused on setting the staff levels, taking a more detailed approach towards the mail centre. We also modelled additional operational constraints, such as the tethering of different WAs, and proportionally splitting the flows of material between different downstream WAs. Finally, we also highlighted a number of possible different objectives for the mail centre staffing problem. These include minimising the number of required workers, maximising the 'smoothness' of a shift, or minimising the cost of purchasing new machines.

However, the majority of these models assume the mail volumes are known when optimising. For these models, the way to account for this uncertainty would be to assume the mail volumes would take some known values, such as their expected values. An exception to this is Jarrah et al. (1994a) who acknowledge the randomness in mail volumes, and account for it by solving the model with the expected mail volumes, increased by 25% as a buffer. However, this fails to acknowledge the difference in variation in different mail centres or different streams, meaning that a 25% buffer could be inappropriately high or low. For example, if mail volumes are frequently more than 25% higher than the averages (which we see in Figure 4.1.1), then the mail centre will routinely be understaffed. On the other hand, if volumes greater than this 25% buffer are rare, the mail centre will routinely be overstaffed. Furthermore, unless the streams are highly correlated, it is very unlikely that all will exceed their expectation by 25% on the same day. Therefore, assuming a 25% buffer on all streams will be overly conservative.

There are several possible ways to incorporate stochasticity. These include chance-

constrained programs (Charnes and Cooper, 1959), stochastic dynamic programs (Bellman, 1958) and stochastic programs (Dantzig, 1955). Chance-constrained models seem to line up well with the quality of service standards that mail centres need to adhere to, but unfortunately, frequently are non-convex and computationally intractable (Hong et al., 2011). Stochastic dynamic programs have the sequential nature that is present in the sequential sorting facility. However, these models also suffer from tractability issues (Powell, 2011).

As a result, we use stochastic programming, which we described in Chapter 2. This will better account for the variation in the mail volumes by explicitly including the expectation of the delayed mail in the objective function, improving on the work of Jarrah et al. (1994a) in handling the uncertainty in the mail centre problem.

While our model in Chapter 3 addresses the staffing problem well for deterministic mail volumes, this needs to be extended to account for randomness in the mail volumes. Stochastic programming seems to be an ideal approach to deal with this uncertainty. We therefore will extend our model from Chapter 3 with a stochastic programming approach. This will provide a better model for shift managers to use when solving this problem.

4.3 Problem description and mathematical model

4.3.1 Problem description

We propose a stochastic extension to the sequential sorting facility problem described in Chapter 3. This features the same concepts of mappings, WAs, and the same operational constraints. Our aim is to develop a workplan for the whole day, determining how many processing units are required in each WA at each time period. This needs to be done without knowing the exact mail volumes that will arrive on the given day. In addition, once the mail volumes become known, we also need to decide how to route the

mail through the mail centre across time, and how much mail will be left as delayed.

The ideal workplan will balance the number of processing units being used (as fewer processing units is cheaper) with having enough capacity to be able to sort the mail. Unlike the deterministic model, we cannot simply set a minimum required volume of mail to sort each shift as a capacity in the model. Given the variation in mail volumes, it would be either impossible or prohibitively costly to roster enough staff to ensure the minimum is met regardless of the observed mail volumes. Instead we have to try and minimise the expected amount of delayed mail on a given day.

The change to the problem setting is the inclusion of *scenarios*. Each scenario represents a set of possible mail volumes arriving at the mail centre on a given day. We assume that a set of potential scenarios is already available. For example, this set could be from existing empirical data, or forecast models. The exact mail volumes are only observed after setting the workplan. In a practical setting, this would parallel a shift manager having experience in daily mail volumes, using this experience in setting the workplan the week before the shift, and getting exact mail volumes the day of the shift.

We make a number of the same assumptions about the network structure and operations that we did in Chapter 3. A key assumption we make is that workers can switch WAs instantly. That is, they can be working at at WA in one time period, and another in the next. Furthermore, we assume that workers are able to work in any WA in the centre.

Regarding the scenarios, we assume that the only difference between the scenarios is the mail volumes. The mail centre structure and parameters (e.g. throughputs, capacities, cost of delayed items, etc) are assumed to be known constants. This assumption could be relaxed if the appropriate data was provided.

4.3.2 Stochastic programming model

Here we give the formulation for our stochastic programming model for staffing the mail centre. We base our model on the deterministic model, shown in Chapter 3, making the necessary adjustments to extend it to a stochastic program.

Deterministic mail centre model

We recall that the deterministic model is a network design model consisting of a base network $\mathcal{B} = (\mathcal{W}, \mathcal{E})$ describing the mappings ($\mathcal{E}$) between a set of WAs ($\mathcal{W}$) in a network. We then time-expand this network for our desired number of time periods, and add the appropriate source and sink nodes. We also add a dummy 'completion' WA, to 'hold' the mail that has been completely sorted.

Each node (aside from the sources and sinks) represents a WA w at a given time period t. There is also an associated demand b_i^k associated with each node i and each commodity k. For the majority of the nodes in the network, this demand will be 0. That is, the flow is only allowed to pass through the node. For the sources (and sinks), the demands of the commodity give the (assumed) volume of that stream of mail in the mail centre for that day.

The arcs in the network represent movement of mail between both WAs and time periods. These show the possible ways that mail can move in the mail centre. For example, if mail is processed at WA w in time t and then moves to WA w', this mail travels along arc $(w,t),(w',t+1)$. If it is not processed, and remains in WA w, it travels along arc $(w,t),(w,t+1)$. Each arc has a commodity-specific capacity v_a^k, to direct streams along the correct mappings. The arcs coming from node $i = (w,t)$ also have a joint capacity, proportional to the number of processing units scheduled in WA w at time t. Finally, a number of arcs are designated as penalty arcs. These are arcs leaving WAs where mail must be sorted in the final time period of a shift, but not connected to downstream WAs in the network. Instead, these arcs either connect to the sink, or to

the same WA in the first time period of the next shift. These WAs will have a user-set penalty for their uses.

There are two major decisions that the shift manager needs to make. The first consists of how many processing units to roster in each WA w for each time period t. These variables are denoted by y_{wt}. The second is how to direct the mail in the network throughout the whole time period. The amount of stream k to send along arc a is denoted by x_a^k. These two objectives are often in conflict with each other. Rostering more workers in the workplan has a higher cost, however, it increases the throughputs of the WAs. This means that less mail needs to be directed along the delay arcs, which results in penalties.

Extending the deterministic model to the stochastic model

The main change we need to make to the deterministic model is to include scenarios into the model. We introduce a new set of scenarios, $\mathcal{S}$. Each scenario $s \in \mathcal{S}$ has an associated probability p_s.

In practical terms, a scenario represents mail volumes for various commodities on a given day. In terms of our model, we can represent this by having different demands at the sources and sinks for the different scenarios. That is, we alter our node demands b_i^k to be b_i^{ks}, representing the demand for commodity i and node k in scenario s.

Having introduced the scenarios, we also need variables to show how the mail is directed around the network for each individual scenario. We do this by amending the flow variables. These variables now represent flows of commodities along arcs for specific scenarios. That is, we now add the s index to our x_a^k variables, to obtain x_a^{ks}, which represents the amount of stream k sent along arc a in scenario s.

Having established how to consider scenarios, we now present the full stochastic formulation for the mail centre problem.

Sets and indices

- $\mathcal{S}$: the set of scenarios in the model, indexed by s.
- $\mathcal{W}$: set of WAs indexed by w.
- $\mathcal{E}$: set of mappings between WAs.
- $\mathcal{I}$: set of indirect mappings.
- $\mathcal{H}$: set of tethered WAs.
- $\mathcal{J}$: set of shifts, indexed by j.
- $\mathcal{T}$: set of time periods, indexed by t.
- $\mathcal{T}_j$: set of time periods associated with shift j. Note, $\mathcal{T}_j \subset \mathcal{T}$.
- $\mathcal{N}$: the set of nodes in the time-expanded network, indexed by i.
- $\mathcal{A}$: the set of arcs in the time-expanded network, indexed by a.
- $\delta^+(i)$ and $\delta^-(i)$: the sets of outgoing and incoming arcs for node i, respectively.
- $\mathcal{K}$: the set of commodities, indexed by k.
- $w_O(a)$ and $w_D(a)$: the origin and destination WAs associated with arc a.
- $t_O(a)$ and $t_D(a)$: the origin and destination times associated with arc a.
- $ID^+(w, w', t) := \{a | w_O(a) = w, w_D(a) = w', t_O(a) > t, t_D(a) = t_O(a) + 1\}$: a set of all the arcs originating at WA w and finishing at w', originating at a time later than t. This is used in defining constraints to enforce the indirect steams.
- $ID^-(w) := \{a | w_O(a) \neq w, w_D(a) = w\}$: a set of all the arcs originating at different WAs, and finishing at WA w. This is also used to define the indirect stream constraints.

Parameters

- p_s: the probability of scenario s.
- c_w: the number of staff members required to operate one processing unit in WA w.
- b_i^{ks}: the demand for commodity i and node k in scenario s. For the majority of nodes and commodities, this will be 0. That is, flow is simply passing through here. At the source nodes, this will be positive, as this is where flow enters the network. For sink nodes, this will be negative, indicating where flow leaves the network.
- v_a^k: the commodity specific capacity for arc a and commodity k. These are used to enforce that the commodities follow the correct mappings.
- $\rho_{w,w'}$: the proportion of total flow passing from WA w to WA w' from indirect mapping $(w, w') \in \mathcal{I}$.
- u_w: the total WA processing capacity per time period per processing unit for WA w.
- C_{wt}: the processing unit capacity for WA w in time t.
- α: the weighting given to the first stage objective. We set $0 \leq \alpha \leq 1$.

Decision variables

- First-stage decisions:
  - y_{wt}: number of processing units rostered in WA w for time t. Here, we use the term 'processing unit' to describe one person or machine that is used to sort the commodity in a WA.

  - g_j: auxiliary continuous variable, giving the number of workers required for shift j. This is explained fully in Section 4.3.3
  - S_{wt} and F_{wt}: auxiliary binary variables indicating if WA w has started and finished in or before time t, respectively.

- Second-stage decision variables:
  - x_a^{ks}: amount of commodity k sent along arc a in scenario s.
  - d_w^s: The amount of r priority delayed mail generated by WA w in scenario s. Section 4.3.3 gives more details on this variable.

Formulation

$$\min \alpha \sum_{j \in \mathcal{J}} g_j + (1-\alpha) \sum_{s \in \mathcal{S}} p_s \left(\sum_{a \in \mathcal{A}_{All}} \sum_{k \in \mathcal{K}} x_a^{ks} + \sum_{w \in \mathcal{W}_R} d_w^s \right) \tag{4.3.1}$$

$$\text{s.t. } g_j \geq \sum_{w \in \mathcal{W}} c_w y_{wt}, \ \forall t \in \mathcal{T}_j, j \in \mathcal{J} \tag{4.3.2}$$

$$\sum_{a \in \delta^+(i)} x_a^{ks} - \sum_{a \in \delta^-(i)} x_a^{ks} = b_i^{ks}, \ \forall k \in \mathcal{K}, i \in \mathcal{N}, s \in \mathcal{S} \tag{4.3.3}$$

$$x_a^{ks} \leq v_a^k, \forall a \in \mathcal{A}, k \in \mathcal{K}, s \in \mathcal{S} \tag{4.3.4}$$

$$\sum_{k \in \mathcal{K}} \sum_{a | (w, w_D(a)) \in \mathcal{E}} x_a^{ks} \leq y_{wt} u_w, \forall w \in \mathcal{W}, s \in \mathcal{S} \tag{4.3.5}$$

$$\sum_{a \in ID^+(w,w',t)} x_a^{ks} = \rho_{w_1 w_2} \sum_{a \in ID^-(w)} x_a^{ks}, \ \forall k \in \mathcal{K}, \ (w, w') \in \mathcal{I}, \ t \in \mathcal{T}, s \in \mathcal{S} \tag{4.3.6}$$

$$y_{wt} \leq C_{wt} S_{wt}, \ \forall w \text{ s.t. } \exists \, w' \in \mathcal{W} \mid (w', w) \in \mathcal{H} \tag{4.3.7}$$

$$y_{wt} \leq C_{wt}(1 - F_{wt}), \ \forall w \text{ s.t. } \exists \, w' \in \mathcal{W} \mid (w, w') \in \mathcal{H} \tag{4.3.8}$$

$$S_{wt} \leq S_{w,t+1}, \ \forall t \in 2, ..., T, w \in \mathcal{W} \tag{4.3.9}$$

$$F_{wt} \leq F_{w,t+1}, \ \forall t \in 2, ..., T, w \in \mathcal{W} \tag{4.3.10}$$

$$F_{wt} \geq S_{w't+1}, \ \forall t \in 2, ..., T, \ (w, w') \in \mathcal{H} \tag{4.3.11}$$

$$\sum_{t<T} \sum_{a \in \mathcal{A}_{wt}} x_a^{ks} + d_w^s \geq r_w, w \in \mathcal{W}_R, \forall s \in \mathcal{S} \tag{4.3.12}$$

$$x_a^{ks} \geq 0, \ \forall a \in \mathcal{A}, k \in \mathcal{K}, s \in \mathcal{S} \tag{4.3.13}$$

$$y_{wt} \leq C_{wt}, \ \forall w \in \mathcal{W}, t \in \mathcal{T} \tag{4.3.14}$$

$$y_{wt} \in \mathbb{N}_0, \ \forall w \in \mathcal{W}, t \in \mathcal{T} \tag{4.3.15}$$

$$S_{wt}, F_{wt} \in \{0, 1\}, \ \forall w \in \mathcal{W}, t \in \mathcal{T} \tag{4.3.16}$$

Objective (4.3.1) minimises the maximum workers in the mail centre, and the expected cost of the delayed mail over the scenarios (we discuss our choice of objective in Section 4.3.3). Constraint (4.3.2) forces the auxiliary g variables to take the correct values. Constraint (4.3.3) is the mass balance constraint for the flows in the network.

The correct mappings are enforced by the commodity-specific arc capacity constraints (4.3.4). Constraint (4.3.5) has two important functions. Firstly, it links the number of processing units rostered in a WA with its sorting capacity. Secondly, it forces the use of the same workplan across all scenarios. The indirect streams are enforced with Constraints (4.3.6). The tethered WAs are enforced with Constraints (4.3.7), (4.3.8), (4.3.9), (4.3.10), and (4.3.11). Constraint (4.3.12) is used to help control the auxiliary d variables, explained more in Section 4.3.3. Lastly, Constraints (4.3.13), (4.3.14), (4.3.15), and (4.3.16) define the variable types and bounds.

4.3.3 Objectives

Stochastic programs often have two objectives - one to govern the first stage, and one for the second. Here, we discuss our choice for these objectives, as well as a proposed method to balance them.

First stage objective

For this chapter, we will minimise the number of workers required for each shift as our first-stage variable. This is done by minimising the maximum number of workers required in each shift, as was done in Chapter 3. This is represented by the variable g_j and enforced by Constraint (4.3.2), with the objective function given by:

$$\sum_{j \in \mathcal{J}} g_j.$$

This objective minimises the maximum workers in the mail centre at any one time period (referred to as MMWT).

Second stage objective

The first stage objective focuses on the staff workplan. However, we also need to measure how this workplan performs in the second stage once the random mail volumes become known. The most obvious is to minimise delayed mail - that is, mail that should be sorted within a shift, but has to be passed to the next shift. In the deterministic model in Chapter 3, this was calculated by counting mail that travelled on the arcs of the time-expanded network that span the change of shifts. We do the same in the stochastic model, but now we do this for each scenario.

From the priorities listed in Chapter 3, there are two types of priorities we need to consider - "All" and r priorities. For any WA with "All" as priority in a given shift, we penalise any flow along the arc linking this WA at the final time in this shift to the same WA in the next time period (or, in the case of the final shift, to the sink node). Mathematically, we write this as

$$\sum_{a \in \mathcal{A}_{All}} \sum_{k \in \mathcal{K}} x_a^{ks}$$

for each scenario s. We define $\mathcal{A}_{All}$ as the set of all arcs which start at a WA with "All" priority in the final time period of a shift, and finish at either the same WA in the first time period of the next shift, or the sink node.

The r priority WAs are more complex. For these WAs, we calculate the volume of flow processed over a shift, and then penalise any shortfall. To do this, we first introduce a variable $d_w^s \geq 0$ to represent the delayed flow in WA w in scenario s. This is defined through the constraint:

$$\sum_{t<T} \sum_{a \in \mathcal{A}_{wt}} x_a^{ks} + d_w^s \geq r_w, w \in \mathcal{W}_R, \forall s \in \mathcal{S}$$

where $\mathcal{A}_{wt}$ is the set of all arcs originating at node (w, t) and terminating at $(w', t+1)$

where $(w, w') \in \mathcal{E}$, $\mathcal{W}_R$ is defined as the set of WAs with an r priority and r_w is the target value for WA $w \in \mathcal{W}_R$.

Having defined these variables, the objective which would give this is:

$$\sum_{s\in\mathcal{S}} p_s \left(\sum_{a\in\mathcal{A}_{All}} \sum_{k\in\mathcal{K}} x_a^{ks} + \sum_{w\in\mathcal{W}_R} d_w^s \right) \tag{4.3.17}$$

We refer to this objective as the delayed mail cost, or DMC.

Balancing objectives

In stochastic programs, the first and second stage objectives are often added together to create one objective function. We are concerned in this problem by the difference in magnitude between the first and second stage objectives. The maximum number of workers will be in the orders of hundreds, whereas delayed mail can be in the tens or hundreds of thousands.

To account for this, we will experiment with giving different weightings to the two objectives. We do this using the weighted-sum method from existing multi-objective optimisation literature. In this case, we define the parameter α to be the weighting we give the first-stage objective, with $0 \leq \alpha \leq 1$. We then let $1 - \alpha$ be the weighting for the second-stage objective.

With these, our objective becomes:

$$\alpha \sum_{j\in\mathcal{J}} g_j + (1-\alpha) \sum_{s\in\mathcal{S}} p_s \left(\sum_{a\in\mathcal{A}_{All}} \sum_{k\in\mathcal{K}} x_a^{ks} + \sum_{w\in\mathcal{W}_R} d_w^s \right).$$

4.3.4 Allowing changes to the set workplan

In Section 4.3.2 we proposed a two-stage model with recourse, in which the only second-stage decision was when to process mail, and how much mail to leave as delayed (rep-

resented by the x and d variables). The workplan (i.e. the y variables) were set in the first stage. This is a very restrictive interpretation of the mail centre. It may in fact be possible to make some small adjustments to the workplan on the day of the shift, in response to some unexpected mail volumes. Say for example, an equal number of workers have been assigned to work in WAs that process first class and second class mail. However, on a particular day, a surprisingly high volume of first class letters arrive to be sorted. In this situation, it may be beneficial to move some workers from the second class letters WA to the first class letters WA. This represents the shift manager making minor adjustments to the workplan in response to observing mail volumes. We say 'minor' as these changes would be made very soon before the shift, and hence it would be difficult to make large-scale changes at this stage. In this section, we outline changes to our model which allow for such changes to a workplan to be made in the second stage.

To allow changes to the workplan, we introduce a new variable:

- q^s_{wt}: the change to the number of processing units rostered in WA w for time t in scenario s.

These allow the workplan to change for various scenarios in case a specific WA observes either more or less mail than planned for. As they represent changes in the integer y variables, the q variables are integral as well. Since the change to the y variables can be positive or negative, the q variables are unbounded. However, the number of workers in WA w at time t must still be between 0 and C_{wt}. Therefore, we need to add the constraint:

$$0 \leq y_{wt} + q^s_{wt} \leq C_{wt},\ \forall w \in \mathcal{W}, t \in \mathcal{T}, s \in \mathcal{S}.$$

If the workplan is changed, then the node capacities in the network will change. We

change Constraint (4.3.5) to

$$\sum_{k\in\mathcal{K}}\sum_{a|(w,W_d(a))\in\mathcal{E}} x_a^{ks} \leq (y_{wt}+q_{wt}^s)u_w, \forall w\in\mathcal{W}, s\in\mathcal{S}.$$

to enforce this.

If we are reassigning workers from the current workplan, they need to already be on shift and working at that time period, according to the workplan. Therefore, we cannot reassign more workers than we have in the mail centre at that point. This is equivalent to saying any additional worker we add to one WA must be taken from another WA. We enforce this through the additional constraint

$$\sum_{w\in\mathcal{W}} c_w q_{wt}^s = 0,\ \forall t\in\mathcal{T}_j, \forall j\in\mathcal{J}, \forall s\in\mathcal{S}.$$

Considering the inconvenience of changing the workplan at the last minute, we want to penalise excess changes to the workplan. We do this by adding a penalty term in the objective function, based on these changes. We recognise that the q variables can take positive or negative values, so we cannot simply sum these and add them to Objective (4.3.1). Instead, we create additional auxiliary variables z with the following constraints:

$$0 \leq z_{wt}^s, \forall w\in\mathcal{W}, t\in\mathcal{T}, s\in\mathcal{S}, \tag{4.3.18}$$

$$q_{wt}^s \leq z_{wt}^s, \forall w\in\mathcal{W}, t\in\mathcal{T}, s\in\mathcal{S}. \tag{4.3.19}$$

Given we are trying to minimise the z variables, and that there are no other constraints on these variables, Constraints (4.3.18) and (4.3.19) mean that the z variables will take the value of the corresponding q variable if q is positive, or 0 if q is negative.

We want to add this to Objective (4.3.17). In order to do this, there are two considerations. Firstly, it is unclear how much we should penalise these second-stage

changes to the workplan. Therefore, we define the parameter κ to control the penalty. Secondly, we stated in Section 4.3.3 that we can run into problems with the different magnitudes of the different objectives. Since these second-stage changes will be in the same order as the first-stage objectives, we also will multiply this term by α in the objective. Therefore, we will add the following term to the Objective (4.3.17):

$$\alpha\kappa \sum_{s\in\mathcal{S}} p_s \sum_{w\in\mathcal{W},t\in\mathcal{T}} z^s_{wt}.$$

4.4 Results

To determine the benefits of using a stochastic program to account for the random mail volumes, we perform a series of experiments. We compared the stochastic programming models to the deterministic model presented in Chapter 3. This is to show the benefit of using a stochastic programming model over a deterministic model. We also noted the difference in magnitude between the first and second stage objectives, and hence varied the weights of the two objectives. Finally, we performed an experiment to determine the benefit of allowing changes to the workplan in the second stage of the model. This showed the benefit available to shift managers if such flexibility was allowed in the planning.

4.4.1 Data and setting

We obtained data from a UK-based mail company (UKMC). This data consisted of:

- Data on the mail centre structure/operations, including:
  - The mappings of the streams passing through various WAs.
  - The throughput of machines/staff at different WAs.

  - The capacities of different WAs.

  This data (which we previously used in Chapter 3) was used to build the network for the mail centre, given for each different day in one week (July 20-25, 2020). It consisted of 180 streams, in 60 different WAs. It was noted that there were substantial differences between the WA priorities and stream mappings on the different days.

- Data on the stream volumes on different days. This was given for every day in a 1-year period (March 30 2020 to March 27 2021). The mail volume data consisted of 312 days (one scenario for each day of mail volumes in the dataset, with no data for Sundays, for 52 weeks). We again noted significant differences for the different weekdays. These differences are shown via a boxplot of the mail volumes for each day of the week, given in Figure 4.1.1 in Section 4.1.

4.4.2 Experimental set-up

We used this data to create a time-expanded network for the mail centre. Given the differences in the network structure, WA priorities, and average mail volumes for the different weekdays, we model each day of the week separately in all experiments. This meant that we had six networks, one for each day of the week (excluding Sundays). Using this data, we constructed the network to model the mail centre, and created a scenario set. We modelled the network as having 48 30-minute time periods, split into three 8-hour (16 period) shifts. We chose 30 minute intervals (as opposed to the 10 minute intervals used in Chapter 3) as the stochastic programs are more computationally intensive than deterministic models. Hence, the time intervals were increased to reduce the number of variables and constraints in the model.

Once built, the network consisted of 2,654 - 2,820 nodes and 14,826 - 14,992 edges, depending on the day of the week. The slight differences are due to different mail arrival

Table 4.4.1: Nodes and edges for the networks for each day of the week

Day	Nodes	Edges
Mondays	2,800	14,972
Tuesdays	2,820	14,992
Wednesdays	2,818	14,990
Thursdays	2,814	14,986
Fridays	2,819	14,991
Saturdays	2,654	14,826

patterns and WA operating times between days. The exact nodes and edges for each day of the week are given in Table 4.4.1.

Due to the different patterns of demand on each weekday, we split the mail volume data into six distinct sets (of 52 days each), one for each day of the week (excluding Sundays). Each of these sets was then used as a scenario set with equally probable scenarios. That is, we have a separate stochastic program for each day of the week. We use here historical data directly only for the purposes of testing our approach. If this model were used in practice, we would use some sort of forecasting model which incorporates the latest available estimates we have for mail volumes. When solving these problems, we use the full set of scenarios for each day, or solve for a subset of randomly sampled scenarios.

Given we have mail volume data for an entire year, but mail centre network structure data for only one given week, we assume that the network structure for a given day of the week is constant throughout the period.

Each scenario represents one day of mail volumes arriving at a mail centre. We used the maximum number of workers (MMWT, as described in Section 4.3.3) and the delayed mail cost (DMC, described in Section 4.3.3) as our first and second stage objectives, respectively.

All models were fitted and solved using Gurobi optimiser, version 10.0.3. Python version 3.9.10 was used as the interface. Models were solved using only 1 core (to facilitate parallelisation of solving multiple problems at once), with a time limit of 120

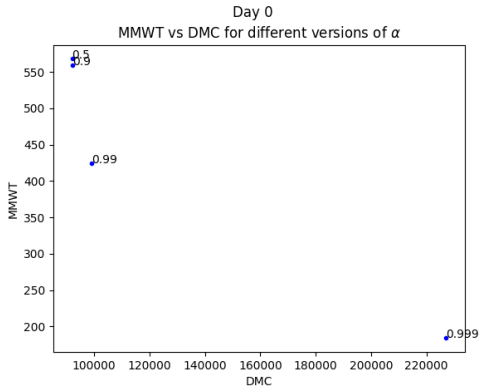
minutes. Models were solved on a high performance computing cluster. Optimality gaps for those models not solved within the time period are given in Appendix 4.A.

4.4.3 Experiments

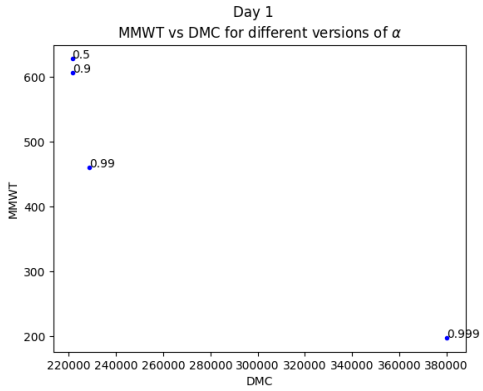
Balancing the first and second stage objectives

We first wanted look at the balance between the first and second stage objectives, using the weighted sum objective (4.3.17). We tested this for $\alpha = 0.5, 0.9, 0.99$, and 0.999. Recall that higher values of α give more weighting to the first stage objective (MMWT). We chose this range as we recognised that the model would be more likely to favour DMC rather than MMWT given the difference in magnitude between the number of workers and the potential number of delayed letters. For this experiment, we solved the stochastic program with all 52 scenarios to avoid any randomness associated with the random sampling of scenario sets. To examine the behaviour of the model for the different values of α, we plotted the first vs the second stage objectives. This is shown in Figure 4.4.1.

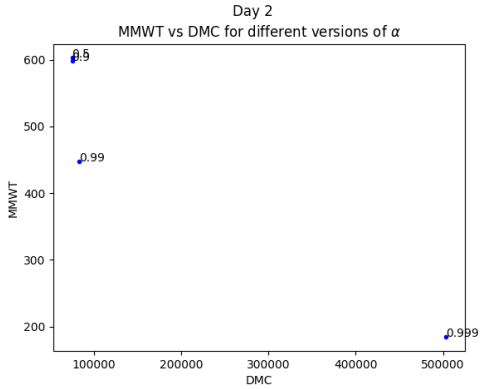
We see that Mondays, Tuesdays, Wednesdays, Thursdays and Fridays display a similar pattern. On these days, we can see that the weighting towards the first stage objective needs to be set very high before the model stops compromising on this objective. For $\alpha = 0.5$ & 0.9, we see that the values of DMC and MMWT are all relatively similar. The MMWT results are between 600-700 workers (across the 3 shifts). The DMC alters somewhat across the days of the week, but is similar for this range of α for each day. A change occurs when $\alpha = 0.99$. This caused the MMWT to drop to 400-450 workers (a roughly 25% drop), with a slight increase in the DMC (between 8 and 24%). When increasing α to 0.999, we see a very big increase in the DMC, increasing between 200 and 600%. The MMWT correspondingly falls to roughly 200 - a decrease of more than 60%. Most of the values of α seem to favour one objective over the other. $\alpha = 0.5$ & 0.9 favour delayed mail, whereas $\alpha = 0.999$ favours MMWT. In order to try



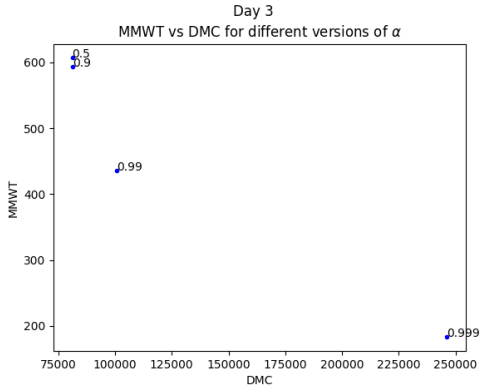
(a) Mondays.



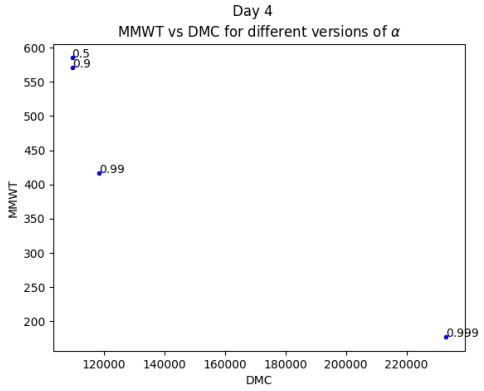
(b) Tuesdays.



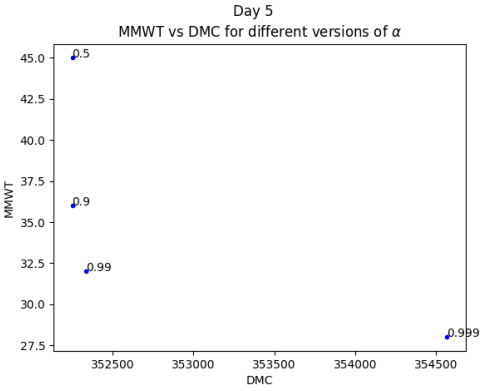
(c) Wednesdays



(d) Thursdays



(e) Fridays



(f) Saturdays

Figure 4.4.1: MMWT vs DMC for different values of α, split by different days of the week. Text next to data points indicates the value of α.

and balance the two, it seems that the best choice for α is 0.99.

The pattern for Saturdays (Figure 4.4.1f) is slightly different. The model for $\alpha = 0.999$ still heavily prioritises MMWT. However, the values for $\alpha = 0.5$, and 0.9 are more spread than the other weekdays, and closer to $\alpha = 0.99$. We again see that $\alpha = 0.99$ gives large reductions in the MMWT (11%-35%) for only small increases in the DMC values (approximately 6%). Therefore, $\alpha = 0.99$ again appears to ideally balance the two objectives.

EV vs Stochastic program

Having determined an appropriate balance between the two objectives, we wanted to determine the benefit of accounting for variation in the mail volumes compared to the deterministic models. We compared:

- The deterministic model from Chapter 3, using the expected value of each stream as the demand for that day, increased by m%. This is referred to as the EV_m solution, adapting the definition of the EV solution from Birge and Louveaux (2011).

- The stochastic programming model (4.3.1) using 3, 5, 10, 20, 25, and 30 scenarios. Given we are comparing against deterministic models (which are computationally much easier to solve than stochastic programs), we wanted to determine how the EV_m solutions performed against stochastic programs of varying sizes.

We compare the stochastic programming models to the EV solutions to see the benefits that can be gained from explicitly accounting for the stochasticity in the daily mail volumes. Alternatively, the deterministic model is much less computationally intensive. Therefore, if there is a deterministic model which performs similarly to the stochastic model, then computational time can be saved by using this model instead. We also do this for increasing values of m to determine if assuming a higher expected demand

improves the deterministic solutions. We choose $m = 0, 20, 40, 60, 80,$ and 100. Note that the EV_0 solution is simply the expected value (EV) solution commonly used in stochastic programming literature. We also test the values of $m = 40, 60, 80, 100$ to determine if there are better or worse values to use for the deterministic model.

For the stochastic programming models, we measured the performance of these solutions by looking at the stability of the out-of-sample (OOS) costs as defined in Sections 2.4.1 and 2.4.3 of Chapter 2. We calculate these OOS values with respect to the full scenario sets consisting of the 52 scenarios described above. This gave us an indication of the quality and stability of estimations to the recourse function for the stochastic programs. To gauge the OOS stability, we solved each stochastic program 20 times, each time with a different randomly-sampled subset of the full scenario set. This gave us an indication of the size and variability in the OOS costs for stochastic programs of various sizes. When looking at the OOS costs, a lower median cost and a narrow spread of values across the runs indicates a better-performing model.

For the EV_m methods, we also calculate the OOS value for these solutions. We note that in this case, the OOS cost for this solution is the EEV, from the stochastic programming literature. We use this as a benchmark to compare to the OOS recourse averages for the stochastic programming models. We refer to the expectation of the EV_m solution as the EEV_m value. We also refer to the EV solution (the deterministic model with no assumed increase in the demands) as the EV_0 solution (with corresponding EEV_0 value) for consistency. Note that unlike the OOS for the stochastic programs, there is no variation in the EEV_m values.

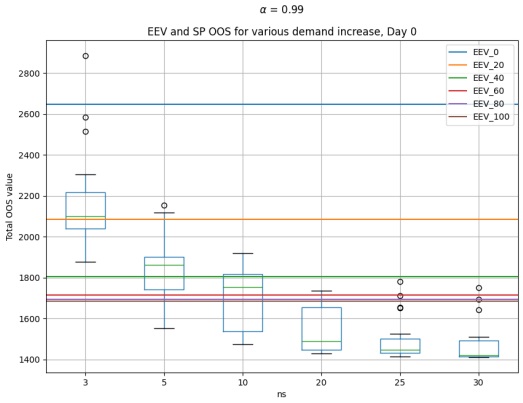
In this experiment, we set $\alpha = 0.99$, following our observations from Section 4.4.3. Recall that the basic model (4.3.1)-(4.3.16) assumes no changes can be made to the workplan in the second stage of the model.

Figure 4.4.2 shows the EEV_m results compared to the average OOS costs for the stochastic program solutions for different numbers of scenarios. Each subplot shows a

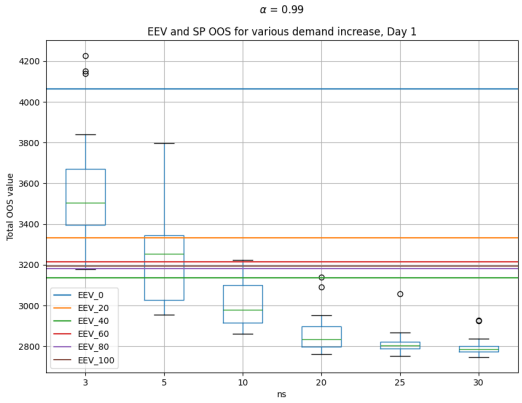
different day of the week.

Initially, we see that the problem is quite unstable for small scenario set sizes (3 and 5 scenarios), with similar OOS costs to the EV_0 and EV_{20} solutions. However, as the scenario set size increases, the stability and median OOS costs improve. Furthermore, if enough scenarios are chosen, the stochastic programs always achieve lower OOS results than all the EV models, with the comparative benefits of the stochastic programs increasing as the scenario set size increases. The stochastic programs generally perform better than all EV models when 20 or more scenarios are used, regardless of the assumed demand. The stochastic programs start to converge at 20 scenarios, showing that less benefit is gained from using more scenarios than this. Furthermore, we see that the EEV_0 value is worse than the median OOS costs for all stochastic programs for all days, with only a small number of outlying OOS runs performing worse than this across all days.

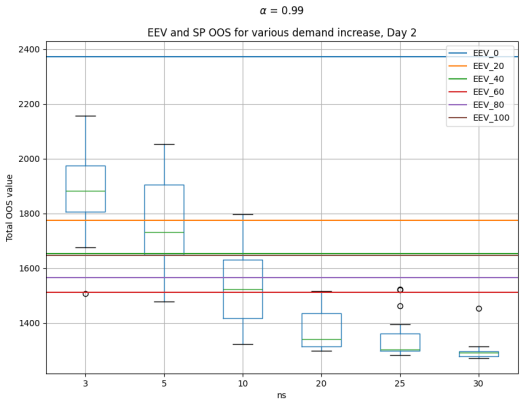
Looking at the EEV_m values for 20, 40, 60, 80, and 100% increases, we see that increasing m does not necessarily result in lower EEV values. That is, we see a trade-off between planning for more mail and incurring higher overall costs. We note that the value of m which minimises EEV_m is different for the different days. The lowest value is EEV_{100} for Mondays (although the EEV_{60} and EEV_{80} values are very similar). However, EEV_{40}, EEV_{60}, and EEV_{80} are the best performing for Tuesdays and Thursdays, Wednesdays and Fridays, and Saturdays respectively. While there is some variation among the days, the stochastic programs with 10 scenarios show either similar or lower median OOS costs than the best EV solutions (only Mondays and Wednesdays have worse OOS scores for 10 scenarios). The majority of OOS runs for 20 scenarios out-perform the best EV solutions for all weekdays.



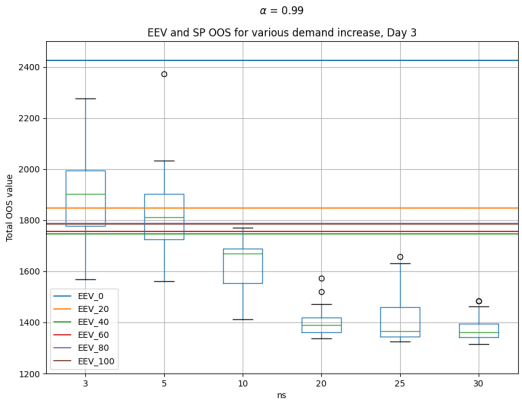
(a) Monday.



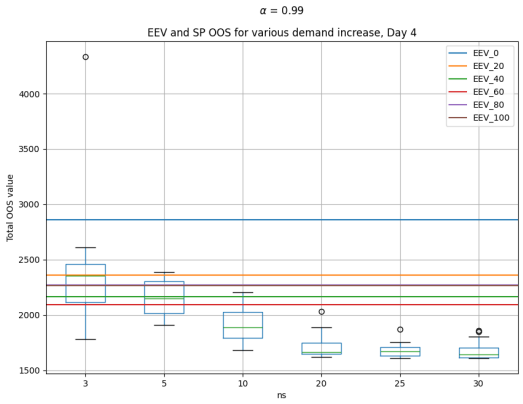
(b) Tuesday.



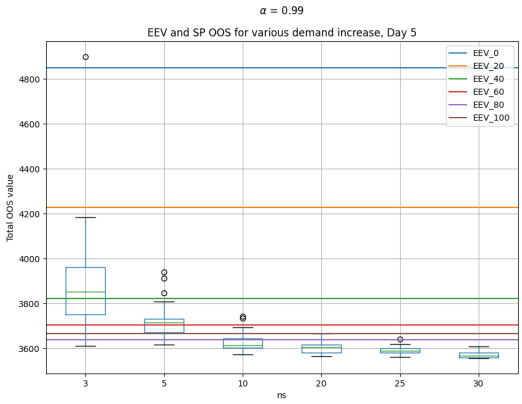
(c) Wednesday.



(d) Thursday.



(e) Friday.



(f) Saturday.

Figure 4.4.2: *EEV* for the *EV* solutions and OOS results for stochastic programs with increasing scenario set sizes for $\alpha = 0.99$, for different days of the week.

Allowing changes to workplans

Having established the benefits of the stochastic model over the deterministic, and determined an appropriate weighting between the first and second stage objectives, we now add more flexibility in the second stage decisions. Previously, the only second stage decision was where to route the mail to minimise delays. Now, we investigate allowing changes to the workplan in the second stage, as outlined in Section 4.3.4.

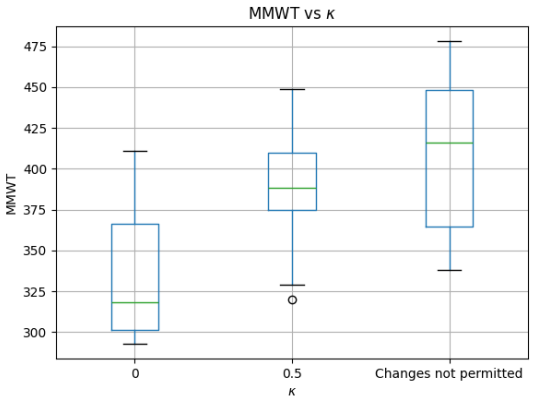
We compare no permitted changes (our previous models) with changes permitted with $\kappa = 0$ and $\kappa = 0.5$. $\kappa = 0$ indicates a model with free changes, and $\kappa = 0.5$ indicates that a worker can be moved for half of the cost of them being assigned in the original workplan. From our previous experiments, we chose to solve the models using 20 scenarios, due to the increased difficulty for solving for all 52 scenarios, and as Figure 4.4.2 showed that the stochastic programs generally started to converge for this many scenarios. As in the previous experiment, we set $\alpha = 0.99$, based on our observations from Section 4.4.3. We again calculate the OOS costs from 20 trials.

To see how these changes affect the model, we look at the MMWT, DMC, and changes penalty OOS costs separately. These are shown in Figures 4.4.3, 4.4.4, and 4.4.5 respectively.

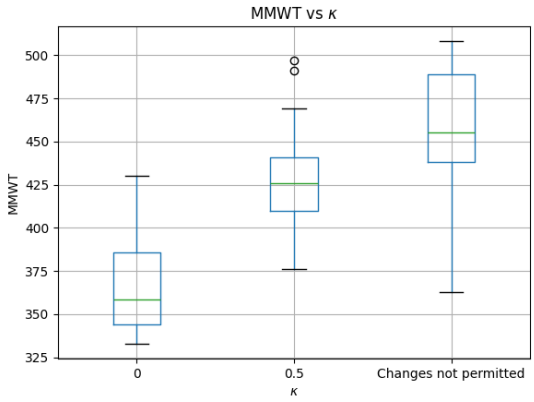
Figure 4.4.3 shows that for all days of the week, $\kappa = 0$ results in lower MMWT values than $\kappa = 0.5$, which in turn has a lower MMWT values than the model with no changes. This makes sense. If more flexibility is allowed in the second stage (or if it is penalised less) then the model can be more aggressive in the first stage - assigning fewer workers that can be moved around in the second stage.

Looking at the OOS delayed mail costs, we see consistent behaviour across the days (Figure 4.4.4). Across all days, the models with changes permitted show similar OOS costs for both levels of κ, with the no-changes model showing higher costs, again highlighting the benefit of additional model flexibility.

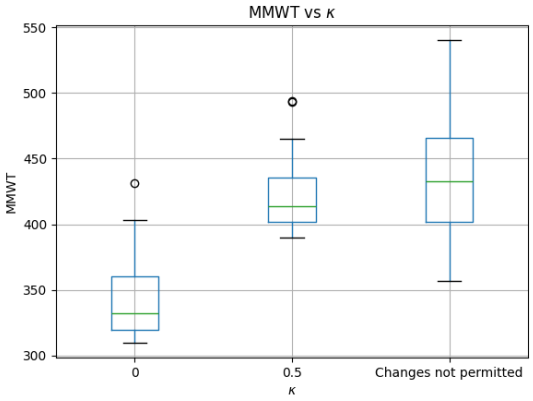
We see that $\kappa = 0$ results in much lower delayed mail costs than $\kappa = 0.5$. However,



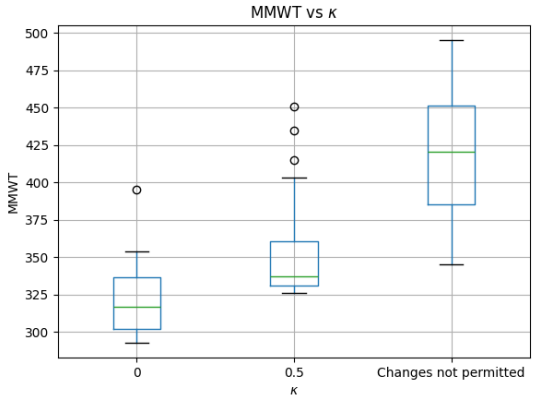
(a) Monday.



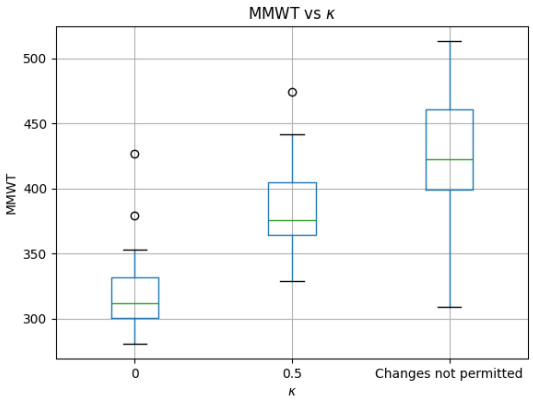
(b) Tuesday.



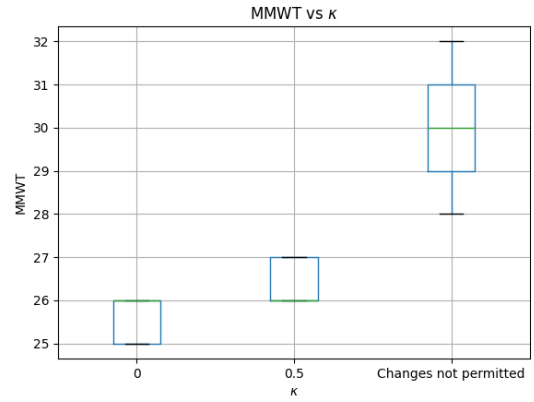
(c) Wednesday.



(d) Thursday.

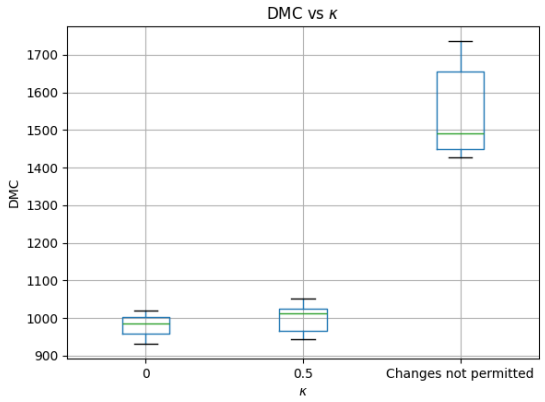


(e) Friday.

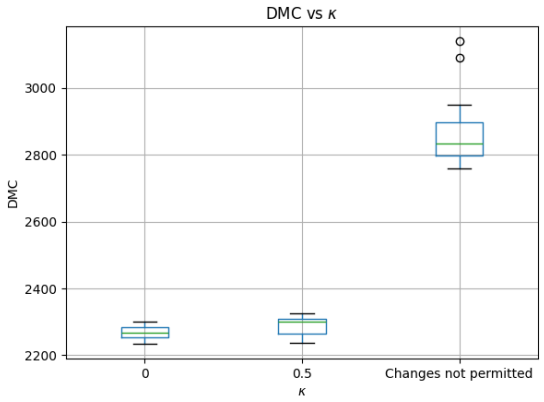


(f) Saturday.

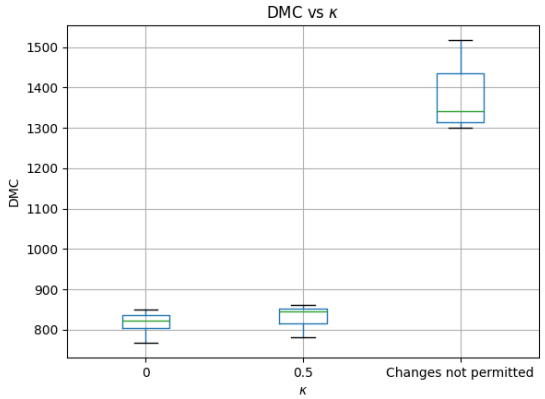
Figure 4.4.3: First stage objective results for the model with changes to the workplan allowed in the second stage $\kappa = 0$, $\kappa = 0.5$, and the original model with no changes allowed, split by weekday.



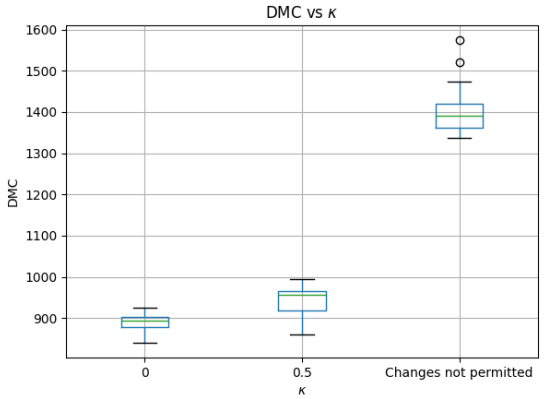
(a) Monday.



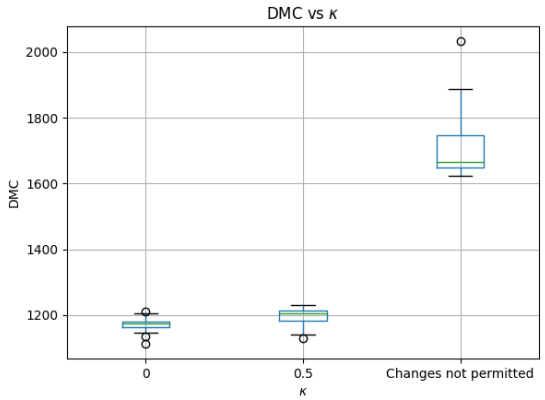
(b) Tuesday.



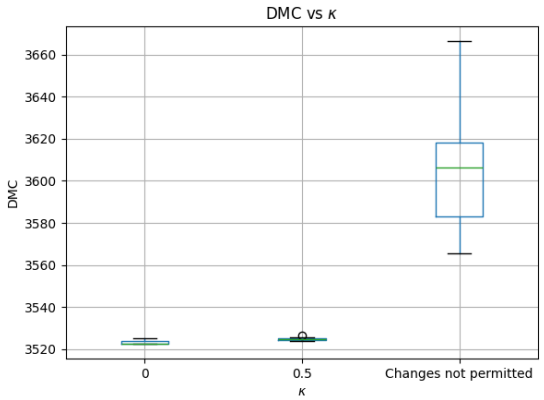
(c) Wednesday.



(d) Thursday.



(e) Friday.



(f) Saturday.

Figure 4.4.4: OOS delayed mail costs for the model with changes to the workplan allowed in the second stage for $\kappa = 0$, $\kappa = 0.5$, and the original model with no changes allowed, split by weekday.

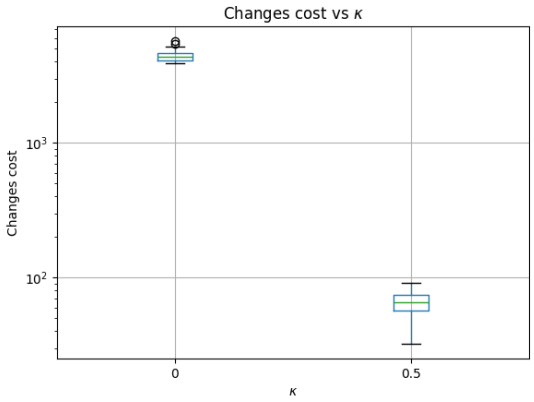
it is unclear how much the workplan has to change in the second stage to achieve this. We examine this by comparing the workplan changes made under the two values of κ, shown in Figure 4.4.5.

We see that there are orders of magnitude of difference between the changes for the two different penalties. This pattern is the same for all days of the week. We especially note the extremely large number of changes when $\kappa = 0$ (recall that one change is one worker changing their assigned WA for one time period). We note that this is in the order of thousands of changes (typically 4,000-5,000), and the order of the number of workers is in the hundreds (300-400). This equates to roughly 80 workers (20-25% of all staff) being changed each time period. This is obviously impractical to implement in the mail centre given the short time frame in which these changes can be done. By contrast, for $\kappa = 0.5$, we see between 60-80 changes per scenario. This equates to 1-2 changes per time period, which is much more feasible.

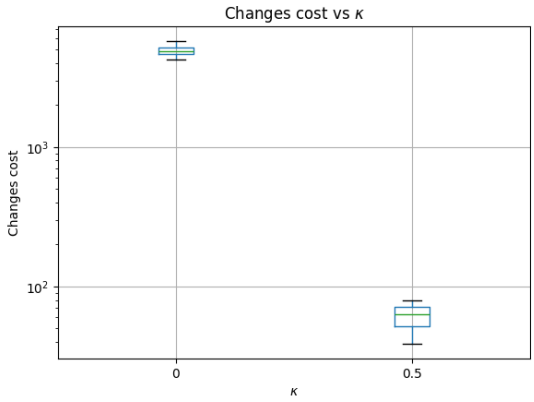
We also note that while there is an order of magnitude in increase in the number of changes, we do not gain a similarly proportional decrease in the maximum workers (Figure 4.4.3). This also suggests that $\kappa = 0$ results in impractical workplans and decisions.

4.5 Conclusions

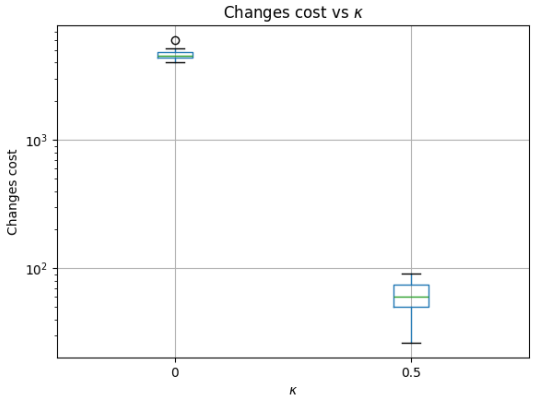
While there is a body of work for staffing mail centres assuming known mail volumes, there has been little work done for random mail volumes. We address this by extending the model developed in Chapter 3 to a stochastic programming model. We examined different weights between the first and second stage objectives, to counteract the difference in magnitude between them. Upon establishing an appropriate weighting, we compared the behaviour of this model to the behaviour of the deterministic model assuming the expected values of the mail volumes, as well as various increases of the mail



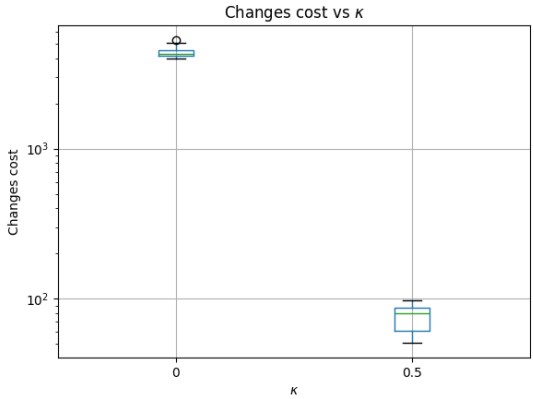
(a) Monday.



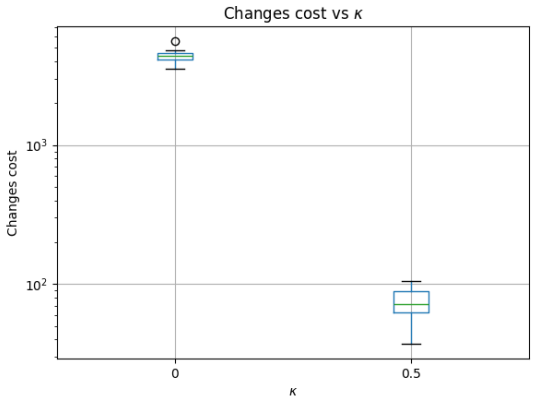
(b) Tuesday.



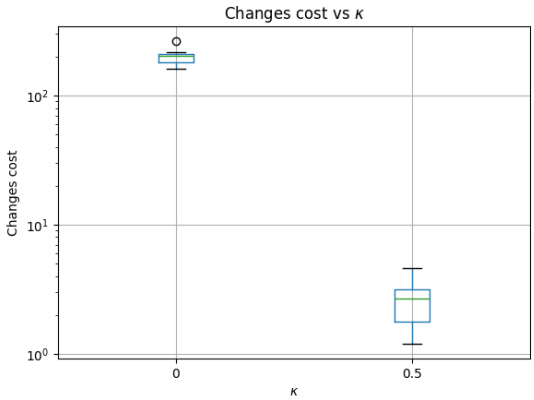
(c) Wednesday.



(d) Thursday.



(e) Friday.



(f) Saturday.

Figure 4.4.5: Average second stage changes per scenario to the workplan for the model with changes to the workplan allowed in the second stage for $\kappa = 0$, $\kappa = 0.5$, split by weekday. Note that these have been plotted on a log scale.

volumes. Finally, we allowed additional flexibility in the second stage of the model, and examined what effect this had on staffing and delayed mail costs.

When examining the different weightings between the objectives, we found that for values of α less than 0.99, the results were similar. At $\alpha = 0.99$ we started to see a tradeoff between the two objectives, with higher α values tipping the model towards favouring the first stage. We then saw this balance when we repeated the EV calculations for $\alpha = 0.99$. In this instance, we now saw that there was a trade-off, in that the best increase differed among the days of the week, and in the majority of cases was lower than 100%.

After applying this value of α to the stochastic programs, we compared stochastic programs of different sizes to deterministic models with increasing expected demand. We saw that the stochastic programming models result in lower total costs than the EV_0 model. We found that this was true for stochastic programs with as few as three scenarios in the scenario set, with the benefits increasing as the sample size increased. When compared to the EV_{20} model, we saw the stochastic program with 3 scenarios performed similarly, but those with 5 or more scenarios outperformed this. We also saw that increasing m improved the overall performance of the deterministic models. This resulted in the best EV_m models performing similarly to the stochastic program with 10 scenarios. However, with 20 or more scenarios, the stochastic program outperformed all EV models.

The final experiment showed the benefits we get from allowing changes in the second stage of the model. We found that changes to the workplan at any penalty resulted in lower delayed mail costs for all days, and lower staffing costs for all days except Saturdays. In this instance, having no changes on the penalty ($\kappa = 0$) out-performed penalising the changes ($\kappa = 0.5$). However, the number of second-stage changes made to the work plan is much higher with no penalty, and would be impractical to implement in practise. Therefore, allowing changes with $\kappa = 0.5$ gave more appropriate results.

Despite the clear improvements in the models, we note some limitations with the work. When the objectives are appropriately weighted, we need larger scenario sets (more than 10 scenarios) to see the benefit over the deterministic models. However, we recall that our scenario sets were chosen by random sampling. As discussed in Section 2.4 of Chapter 2, there are many better ways to select scenario sets. We develop a new framework for this in Chapter 5, but the simple application of an established scenario set method could also prove beneficial. Furthermore, we have only assumed one possible method of change for the adjustments to the workplan in the second stage. Other options could include being able to bring in additional workers at a higher cost (similar to overtime workers).

This work contributes to the literature by showing that mail centre staffing can be improved by accounting for randomness in mail volumes. It also advocates for allowing flexibility after setting the staff levels to respond to observed mail volumes.

4.A Convergence and optimality gaps

Here we give the optimality gaps for the fitted models. Each gap is expressed as the proportion difference between the best solution found and the best bound found.

Table 4.A.1 shows the optimality gap for the models fitted for the different values α split by weekday. We see the gaps are less than 1% for all weekdays when $\alpha = 0.5$, and 0.9. It is also less than 1% for $\alpha = 0.99$ on Mondays, Tuesdays, Fridays, and Saturdays, and is 1.09% for Wednesdays.

We see larger gaps for $\alpha = 0.99$ on Thursdays, and for $\alpha = 0.999$ on all days except Saturdays. For $\alpha = 0.999$ the largest gap we see is for Thurdays, with a gap of 12.75%. The gaps for $\alpha = 0.999$ on other days ranged from 4.5% (Fridays) to 7.8% (Wednesdays). While concerning, we note that for this value of α (0.999) we see that the DMC is much larger than all other values (1.7-5 times larger) and the MMWT is

much smaller than the other values (2 - 3.5 times smaller). Therefore even if we ran the model for longer, and it were to converge, and even if the convergence occurred at the best bound, the model with this value of α would still be in a similar position on the curve relative to the other values of α. Given the gaps and relative differences in the objectives for $\alpha = 0.999$, we conclude that the lack of convergence will not affect the relative position of this model compared to the other values of α. By the same logic, we realise that the model for $\alpha = 0.99$ for Thursdays will also remain in roughly the same relative position, and as such will remain the ideal compromise point.

Table 4.A.1: Optimality gaps for the stochastic programs for different values of α, split by day of the week.

(a) Mondays

α	Optimality Gap
0.999	0.0681
0.99	0.0062
0.9	0.0005
0.5	0.0001

(b) Tuesdays

α	Optimality Gap
0.999	0.0467
0.99	0.0009
0.9	0.0002
0.5	0.0001

(c) Wednesdays

α	Optimality Gap
0.999	0.0781
0.99	0.0109
0.9	0.0005
0.5	0.0002

(d) Thursdays

α	Optimality Gap
0.999	0.1275
0.99	0.0940
0.9	0.0003
0.5	0.0001

(e) Fridays

α	Optimality Gap
0.999	0.0453
0.99	0.0015
0.9	0.0003
0.5	0.0001

(f) Saturdays

α	Optimality Gap
0.999	0.0001
0.99	0.0001
0.9	0.0001
0.5	0.0000

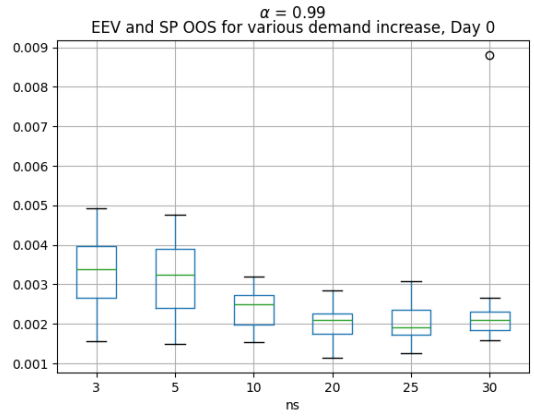
We see larger optimality gaps for the models allowing changes to the workplan in the second stage (Figure 4.A.2). While often less than 3%, these can reach 6%, and we note much larger problems with convergence on Thursdays (with gaps of 10-20%). This is concerning. However, we note that

- The pattern of results was similar for all days, regardless of the optimality gaps.

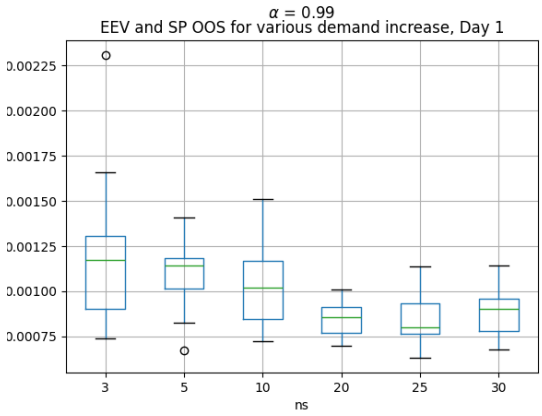
- The optimality gaps were generally larger for the model with $\kappa = 0.5$. This meant that these results could improve if the model was left to run for longer. However, the difference in the gap size was smaller than the proportional difference in the objectives. This meant that even if this model convergence, it would still result in higher values than for $\kappa = 0$.

- We acknowledge that $\kappa = 0$ resulted in an infeasible number of changes, and would never be implemented in practise. Therefore, we are more concerned as to how $\kappa = 0.5$ performed compared to the model with no changes. This model performed worse than the model with $\kappa = 0.5$, and with smaller optimality gaps. Therefore, even if we did achieve convergence, the benefit of using the model with $\kappa = 0.5$ would only increase.

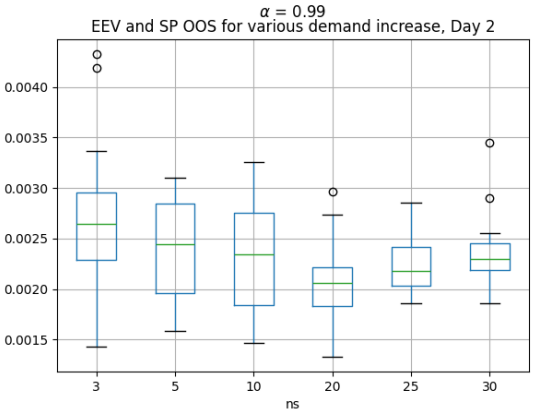
For Thursdays, we acknowledge the very large gaps. However, we see that the results for Thursdays were similar to the other days of the week, suggesting that the overall pattern of results would be the same if this were to converge.



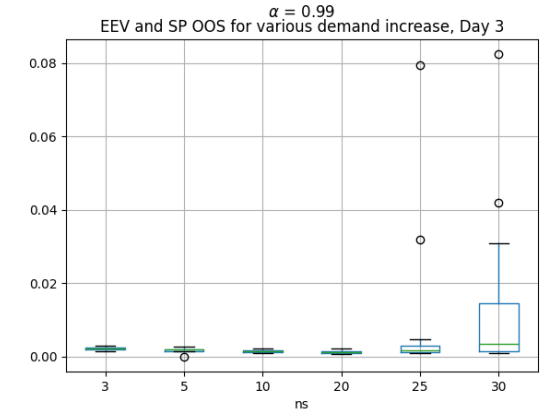
(a) Monday.



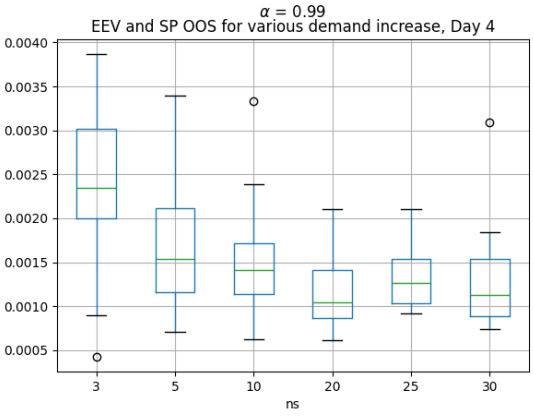
(b) Tuesday.



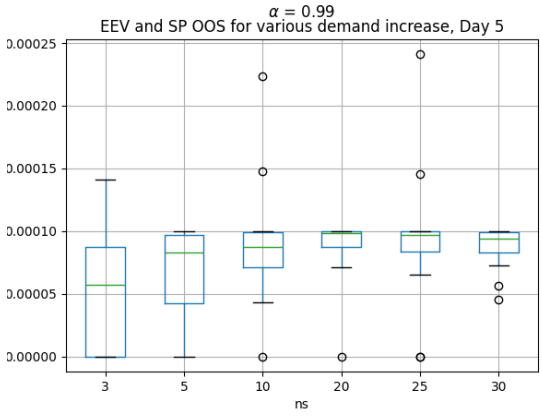
(c) Wednesday.



(d) Thursday.

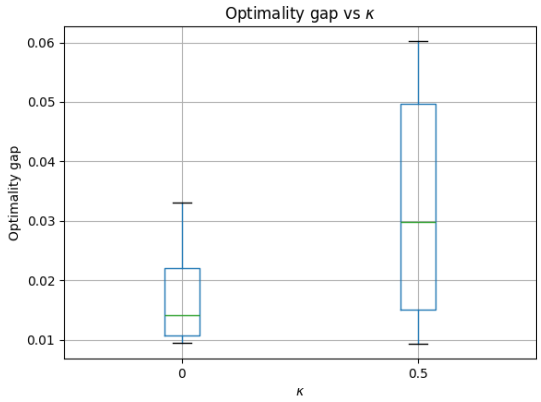


(e) Friday.

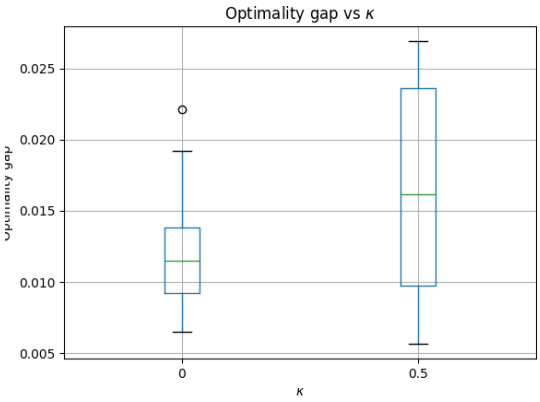


(f) Saturday.

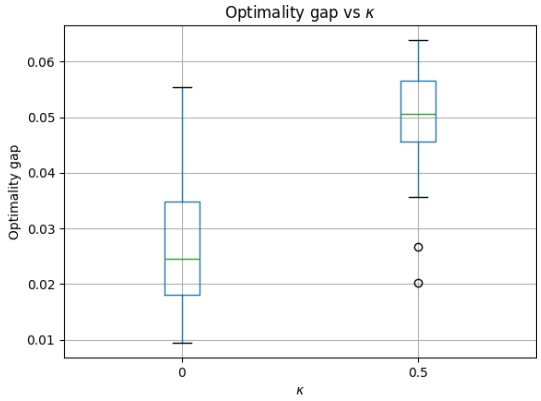
Figure 4.A.1: Optimality gaps for calculating the first stage decisions for the stochastic programs with $\alpha = 0.99$, split by weekday.



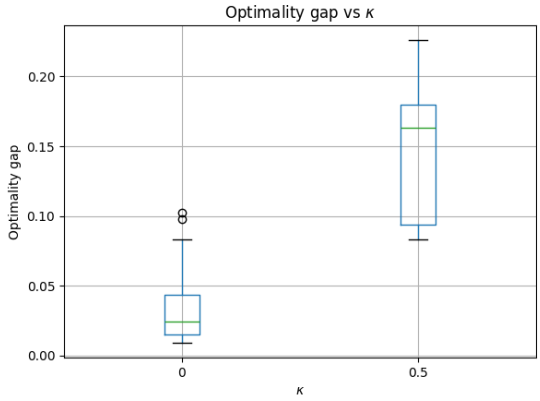
(a) Monday.



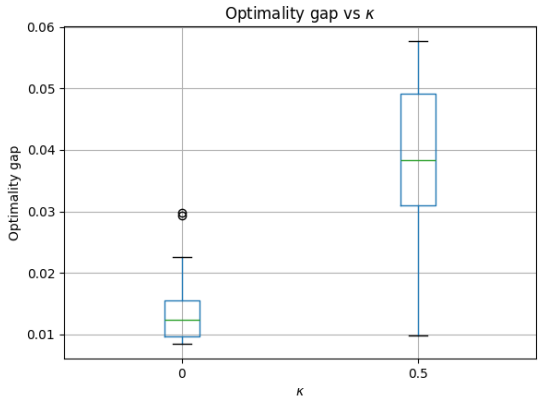
(b) Tuesday.



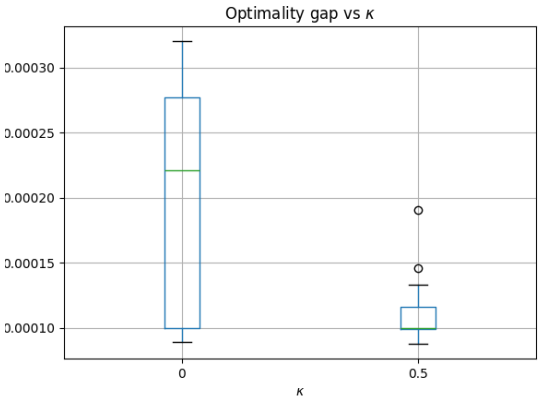
(c) Wednesday.



(d) Thursday.



(e) Friday.



(f) Saturday.

Figure 4.A.2: Optimality gaps for calculating the first stage decisions the model with changes to the workplan allowed in the second stage for no penalty, and a penalty of 0.5, split by weekday.

Chapter 5

General problem-based scenario reduction

5.1 Introduction

Decisions frequently need to be made before all information is known. In portfolio selection, capital needs to be allocated to different investments without knowing the exact returns. In facility location, facilities need to be constructed and opened despite uncertain customer locations and demands. In power generation, contracts for energy supply need to be bid on before demand is known. In all of these cases, we can use stochastic programming to determine the optimal first stage decisions, accounting for this uncertainty.

Two-stage stochastic programming with recourse was first proposed by Dantzig (1955). Under this paradigm, the decision maker has a number of *first-stage* decisions that need to be made before a random process is realised. The decision maker then makes a number of *second stage decisions*, also called *recourse actions*, in response to this, which have associated costs. For example, in a facility location problem, the first-stage decisions are where to open new facilities. Once opened, the customer locations

and demands are observed. The second stage decisions/recourse actions may then be which facilities to serve which customers from, with different costs for the customers, depending on which facility they are served.

Solving a stochastic program involves minimising the cost of the first stage decisions, and the expected cost of the second stage decisions given the first stage. The expectation of the second stage costs is often intractable if the true distribution of the uncertain variables is either very large or continuous. Therefore, the true distribution is often approximated with a discrete distribution $\mathcal{S}$.

The problem in stochastic programming comes with the selection of the scenario set $\mathcal{S}$. This is referred to as *scenario set selection*. This broadly falls into two categories. We can start with an empty scenario set, and populate it by generating new scenarios, with some *scenario generation* process. For example, sampling from a distribution, or constructing a scenario set with desired statistical properties (Høyland et al., 2003). Alternatively, we can start with a large (discrete) scenario set, and select a more tractable subset, with a *scenario reduction* process. This can be a more appropriate approach when the starting distribution is discrete, or is an empirical distribution of observed outcomes. This approach is the focus of this chapter.

There is a balance involved with scenario set selection. As the size of $\mathcal{S}$ increases, the uncertainty is represented more accurately. However, as $\mathcal{S}$ increases, the size of the stochastic program increases, and can quickly become intractable. Therefore, the challenge is to pick $\mathcal{S}$ such that the uncertainty is accurately represented, but without $\mathcal{S}$ growing too large.

Different methods have been developed to select scenario sets. Traditionally, these methods have been *distribution-based*. That is, the scenario set is selected using information about the true distribution of the problem, without consideration of the problem. For example, sampling from a given distribution, or minimising the Wasserstein distance between the true distribution and a target distribution (Dupacova et al.,

2003).

More recent approaches incorporate information about the specific problem into the scenario set selection method. These methods are referred to as *problem-based*. The rationale for this is that by incorporating problem information, the methods can create more concise scenario sets. For example Fairbrother et al. (2022) consider the problem of minimising a tail risk measure in a stochastic program. The authors identify an area of the distribution called the risk region, which determines the value of the tail risk measure. They generate scenario sets in which nearly all scenarios are within this region.

Unfortunately, the key feature of problem-based methods are often also their drawback. The methods can often not be applied to a wide range of problems, or need careful tailoring to be used in different contexts. For example, the method proposed by Fairbrother et al. (2022) can only be used where the risk region can be identified for the problem. However, there have been recent advancements which have more general requirements. Firstly, the ability to solve the problem with single or very few scenarios. Secondly, the ability to evaluate solutions out of sample. Essentially, these methods work by finding possible solutions to the problem, and testing how these solutions perform for different scenarios in the set. The information from this testing is then used in the scenario set selection procedure. Examples include Hewitt et al. (2022); Keutchayan et al. (2023); Zhang et al. (2023); Narum et al. (2024).

In this chapter, we use the similarities between these recent papers to develop a framework for problem-based scenario reduction incorporating ideas from the recent literature cited above. The framework consists of creating a pool of candidate solutions, calculating the recourse values for each scenario and each candidate solution, and using this information in existing scenario reduction techniques. This framework treats these three steps independently, allowing users to select different approaches for each step of the problem. This framework includes several existing methods, and suggests different

approaches by combining different steps in new ways.

This chapter makes two main contributions. The first is the proposed framework for general problem-based scenario reduction describe above. The second is that we perform computational experiments, applying this framework to a number of problems, transformations, and scenario reduction methods. This demonstrates how the framework can be generalised. Furthermore, we find that creating the candidate solutions by solving smaller stochastic programs generally (but not always) performs better than other candidate solutions. However, we did not find one scenario reduction method that consistently outperformed others.

The structure of the paper is as follows. In Section 5.2, we review the relevant literature. Section 5.3 gives a motivating example, and outlines our proposed framework. We present the results from our computational experiments in Section 5.4, and our conclusions are discussed in Section 5.5.

5.2 Literature review

Initial scenario set selection procedures were generally distribution-based. The simplest of these is Monte Carlo sampling, where scenarios are independently sampled from some distribution (either continuous or a larger existing discrete set). This has been developed with more sophisticated sampling techniques, including antithetic and Latin-hypercube sampling (Higle, 1998), which were used to reduce the variance in sampled sets. Other approaches instead construct a scenario set with specific statistical properties. For example, Høyland et al. (2003) develop a scenario generation method which creates a scenario set matching pre-set moments and correlations. Kaut and Wallace (2011) proposed another generation method which can match pre-described copulas. This gives the user more control over the dependency structure. Further distribution-based methods choose reduced sets that are 'closest' to some original set or distribution,

e.g. Pflug (2001) and Dupacova et al. (2003) ('Closest' in this context means the set with the smallest Wasserstein distance from the reference set).

As the field has developed, several authors have shown that gains can be made from incorporating problem-specific information into the scenario selection process. The early pioneers of this were Dantzig and Glynn (1990) and Infanger (1992), who used importance sampling when drawing their random samples. The idea of importance sampling is that the probabilities with which observations are drawn are deliberately biased to favour certain regions of the distribution (usually regions of low probability) more than the original distribution. The samples are later de-biased when taking the estimator. However, their sampling procedures require that problems are of a specific form.

Feng and Ryan (2016) extended the work of Dupacova et al. (2003) to include problem information. Whereas Dupacova et al. (2003) used the Wasserstein distance as a metric to select scenarios, Feng and Ryan (2016) develop sensitivity indices, which take given first and second stage solutions, and return a value providing some information about the problem. These are then used to cluster the scenarios. The sensitivity indices are functions chosen specific to the problem to give information which will be relevant. A similar method was developed by Sun et al. (2018), who recognised that for their problem (a power generation problem) a new variable (total power flow) could be found by solving single-scenario deterministic models, which was beneficial in their clustering procedure for scenario reduction. Again though, both these methods create new information which is only relevant to the specific problems they are solving. Feng and Ryan (2016) note that 'useful [sensitivity] indices are problem-specific', as demonstrated by the indices they choose. Sun et al. (2018) find that the scenario-specific problems they solve have a problem-specific interpretation, which is why these can be used for their problem.

A different approach was taken by Prochazka and Wallace (2020), who created a

pool of solutions (via problem-specific heuristics) to act as a proxy for the feasible region. They then chose a scenario set that would minimise the distance of the average recourse for the reduced set vs the full set, over all solutions in the pool. However, they chose this set by developing a loss function to minimise. This loss function is generally intractable for most problems, and requires tailored heuristics, limiting the applicability of the method. Indeed Prochazka and Wallace (2020) note that a drawback of their method is its specificity to their problem. This makes empirical comparisons between their method and others difficult.

We take particular interest in a number of new approaches without this drawback. These approaches take pre-existing methods, such as clustering (Narum, 2020; Hewitt et al., 2022; Bertsimas and Mundru, 2023; Keutchayan et al., 2023), Singular Value Decomposition (SVD) (Narum et al., 2024), and property-matching (Prochazka and Wallace, 2020; Zhang et al., 2023). They then apply these methods not to the scenarios themselves, but to the scenarios after 'transforming' them to a new space. Whilst there are some variations in the approaches, they all share a common element. Namely, they all get additional information about the problem by using 'candidate' first stage solutions and finding the second-stage/recourse costs for each combination of scenario in the full set and candidate solution. This can be thought of as transforming the scenarios to a different space, where different dimensions in the space measure the performance of different candidate solutions on that scenario. We note that the requirements of a problem to be used with these methods are quite nonrestrictive. Specifically, the only requirements are that the problem needs quick methods of finding appropriate candidate solutions, and that the methods are compatible with testing these solutions with specific scenarios.

An important difference in these methods comes from how the candidate solutions are selected. Hewitt et al. (2022) and Bertsimas and Mundru (2023) create candidate solutions by solving a deterministic program for each scenario in the full scenario set.

They then test each solution with each scenario, and measure how much worse that solution/scenario combination is than using the candidate solution for the given scenario. A way to think about this is that they are calculating the *opportunity cost* of planning for scenario a if scenario b is actually observed. After performing this 'transformation', the information is used in different ways. Bertsimas and Mundru (2023) replace the L_2-norm in the objective in an optimal transport problem (thus altering the Wasserstein distance metric) and select scenarios based on this. They show that solving this problem is equivalent to performing k-means clustering on the problem, using the opportunity cost as the distance. We note there that some of their theoretical results make assumptions that cannot necessarily be generalised. For example, they assume scenarios have unique optimal deterministic solutions. However, the problem-based distance metric itself is generalisable. Hewitt et al. (2022) use the opportunity cost as the distance between scenarios, and then perform spectral clustering for reduction instead, whereas Keutchayan et al. (2023) make a transformation to a similar space to perform a method similar to k-medoids.

Other approaches use candidate solutions without creating distance metrics from them. Zhang et al. (2023) also employ single-scenario candidates. They choose the scenario set (of desired size) which minimises the difference in OOS recourse between the chosen set and the full set for the candidate solutions. This is equivalent to a sparse regression problem, which is solved with a heuristic. They also reduce the set of candidates by solving a solution pooling problem (they show this reduces to a facility location problem). Another approach is taken by Narum et al. (2024). They first create the output distribution by evaluating how the scenarios perform when applied to a number of candidate solutions. This is similar to Hewitt et al. (2022) and Keutchayan et al. (2023). However, Narum et al. (2024) create their candidate solutions using small scenario sets (each a subset of the full set) rather than a single scenario, and solving the stochastic program over each of these sets. They then use SVD to decompose the output

distributions of the stochastic program, and choose a new scenario set which preserves the expectations of the most important singular vectors for the output distribution. They show this method to be superior to multiple distribution-based methods for solving a number of different problems.

We note that the general idea behind a number of the papers above is to transform the scenarios to some problem-based space, and then apply a scenario reduction method to select the scenario set. However, the work in previous papers seem to treat these two steps as both embedded within a larger scenario set selection approach. Our proposed framework de-couples these two steps, giving the user the ability to choose the transformation and the method which suits the problem.

Another aspect that has not been addressed by existing literature is a numerical comparison of using the different spaces to see how they perform. Examples of this exist for distribution-based methods (Higle, 1998; Linderoth et al., 2006), but not yet for problem-based. This is important, as (in our proposed framework) the creation of the different spaces is independent of actually performing the scenario reduction method. This would give insight into how the different spaces perform in different settings, or if one space seems to perform better in general than the others. We also note the value of comparing different scenario reduction methods (e.g. clustering, SVD) combined with different spaces. We therefore conduct a numerical comparison of the commonly used transformations in the literature.

Our two main contributions are to propose a framework for generalised problem-based scenario reduction, and to use this framework to perform a numerical comparison of existing methods and transformations. The proposed framework will generalise a number of existing problem-based scenario reduction methods, and allow for the use of different transformations and reduction techniques to be combined. The numerical study will shed light on how these different spaces compare to one another.

Table 5.3.1: Scenarios for the mail centre example

	Letters	Parcels
Scenario 1	0	0
Scenario 2	0	500
Scenario 3	0	700
Scenario 4	1,000	200
Scenario 5	1,000	300
Scenario 6	1,000	1,000

5.3 Motivation and proposed framework

5.3.1 Motivating example

Literature has shown the advantage of problem-based scenario reduction. Specifically, multiple authors create problem-based information by evaluating the recourse when scenarios are applied to given candidate solutions. These recourse costs are then used with reduction techniques, such as clustering (Hewitt et al., 2022), dimensionality reduction (Narum et al., 2024) or property matching (Zhang et al., 2023). The rationale behind these approaches is that when reducing the large scenario set, we want to treat scenarios similarly (e.g. group them in the same cluster) if they perform similarly on given solutions.

To see the advantage of this approach, we consider a small example of a mail centre over one time period. Suppose our small mail centre consists of only two streams - letters and parcels - with a dedicated work area (WA) for each. We assume that one worker in the letters WA can sort 100 letters per time period, whereas, one worker in the parcels WA can sort 20 parcels per time period. The cost for failing to sort either letters or parcels is 1 unit per 50 unsorted letters (or part thereof), or 4 units per 50 unsorted parcels (or part thereof). Our objective is to minimise the total cost of unsorted items plus the total number of workers allocated to sort the items.

Consider that we have a scenario set of possible mail volumes shown in Table 5.3.1. We visualise this in the two-dimensional space shown in Figure 5.3.1.

Suppose we want to reduce this set to 3 scenarios. One method we could use to do this is k-medoids clustering. In this method, a 'medoid' centre is found for each cluster, and the scenarios are grouped based on which medoid they are closest too. The 'medoid' for each cluster is defined as the scenario within the cluster with the lowest average distance to the other points in the cluster. By 'distance' we mean Euclidean distance (although other metrics can be used as well). The resulting clusters will group scenarios which are closer together.

When applied to scenario reduction, the number of clusters is set to be the size of the new scenario set. The method then clusters the scenarios, and the medoids of the clusters are chosen to be the new scenario set. The idea behind this is that the scenarios in a given cluster will behave similarly to the medoid of their cluster.

In Figure 5.3.1, we plot the scenarios in Euclidean space (which we will refer to as the 'scenario space'). We see that scenarios 1, 2, and 3 are closer to each other than they are to scenarios 4, 5, and 6, with 6 being much further away from scenarios 4 and 5. Therefore, the clustering algorithm should (and does) sort the scenarios into these groups when k-medoids is applied (the groupings are shown by the different colours). The medoid scenarios of the clusters are shown by the triangles. While this is trivial for the orange cluster (scenario 6) and the yellow cluster (scenarios 4 and 5), we see that the 'middle' scenario (scenario 2) is chosen for the blue cluster (scenarios 1, 2, and 3).

Scenarios 2, 5, and 6 are the representative scenarios chosen for their respective clusters, with assigned probabilities of $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{6}$, respectively. From Figure 5.3.1, this appears a reasonable scenario set to select. The clusters seem to divide the set into relatively similar scenarios, with the probabilities representing the concentration of probability mass appropriately.

Issues arise because two quite different scenarios can perform similarly on a given solution. To demonstrate this, we chose two example solutions, shown in Table 5.3.2.

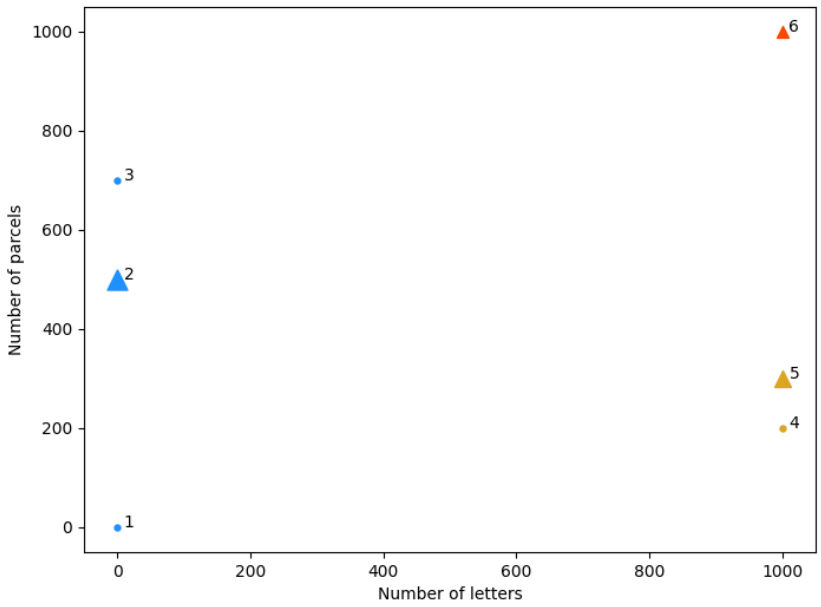


Figure 5.3.1: Scatter plot of scenarios from Table 5.3.1. Different colours represent the different clusters chosen. Triangles represent the medoid scenario kept by the scenario reduction method. The size of the triangles represents the relative probability mass in the reduced scenario set.

Table 5.3.2: Number of workers in each WA for the two candidate solutions.

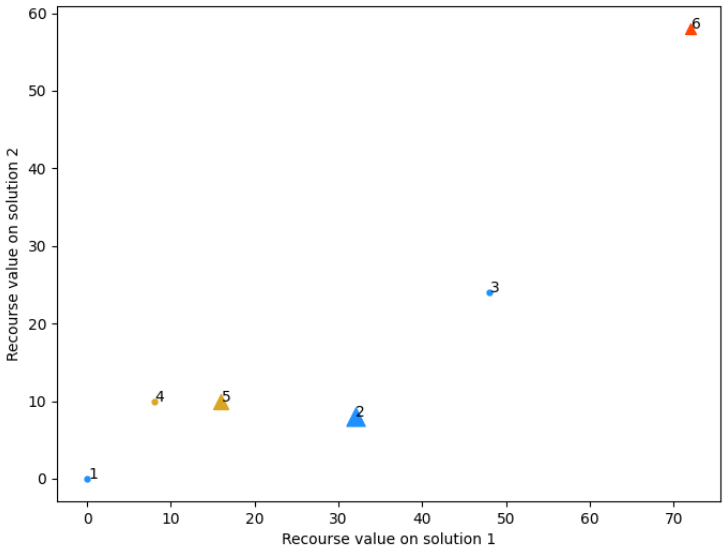
	Letters WA workers	Parcels WA workers
Solution 1	20	5
Solution 2	5	20

The first solution prioritises sorting letters over parcels, the second does the reverse. We calculate the recourse value for each scenario with the two solutions, and plot these recourse values in Figure 5.3.2a.

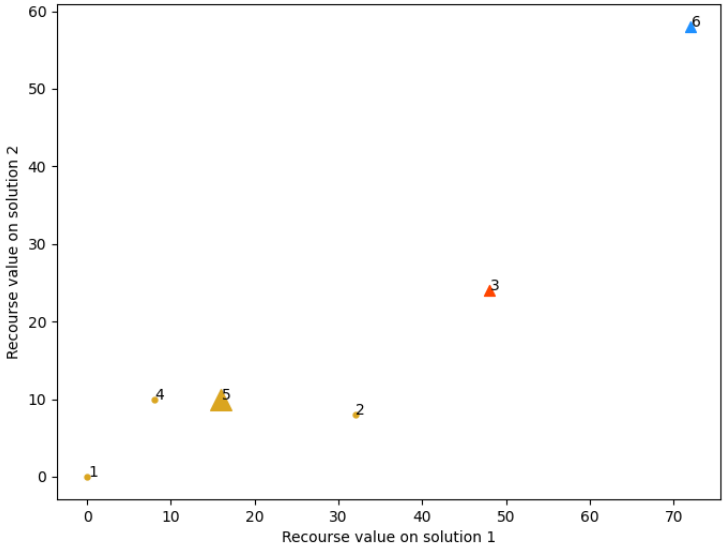
Figure 5.3.2a shows the new behaviour of the scenarios in this transformed space, which we refer to as the *recourse space*. We see that the relative distances between the scenarios has changed. The most notable change is for scenarios 4 and 5, which now lie in between scenarios 1 and 2, and scenario 3 which has moved closer to scenario 6. Looking at the medoids previously found, we now see that scenario 5 is now much closer to scenario 1 than scenario 2 is, and that scenario 3 has moved quite far away from any other scenarios. However, much of the probability mass ($\frac{1}{2}$) is still located at scenario 2, meaning less mass is located near scenario 3 or scenario 1. This means that if these scenarios represent important regions of the space, then this information will be lost in the reduced set. The region near scenario 1 will be under-represented, and the region near scenario 3 will not be represented at all.

Figure 5.3.2b shows the clusters we obtain if we perform the clustering after transforming to the recourse space. If we cluster in this space, we would obtain clusters of scenario 1, 4, 5, and 2, scenario 3, and scenario 6. These clusters clearly group scenarios in a more appropriate way than our previous clustering. In addition, the medoid scenarios (3, 5, and 6) and their probabilities ($\frac{1}{6}$, $\frac{2}{3}$, and $\frac{1}{6}$, respectively) distribute the mass across the space more appropriately. The assignment of $\frac{2}{3}$ to scenario 5 covers the high number of scenarios with low recourse on both solutions (i.e. the bottom left corner of the plot) and the addition of $\frac{1}{6}$ probability to scenario 3 now covers regions of medium-high recourse on solution 1, and medium recourse on solution 2.

The two different scenario sets (found by clustering before vs after transforming the



(a) Scatter plot of recourse values for applying each candidate solution in Table 5.3.2 to each scenario in Table 5.3.1. The colors and marker symbols show the same clusters and representative scenarios as shown in Figure 5.3.1.



(b) Scatter plot of recourse values for applying each candidate solution in Table 5.3.2 to each scenario in Table 5.3.1. The colors now represent the clusters chosen if clustering is performed in this space. Again, the triangles show the chosen scenarios, with sizes proportional to their assigned probabilities.

Figure 5.3.2: Clusterings performed in the 5.3.2a scenario space, and 5.3.2b the recourse space, projected into the recourse space.

Table 5.3.3: OOS results for the solutions found using standard clustering or recourse space clustering.

	Scenario space clustering	Recourse space clustering
Letter WA workers	10	10
Parcels WA workers	25	15
Cost scenario 1	35	25
Cost scenario 2	35	41
Cost scenario 3	51	57
Cost scenario 4	35	25
Cost scenario 5	35	25
Cost scenario 6	75	81
Average cost	44.33	42.33
Optimality gap	4.51%	0.00%

scenarios) yield very different solutions. To show this, we used the two reduced scenario sets to solve the mail centre problem, and calculated the average recourse cost over the full scenario set. These are shown in Table 5.3.3.

We see that the recourse space clustering gives a different solution to that of the scenario space clustering. Both allocate 10 workers to the letters WA. The original clustering (which we will call the *scenario space* clustering) allocates 25 to the parcels WA, whereas the recourse space clustering solution only allocates 15. This results in the recourse space clustering solution having 10 fewer workers overall. The reason for this difference is that the second scenario set (found when clustering after transforming) places more probability mass in the region containing scenarios 1, 2, 4, and 5. These are the scenarios with fewer parcels. Therefore, fewer workers are allocated to the parcels WA. While each solution performs better on three of the six scenarios, we see the average cost over all the scenarios is lower for the recourse space clustering solution. Furthermore, when solving the entire stochastic program over all 6 scenarios, we obtain an objective value of 42.33. We see that the recourse space clustering solution achieves the same objective value as the full problem (meaning it is an optimal solution). However, the scenario space clustering solution is 4.51% worse than the solution from the full model. This shows the recourse space clustering leads to a better overall solution.

Our previous example shows both the issues with the performance of scenario reduction using only scenario information, as well as the benefit of incorporating problem information in the reduction process. We also see that in both the example from the literature and our toy example, this similarity/dissimilarity was found by applying a number of solutions to these scenarios, and using this information in the scenario reduction procedure. This suggests that a common framework could be used to make a number of existing scenario reduction methods problem-based. In the remainder of this section, we propose and describe such a framework.

5.3.2 Proposed framework

Our framework consists of three main steps. The first step in this framework is to create a pool of M *candidate solutions*, which we denote y_m, $m = 1, ..., M$. The idea behind using the candidate solutions is to provide the information as to how similarly different scenarios react to the different solutions. Using a pool allows us to examine the scenarios in multiple situations. The user can use a number of methods to calculate these solutions. We discuss options for this in Section 5.3.3.

After creating the candidate decisions, we need to use them to transform the scenarios. We denote our full or true scenario set as $\mathcal{S} = [\xi_1, ..., \xi_S]$ of size S. We transform each scenario $\xi_s \in \mathcal{S}$ into the recourse space (as also defined in our motivating example). We do this by calculating the recourse values $Q(y_m, \xi_s)$ for each candidate solution y_m and scenario ξ_s. These Q values are then used to populate the $S \times M$ *recourse matrix*:

$$R := \begin{pmatrix} Q(y_1, \xi_1) & \cdots & Q(y_m, \xi_1) \\ \vdots & \ddots & \vdots \\ Q(y_1, \xi_S) & \cdots & Q(y_m, \xi_S) \end{pmatrix}$$

where R represents the projection of the scenarios into the recourse space.

This information is then used in the scenario reduction methods. Any scenario

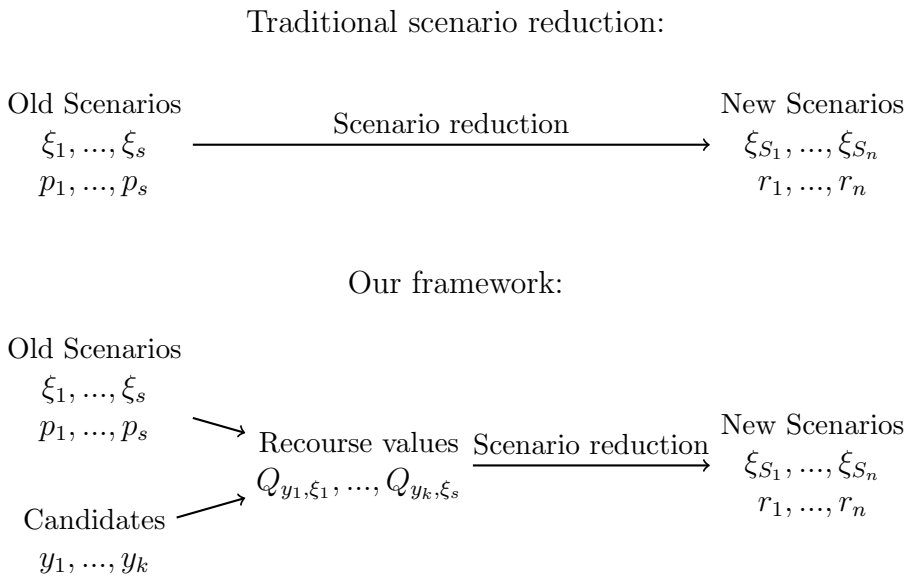


Figure 5.3.3: Diagram comparing approaches to scenario reduction.

reduction that works directly on the scenarios (i.e. in the scenario space) can be applied using the recourse space instead. For example, k-medoids clustering normally uses the Euclidean distance between the rows of R to make this method problem-based.

Note that some authors use R to create new variables/distance metrics which they use instead of R. We give more details of this in Section 5.3.3.

In summary, our framework consists of three steps:

1. Creating a pool of candidate solutions, $y_1, ..., y_M$.

2. Calculating the recourse matrix R, to transform the scenarios to the recourse space.

3. Using the information from the transformed scenarios in the chosen scenario reduction method (for example, k-medoids clustering).

A diagram of this framework is shown in Figure 5.3.3.

Figure 5.3.3 highlights the independence of each of the above steps, allowing different methods to be combined. They also require very little of the specific stochastic problem they are applied to, meaning this framework can be widely applied. All that is required

is a set of candidate solutions (or a suitable method to generate this set) and to be able to calculate the Q values to populate the R matrix.

5.3.3 Choosing candidate solutions

Part of the new framework involves testing the scenarios on a set of candidate solutions. This is to get recourse values of the scenarios for the scenario reduction methods. There is no set procedure on how to choose the candidate scenarios. As a guide, we suggest that candidate scenarios be:

1. Of reasonable quality.
2. Somewhat varied.
3. Relatively easy to calculate.

Points 1 and 2 were suggested by Narum et al. (2024). Point 1 is motivated by results from Narum et al. (2024) (and supported by similar results from Zhang et al. (2023)) that bound the approximation error on using the set of candidate decisions to approximate the expected recourse from other candidate decisions. The authors give a bound on this approximation containing a term proportional to the distance the new decision is from the set of candidate decisions. They argue that if the set of candidate decisions is therefore of higher quality, this will be closer to the good solutions we are trying to find (and approximate). The need for varied solutions (Point 2) is to better represent the space of recourse values for a wide range of feasible solutions. We suggest Point 3 as we need to calculate multiple candidate solutions. Therefore, we should avoid excessive computational times to calculate each solution.

There are many different and possible ways to choose the number of candidate solutions, and the methods of calculating them. Previous methods in the literature outline some possibilities:

- Single-scenario candidates (Hewitt et al., 2022; Keutchayan et al., 2023; Zhang et al., 2023).
- Multi-scenario candidates (Narum, 2020; Narum et al., 2024).
- Problem-specific heuristics (Prochazka and Wallace, 2020).

We explain the single-scenario and multi-scenario candidates in more detail here. As our scope is on creating a general framework, we do not discuss problem-specific heuristics.

Single-scenario candidates

Single-scenario candidates are calculated by solving a deterministic problem for scenarios in the full scenario set. That is, our candidate solutions are:

$$y_m \in \text{argmin}_{y \in \mathcal{Y}} Q(y, \xi_m), m \in S$$

That is, y_m is an optimal decision if ξ_m was the observed scenario.

Some authors (Bertsimas and Mundru, 2023; Hewitt et al., 2022) use R to create a different (but related) distance measure to use for their scenario reduction method, which we refer to as the *opportunity cost* distance (Hewitt et al. (2022) first used this terminology). We briefly describe this here.

The opportunity cost approach The opportunity cost can be viewed as standardising single-scenario candidate recourse values by the recourse found when using the optimal solution for the given scenario. To formally explain this, we first define

$$\delta_{ms} := Q(y_m, \xi_s) - Q(y_s, \xi_s)$$

That is δ_{ms} is the additional cost if scenario s is observed, but we 'plan for' scenario m. This is in effect the opportunity cost of planning for scenario m (hence the name of this space).

There is no guarantee that $\delta_{ms} = \delta_{sm}$. Therefore, to ensure opportunity cost is symmetric, we make the final calculation, and define:

$$d_{sm} := \frac{1}{2}(\delta_{sm} + \delta_{ms})$$

where d_{sm} is the opportunity cost distance between scenarios s and m. This is the metric used by Bertsimas and Mundru (2023). Hewitt et al. (2022) uses a near-identical metric, which differs in that the two distances are added, but the sum is not multiplied by a half. We will keep the constant $\frac{1}{2}$ in our experiments.

We note that Keutchayan et al. (2023) refer to their approach as the opportunity cost. However, they do not include the $Q(y_s, \xi_s)$ term in their calculation of δ, nor do they perform the symmetry step to calculate d. Therefore, their transformation would be regarded as using single-scenario candidates to calculate the recourse space under our framework.

Multi-scenario candidates

Another approach is used by Narum (2020) and Narum et al. (2024). In their approach, the candidate solutions are found by solving smaller stochastic programs. That is, for each candidate solution m, a small scenario set of size r is selected, and the stochastic program is solved for this smaller scenario set, giving the candidate solutions. The scenario sets are usually selected using random sampling, but could potentially be found by other existing scenario set selection methods.

5.3.4 Requirements on the scenario reduction methods

The requirements of the scenario reduction methods we can use in this framework are more restrictive. The key requirement is that the scenarios chosen in the reduced subset are already present in the full scenario set. This is because there is no method for transforming the scenarios back from the recourse space into the original scenario space. This means that we cannot be certain that any new scenario created by a method will fall in the appropriate region of the recourse space once transformed. For example, in k-medoids clustering, the chosen scenarios are the medoids of the clusters, which are observations (i.e. scenarios) present in the cluster. However, for k-means clustering, the cluster centre is given by a created observation taking the mean value of each dimension of the observations in the cluster. If we used this method of clustering instead, we would be unable to create a new scenario which would take the mean value of the recourse in each dimension in the cluster. This requirement rules out some methods for use within this framework, such as k-means clustering (as described), Latin Hypercube Sampling, variance reduction techniques (such as importance sampling or antithetic sampling), and the moment-matching method of Høyland et al. (2003), among others. We clarify which reduction methods are suitable with our framework in Table 5.3.4.

Table 5.3.4: Scenario reduction methods classified by their suitability with our framework.

Suitable for our framework	**Unsuitable for our framework**
k-medoids clustering	k-means clustering
Output distribution decomposition with SVD	Moment matching
Spectral clustering	Variance reduction
Means matching	Latin hypercube sampling

5.3.5 Applicability to current approaches

The purpose and benefit of this proposed framework is how well it fits with current approaches already used in the literature. Table 5.3.5 describes other papers that fit

Table 5.3.5: Description of how previous literature fits into our proposed framework.

Reference	Candidates	Transformation	Reduction method
Narum (2020)	Multi-scenario	Recourse space	k-medoids
Hewitt et al. (2022)	Single-scenario	Opportunity cost	Spectral clustering
Zhang et al. (2023)	Single-scenario	Recourse space	Sparse regression to match means
Narum et al. (2024)	Multi-scenario	Recourse space	SVD decomposition of output distribution

into this framework.

It is worth noting that some of the methods have additional processing steps which sit outside of the framework. The method of Zhang et al. (2023) for example filters out some candidate decisions before solving. We also note some previous methods which are similar to this framework, but with small differences which make them unsuitable. For example, Keutchayan et al. (2023) uses the recourse space with single scenario candidates, combined with a clustering algorithm which is similar to k-medoids, but with an adjusted distance metric. Bertsimas and Mundru (2023) also use the opportunity cost distances with an algorithm based on k-means clustering.

Table 5.3.5 shows the range of candidate decisions, transformations, and scenario reduction methods which have already been used. We see that at each step of the framework (candidate decisions, transformations, and reduction methods), multiple approaches have been used, showing the variety of techniques that can be used with this framework. Furthermore, we identify a number of combinations that have not been used in the literature yet. These include single-scenario candidates with SVD, opportunity cost with k-medoids clustering and spectral clustering with either single or multi-scenario candidates, suggesting these combinations should be tested together.

5.4 Results

5.4.1 Experimental set-up

Having outlined our framework, we apply it to existing transformations in a numerical comparison. This will allow for a meaningful comparison between different approaches, and to test new combinations of candidate solutions, transformations, and methods. This also has the potential to reveal better-performing approaches within a step of the framework (for example, single vs multi-scenario candidates).

We test the suitability of the transformations for general problem-based scenario reduction. Therefore, we apply multiple scenario reduction methods to three different problems:

1. The Stochastic Service Network (SSN) problem from Crainic et al. (2016). This is a network design problem, in which we are designing a network to handle the flow of different commodities with stochastic demands.

2. The Telecommunications Network (TN) problem from Sen et al. (1994). Another network design problem, this describes designing a network to handle an unknown number of requests on a telecommunications network.

3. The stochastic mail centre staffing problem (MC) developed in Chapter 4.

The problems exhibit a number of different characteristics and behave in different ways. An overview of their characteristics is given in Table 5.4.1. The SSN problem is a small problem, which is often stable. The TN problem has a much higher dimension in the scenarios, and can be unstable. The MC problem is much larger than the other two problems, and hence struggles to solve for larger instances. The problem features vary in a number of ways as well, including; problem size, shape of the marginal distributions of variables in the scenarios, and the dimension of each scenario. Further information about the problems and their formulations is given in Appendix 5.A.

Table 5.4.1: Overview of features for the test problems

Problem	**Num. variables**	$\lvert\mathcal{S}\rvert$	**Marginals**	**Dependence**	**Dist. dimension**
SSN	1,260	1,000	Uniform	Independent	10
TN	1,152	1,000	Gamma	Independent	82
MC	286,179	52	Empirical	Empirical	180

For the SSN and TN problems, scenarios are generated through a distribution-based process. Since we are testing scenario reduction methods, we use the process to generate 1,000 scenarios for each, and treat this as the full scenario set. When conducting experiments, we reduce this set down to 3, 5, 10, and 20 scenarios. For the MC problem, scenario data was provided by a UK-based mail company, covering observed mail volumes in a given mail centre over a one-year period (April 2020 to March 2021). We then use this data as the empirical distribution of the scenarios. Given the numerous differences in the mail centre problem between days of the week, we only solve the problem for Mondays. This gives a full scenario set of 52 scenarios. Due to the large size of this problem, we reduced our full scenario set to sets of 2 - 10 scenarios for this problem when performing our experiments.

For each method, we test four combinations of transformations and methods. For transformations we consider:

1. The Euclidean distance between the different scenarios. We refer to this as the *scenario space*.

2. The recourse space using single-scenario candidates (SSC). The candidate decisions were found by randomly selecting scenarios from the full scenario set, and solving the problem for each given scenario. We used 100 candidate decisions for the SSN and TN problems. Given we only had 52 scenarios for the MC problem, we created a candidate decision for every scenario in the set for this problem.

3. The opportunity cost (OC) space with single-scenario candidates, as described in Section 5.3.3.

4. The recourse space using multi-scenario candidates (MSC). The candidate decisions were found by randomly sampling 6 scenarios from the full scenario set and solving a small stochastic program over these scenarios for each candidate decision. We used 100 candidate scenarios for each problem.

For methods, we use the following scenario reduction techniques:

1. k-medoids clustering (KM) (Kaufman, 1990). This clustering method partitions the set of observations into clusters, where the cluster centroid is the observation in the cluster with the lowest average distance to other observations in that cluster.

2. The SVD and output decomposition method (Narum et al., 2024). In this method, SVD is performed on a space matrix, and a linear program (with a random objective) is solved to select a scenario set which preserves the expectations of the singular vectors with the highest singular values.

3. Spectral clustering (SC) (Von Luxburg, 2007). This clustering method constructs an affinity matrix of the observations, before clustering the observations based on the eigenvalues of the affinity matrix.

As stated in Section 5.3.4, these methods are scenario reduction methods that return a subset of the full scenario set, and thus are suitable for our framework. We also note from Table 5.3.5 that they have also all been used in problem-based scenario reduction before. We note that we do not use the means-matching method from Zhang et al. (2023). We found that this method is more complicated to implement due to the filtering step, and the bespoke algorithm for solving the regression problem.

For each combination of problem, method, and transformation, we will compare the out-of-sample (OOS) stability to evaluate the performance of the different approaches. In the OOS stability, we reduce the initial scenario set to the desired number of scenarios 20 times. For each of these 20 reduced scenario sets, we solve the stochastic program to get a solution. We then apply this solution to each of the scenarios in the full

scenario set, find the objective cost, and take the mean of these objective costs. We then examine the spread of the 20 average recourse costs to determine the effectiveness of the scenario reduction.

All mixed integer/linear programming was done with Gurobi v10.0.3, with the Python v3.9 API. Additional programming was also conducted in Python v3.9. We apply the k-medoids clustering with the alterating algorithm from the 'kmedoids' python package v0.4.0. We implemented the SVD procedure from Narum et al. (2024) in Python. We also implemented the spectral clustering algorithm as presented in Hewitt et al. (2022). We made one modification to the method. Instead of using the k-means clustering for the clustering step, and then finding cluster medoids, we instead applied the k-medoids clustering algorithm directly to perform the clustering on the eigenvectors. This was so that the method directly returned scenarios that already existed in the full scenario set. In the procedure of Hewitt et al. (2022), the authors manually found the cluster medoids after the clustering had been performed. Furthermore, we found that the implementation of the k-means clustering made the method too stable for our problems.

5.4.2 Computational results

Random sampling

Firstly, we examine the OOS results for the three problems when using random sampling. This shows the stability and behaviour of the problems, and acts as a baseline method to which we can compare the other results. Figure 5.4.1 shows these results. The SSN problem has OOS values ranging between 0 and 500, the TN problem ranges between 0 and 35, and the MC problem ranges between 80,000 and 300,000. We notice that the SSN problem and TN problem both come quite close to 0 recourse at 20 scenarios. Specifically, the SSN problem achieves a median recourse of approximately 4, and the TN problem achieves a median recourse of 2. These problems also both show a

steady improvement in the spread of the recourse values for the increases in the scenario sizes. The MC problem behaves slightly differently. Firstly, the recourse values do not converge as closely to the minimum values (albeit over a different range of scenarios) as the other problems do. Furthermore, while the spread of the OOS values do narrow as scenarios are increased, this does not occur as quickly as it does for the other two problems. This suggests that this problem is a more unstable stochastic program to solve than the other two.

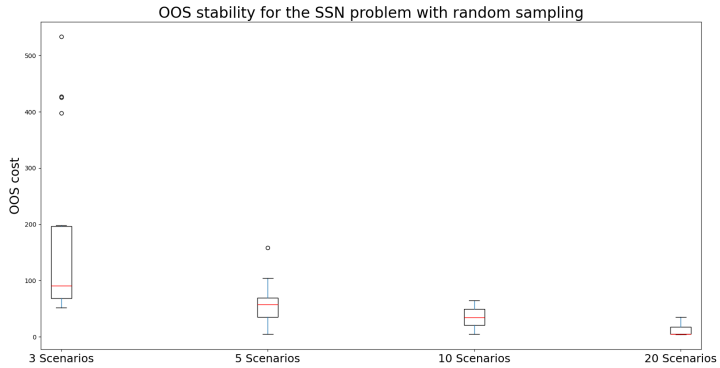
Having looked at the baseline levels for our three test problems, we now examine their results when using our scenario reduction methods. Figures 5.4.2, 5.4.3, and 5.4.4 show the OOS results for the SSN, TN, and MC problems, respectively. Each figure shows the results for the problem for the three different methods within the subfigures. Each subfigure shows the OOS results from 20 runs for the three different spaces for reduced scenario sets of an increasing sample size. A superior method/space is indicated by both a lower median value to the OOS results (to show a method is performing better) and a lower spread of values (to show a method is estimating the recourse cost consistently).

Stochastic service network problem

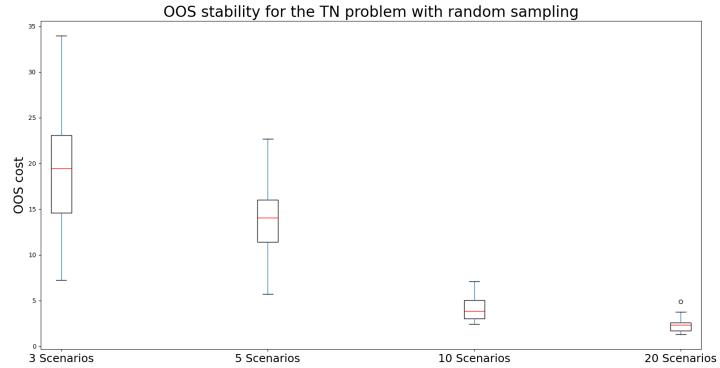
We find that in most cases using MSC results in a lower median and spread of OOS values, shown in Figure 5.4.2.

For the KM method (Figure 5.4.2a), we see that MSC has the lowest median OOS cost for all scenario set sizes, and has the lowest spread of OOS values when using 5 or more scenarios. OC and SSC have similar medians, with SSC having slightly lower medians for 10 and 20 scenarios. The scenario space has a much higher median than all other spaces for 3 scenarios, but relatively similar medians to SSC and OC when 5, 10, or 20 scenarios are used.

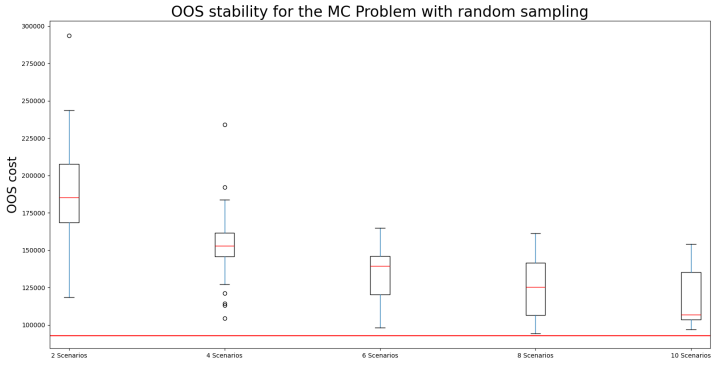
For the SVD method, we see that the MSC actually has the highest median OOS



(a) Out of sample stability for the RS reduction method applied to the SSN problem.



(b) Out of sample stability for the RS reduction method applied to the TN problem.



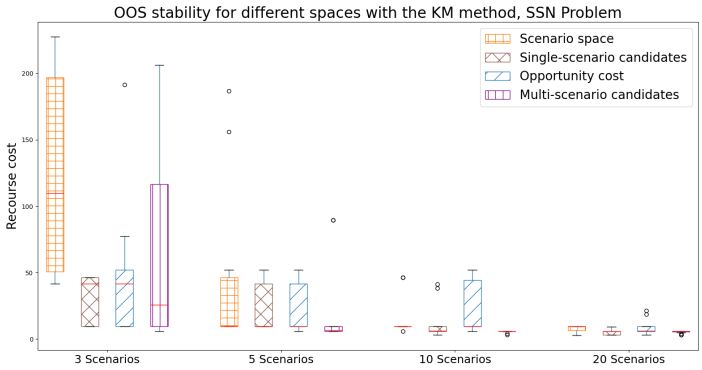
(c) Out of sample stability for the RS reduction method applied to the MC problem.

Figure 5.4.1: OOS results for the random sampling method applied to the three problems

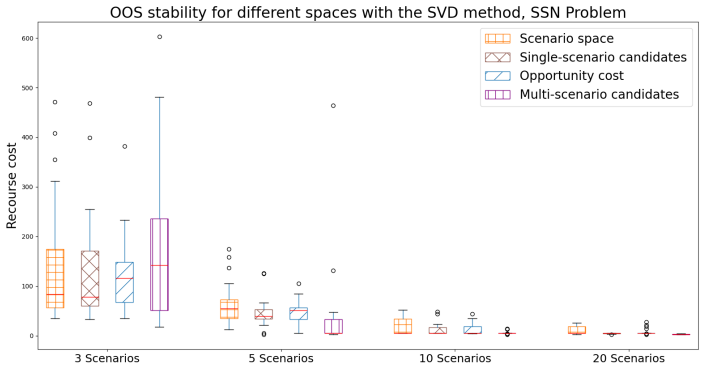
of the four transformations, with the lowest being the SSC, when 3 scenarios are used. However, when more scenarios are used, we see that MSC then has the lowest OOS cost of the spaces for 5 and 10 scenarios, and equal lowest with the other spaces at 20 scenarios. SSC and OC perform relatively similarly across all scenario sizes (SSC being slightly lower for 3 and 5 scenarios). The scenario space performs better than OC and MSC, and similarly to SSC for 3 scenarios. However, it performs either equally or worse in terms of median OOS, and worse in terms of spread than the other spaces for 5, 10, and 20 scenarios.

The behaviour of the SC method in Figure 5.4.2c is different again. We see that the MSC generally shows the lowest median OOS cost of the 4 spaces, being the lowest for 3, 10, and 20 scenarios and second lowest (after OC) for 5 scenarios. We also see the scenario space being the highest OOS cost for 5 or more scenarios.

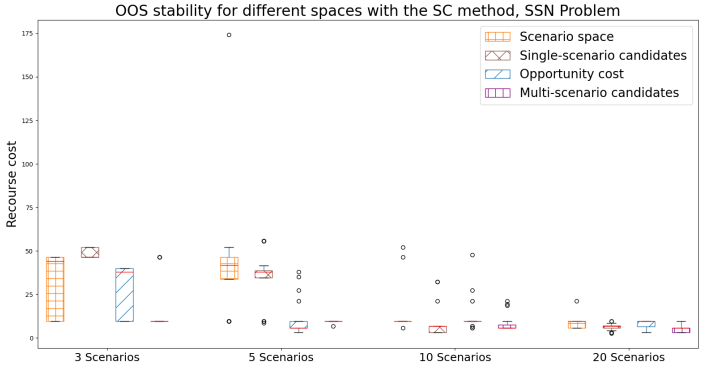
We note two main differences between the SC method and the other methods. First, we note that the OC space shows clearly lower OOS than SSC for 3 and 5 scenarios, with the two being either similar, or SSC performing better at lower scenarios for the other methods. Second, the spread of OOS values is much lower for all spaces when three scenarios are used. This could be because the nature of spectral clustering means that the k-medoids clustering step is only performed in s dimensions. When $s = 3$, this is lower than the dimension of the scenarios of this problem (10). Clustering algorithms generally perform better in lower dimensions (Houle et al., 2010), meaning that it is possible that the method more consistently finds optimal or close to optimal clusters across the different OOS runs, compared to the standard k-medoids clustering method. We also note that the SVD method has more inherent randomness built into the method than clustering methods (which do have a theoretical 'best' clustering).



(a) Out of sample stability for the KM reduction method applied to the SSN problem.



(b) Out of sample stability for the SVD reduction method applied to the SSN problem.



(c) Out of sample stability for the SC reduction method applied to the SSN problem.

Figure 5.4.2: OOS results for the three reduction methods applied to the SSN problem

Telecommunications network problem

For the TN problem, shown in Figure 5.4.3, we again see that MSC shows the lowest median OOS values in the majority of cases. It is the lowest for all scenario set sizes for the KM and SVD methods, and the lowest for 5, 10, and 20 scenarios.

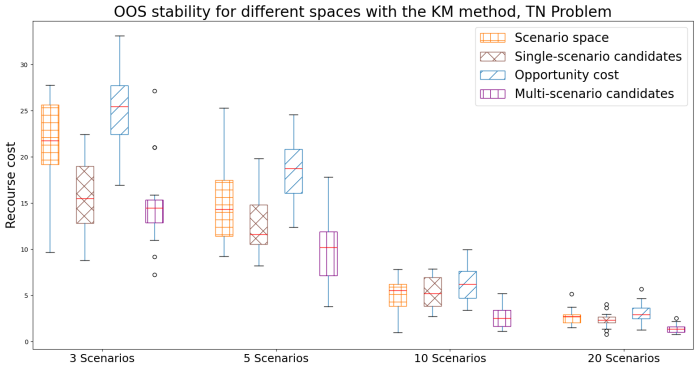
For the KM method (Figure 5.4.3a), we note that the OC space is the worst performing of the four spaces. This is particularly noticeable for smaller scenario set sizes (3 and 5 scenarios). Between the other two spaces, the SSC space performs better than the scenario space for 3 and 5 scenarios, and the two perform similarly for 10 and 20 scenarios.

The spaces perform more similarly as scenario set sizes increase for the SVD method (Figure 5.4.3b). The MSC space performs best, the OC and SSC spaces perform similarly (except for 5 scenarios, where SSC performs better), and the scenario space performs the worst. However, the difference between the spaces reduces as the number of scenarios increases. We also see that for each scenario set size, the spread of OOS values for each space is reasonably similar, especially as the number of scenarios increases.

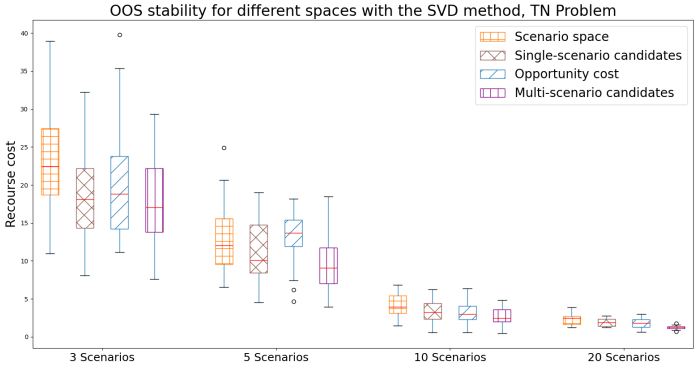
For the SC method, there is less of a consistent pattern between the spaces for different scenario set sizes. For larger scenario sets, we again see MSC having the lowest median OOS, and OC having the highest, with all spaces having similar OOS spreads. However, for 3 scenarios, we see MSC has the highest median OOS, and that the medians for the other three spaces are similar (as they are for 5 scenarios as well).

Mail centre problem

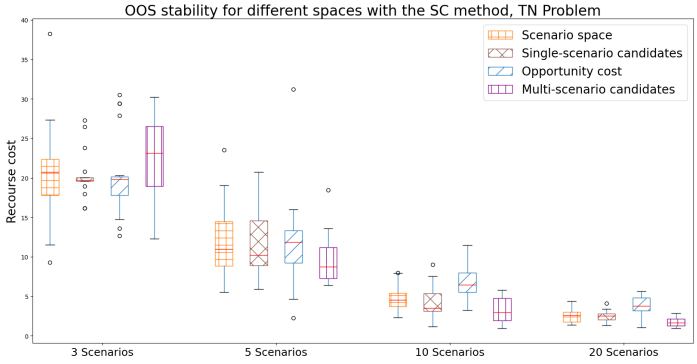
The results for the MC problem are more mixed, as shown in Figure 5.4.4. First, we note that in this instance, the full scenario set is small enough that we can solve the stochastic program over the entire set. The objective value for the full stochastic program is shown in each sub-figure of Figure 5.4.4 via the solid red line. Secondly, we note that there is much more varied behaviour of the three spaces between the methods



(a) Out of sample stability for the KM reduction method applied to the TN problem.



(b) Out of sample stability for the SVD reduction method applied to the TN problem.



(c) Out of sample stability for the SC reduction method applied to the TN problem.

Figure 5.4.3: OOS results for the three reduction methods applied to the TN problem.

for this problem.

For the KM method (Figure 5.4.4a), the MSC space shows the highest median OOS of all spaces over all scenario set sizes. This is in contrast to the previous problems and methods, where MSC was routinely the lowest. Instead, we see that the SSC space has the lowest OOS cost for 4 or more scenarios (with OC having the lowest for 2 scenarios). The scenario space performs poorly with 2 scenarios (only MSC has a higher median OOS cost), but performs similarly to OC for increasing sample sizes.

The SVD method (Figure 5.4.4b) appears to behave more similarly to the previous two problems. The MSC space has the lowest median OOS for 4, 6, 8, and 10 scenarios, and has the second lowest (after SSC) for two scenarios. However, we see that the other three spaces perform quite similarly to each other as the scenario set size increases. We also note that (unlike the other two methods) the OOS costs for all spaces is approaching the minimum at 10 scenarios.

The results for the SC method are again different to the previous methods (Figure 5.4.4c). In this case, there is not a clear relationship between the spaces as the scenario set size increases. The MSC shows the lowest median OOS cost for 2 and 4 scenarios, but the highest for 6, 8, and 10. Furthermore, we see that the scenario space jumps from having the highest OOS cost for 2 scenarios to the lowest for 8 scenarios. The SSC and OC results also do not seem to show a clear relative relationship to the other spaces as the scenario set sizes increase.

While the spaces behave similarly with the SVD method as they do with the other methods and problems, they behave quite differently with the KM and SC methods for this problem. This is possibly because these methods simply do not perform as well as the SVD method does for this problem. We examine this by comparing the error for each reduction method with multi-scenario candidates for this problem. We compare the errors by reducing to 6 scenarios with each method, and calculating the average error for this scenario set over 10 candidate solutions, found by solving the stochastic

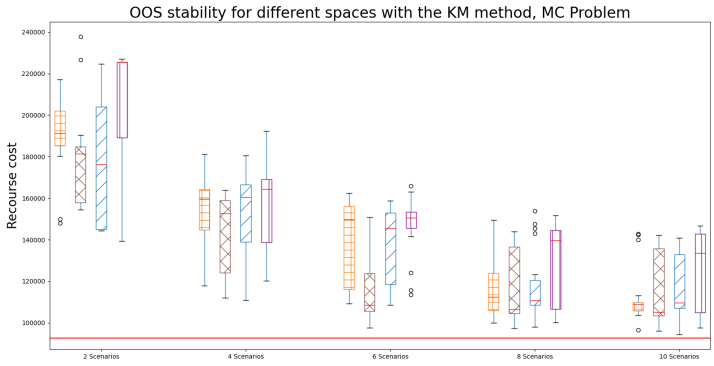
program over 6-scenario scenario sets, chosen by random sampling. We perform 10 trials of this to account for the randomness in the sampling procedures. A detailed explanation of the error calculations is given in Appendix 5.B. We show these errors as boxplots in Figure 5.4.5.

The errors are much lower for the SVD method than the other two methods. This shows that the SVD method is selecting scenario sets that give an objective cost much closer to that of the full stochastic program, indicating that this method is performing better for this problem than the other methods. This is consistent with the OOS results from Figure 5.4.4, explaining why the multi-scenario candidates perform relatively poorly for this problem and these methods.

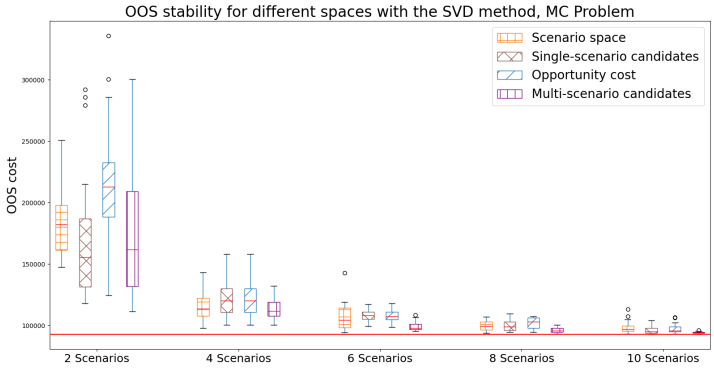
Summary

Looking across the different methods and problems, we broadly see that the MSC space generally has the lowest OOS cost. This is particularly prevalent for the SSN and TN problems. For these problems, we also find that the SSC space performs reasonably well, often being the second-best performing space, particularly as the number of scenarios is increased. Furthermore, it is the best-performing space for larger scenario sets for the MC problem with the KM and SC methods, where the MSC performs relatively poorly. It is possible that the underlying structure of the output distribution for this problem is more complex than the others. Therefore, more variety in the candidate solutions is needed to capture it, hence the SSC space performs better.

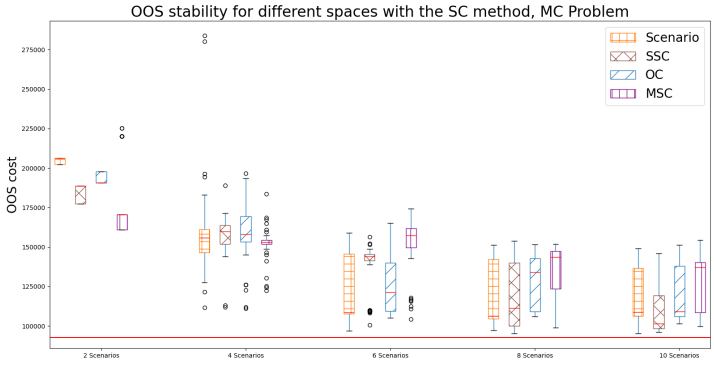
We note that the two most unusual results were for the KM and SC methods with the MC problem. These experiments were the only combinations of problem and method which did not show the MSC to be the best transformation. Further analysis showed that the errors found for the MSC space with either the KM or SC methods were much higher than those of the SVD method. This indicates that these methods are choosing poor scenario sets with this transformation and this problem, suggesting they are not



(a) Out of sample stability for the KM reduction method applied to the MC problem.



(b) Out of sample stability for the SVD reduction method applied to the MC problem.



(c) Out of sample stability for the SC reduction method applied to the MC problem.

Figure 5.4.4: OOS results for the three reduction methods applied to the MC problem. Solid red line shows the true minimum when solving over the entire scenario set.

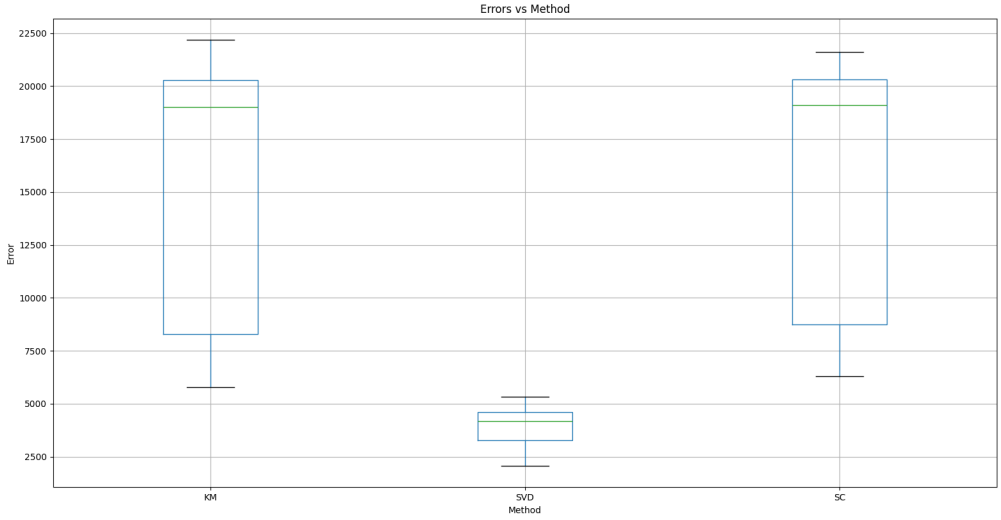


Figure 5.4.5: Sampling errors for the MC problem with the different methods

appropriate for this problem. However, we do find that single candidate solutions perform well for this problem.

5.5 Conclusion

Given the trade-offs involved in scenario set selection for stochastic programming, it is vital to be able to choose a small but representative subset. Previous research shows the benefits of problem-based scenario set selection over distribution based. Here, we present a framework to help generalise existing problem-based scenario reduction approaches, and use this framework to compare and combine existing approaches in the literature. We tested four different transformations with three different scenario reduction methods and three different test problems.

We saw that for most combinations of problem and method, using MSCs gave the best performance regarding OOS costs. Using SSCs also performed reasonably well, especially for larger scenario set sizes. Given these are computationally much easier to calculate, this shows that the SSC space is still a useful approach, showing the benefit

of using these in a general problem-based method. We also saw poorer performance from the MSC space when using the clustering methods. This suggests that there is no 'free lunch' when it comes to choosing candidate solutions. Different transformations may work better or worse for different problems.

A limitation of this work is that there is no way to 'place' a scenario in the recourse space. For example, one of the methods we looked at was k-medoids clustering. In k-medoids clustering, the cluster centre is the medoid member of the cluster. That is, the cluster centre is an already existing point in the cluster set. If we wanted to use k-means clustering instead, then we would need to calculate a new mean point of the cluster, which would be the cluster centre. In the scenario space, this is straightforward. However, there is no way to create a scenario which we know will be the mean of the scenario set once projected onto the recourse space. Therefore, we can not use this framework with such a method. The bigger implication is that we cannot use any scenario generation methods with this framework, by the same logic. Thus, we are limited to scenario reduction techniques. A second limitation is that calculating the recourse matrix can be time consuming, as many deterministic or stochastic models need to be solved to do this. However, we note that once the recourse matrix has been calculated, it can easily be applied to multiple scenario reduction methods.

Despite this limitation, this represents a significant contribution to the scenario reduction literature. This framework gives a straightforward method to make scenario reduction methods problem-based. Very mild conditions are required for the problem to be solved. These amount to being able to find reasonable feasible solutions, and calculating recourse costs for scenarios under given solutions in a reasonable time. We also saw when comparing the different spaces that using MSCs and SSCs performed better than the scenario space in the majority of cases, showing the value of applying this framework. Therefore, we have shown this framework can make valuable improvements to existing scenario reduction techniques.

5.A Description of test problems

5.A.1 Stochastic service network problem

The stochastic service network problem is taken from Hewitt et al. (2022). The problem is a network design problem, with stochastic demand at each node. Our problem is to choose which arcs to install in the network, at the lowest cost, and with the lowest expected cost of unmet demand in the scenarios.

We view our network as a direction graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with a node set $\mathcal{N}$ and arc set $\mathcal{A}$. We also have a set of commodities $\mathcal{K}$, and a set of scenarios $\mathcal{S}$, with associated probabilities $p_s, s \in \mathcal{S}$. Each scenario represents different demands d_i^{ks} giving the demand for commodity $k \in \mathcal{K}$ at node $i \in \mathcal{N}$ in scenario $s \in \mathcal{S}$, as well as the different capacities u_{ij}^s for arc $(i, j) \in \mathcal{A}$ in scenario s.

From here, we somewhat modify the formulation from Hewitt et al. (2022). We assume that if we wish to send flow of commodity k in scenario s along arc (i, j) we can do so in one of two ways. Either:

1. We can install an arc (i, j) for fixed cost f_{ij}, then use this arc at a cost c_{iJ}.
2. We can send the item along auxiliary arc linking nodes i and j at a cost much higher than c_{ij}, denoted as P.

From these, we can define our decision variables:

- y_{ij}: binary decision variable indicating if we will install arc (i, j) in the network.
- x_{ij}^{ks}: continuous variable indicating how much flow of commodity k we send along installed arc (i, j) in scenario s.
- z_{ij}^{ks}: continuous variable indicating how much flow of commodity k we send along auxiliary arc (i, j) in scenario s.

With this information, we obtain the following formulation:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{s\in\mathcal{S}} p_s \left(\sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} (c_{ij}x_{ij}^{ks} + Pz_{ij}^{ks}) \right) \\
\text{s.t.} \quad & \sum_{j\in N^+(i)} (x_{ij}^{ks} + z_{ij}^{ks}) - \sum_{j\in N^-(i)} (x_{ji}^{ks} + z_{ji}^{ks}) = d_i^{ks}, \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \\
& \sum_{k\in\mathcal{K}} x_{ij}^{ks} \leq u_{ij}^s y_{ij}, \forall (i,j) \in \mathcal{A}, \forall s \in \mathcal{S} \\
& y_{ij} \in \{0,1\}, x_{ij}^{ks}, z_{ij}^{ks} \geq 0, \forall (i,j) \in \mathcal{A}, \forall k \in \mathcal{K}, \forall s \in \mathcal{S}
\end{aligned}$$

where $N^+(i)$ are the set of successor nodes to node i, and $N^-(j)$ are the predecessor nodes.

To generate the instances, we followed similar rules to Crainic et al. (2016). When performing our experiments, we generated instances consisting of 10 nodes, 60 arcs, and 10 commodities. The arcs where randomly chosen from the set $\mathcal{A} \times \mathcal{A}$ with equal probability.

The d, u, c, and f parameters were randomly drawn from uniform distributions between 1 and 10 (d and u varied across scenarios, c and f were held constant across scenarios). After random generation, the c and f parameters were adjusted so that we could control the relative importance of the different costs and the tightness of the bounds. For this we defined the capacity ratio C and the fixed cost ratio F as follows:

$$\begin{aligned}
C &= \frac{|A|T}{\sum_{(i,j)\in A} u_{ij}} \\
F &= \frac{|K| \sum_{(i,j)\in\mathcal{A}} f_{ij}}{T \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k} \\
\text{where } T &:= \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}} d_i^k
\end{aligned}$$

For this problem, we set $C = 5$ and $F = 3$. We generated $|S| = 1000$ scenarios as

the full scenario set to sample from. We set the penalty for using the auxiliary arcs to $P = 10000$.

5.A.2 Telecommunications network problem

Similarly to the SSN, the TN problem determines where in a telecommunications network to install extra capacity for telecommunications, given uncertain demand between pairs of nodes. The problem formulation was first given by Sen et al. (1994). We use this formulation, with instances and scenario data from Narum et al. (2024).

For the notation, we let:

- $\mathcal{N} = \{1, ..., n\}$ be the set of links whos capacity can be increased.
- $\mathcal{M} = \{1, ..., m\}$ be the set of point-to-point demand pairs.
- $\mathcal{Z}$ be the set of scenarios, with associated probabilities p_z.
- d_i^z be the stochastic demand for demand pair i in scenario z.
- b be the total possible capacity that can be added to the network.
- $R(i)$ be the set of routes that can be used for demand pair i.
- a_{irj} be a binary variable indicating if link j is present in route $r \in R(i)$.
- e_j be the base/existing capacity for link j.

We then also define the decision variables:

- x_j: the amount of capacity to add to link $j \in \mathcal{N}$.
- f_{ir}^z: the number of connections associated with pair i using route $r \in R(i)$ in scenario z.
- s_i^z: the unmet demand for pair i in scenario z.

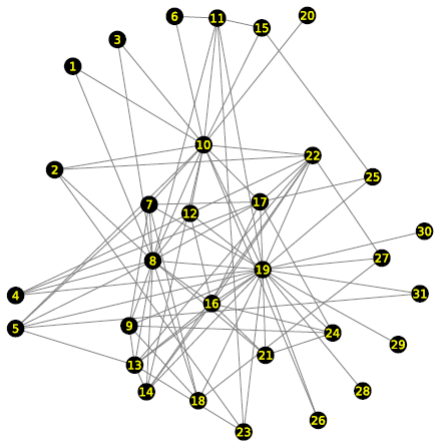


Figure 5.A.1: The network used in the TN problem (Narum et al., 2024)

With this, we can now define the stochastic program

$$\min \sum_{z \in \mathcal{Z}} \sum_{i \in \mathcal{M}} s_i^z$$

$$\text{s.t.} \sum_{j \in \mathcal{N}} x_j \leq b$$

$$\sum_{i \in \mathcal{M}} \sum_{r \in R(i)} a_{irj} f_{ir}^z \leq x_j + e_j, \forall j \in \mathcal{N} \forall z \in \mathcal{Z}$$

$$\sum_{r \in R(i)} f_{ir}^z + s_i^z = d_i^z, \forall z \in \mathcal{Z}$$

$$f_{ir}^z \geq 0, s_i^z \geq 0.$$

The network for our instance is shown in Figure 5.A.1.

This network consisted of 31 nodes, with 82 demand pairs, and 89 links. The median number of routes between each demand pair was 8.5. The demand for each demand pair was drawn from a Gamma distribution with a shape parameter of 5 and a expected value of 25 for all demand pairs. We assumed independence between all demand pairs.

5.A.3 Mail centre staffing problem

We use the mail centre staffing problem presented in Chapter 4. We use minimising the maximum number of workers as the first stage solution, and minimising delayed mail in each scenario as the second stage solution, with $\alpha = 0.5$, and no changes in the second stage permitted.

Sets and indices

- $\mathcal{S}$: the set of scenarios in the model, indexed by s.
- $\mathcal{W}$: set of WAs indexed by w.
- $\mathcal{E}$: set of mappings between WAs.
- $\mathcal{I}$: set of indirect mappings.
- $\mathcal{H}$: set of tethered WAs.
- $\mathcal{J}$: set of shifts, indexed by j.
- $\mathcal{T}$: set of time periods, indexed by t.
- $\mathcal{T}_j$: set of time periods associated with shift j. Note, $\mathcal{T}_j \subset \mathcal{T}$.
- $\mathcal{N}$: the set of nodes in the time expanded network, indexed by i.
- $\mathcal{A}$: the set of arcs in the time expanded network, indexed by a.
- $\delta^+(i)$ and $\delta^-(i)$: the sets of outgoing and incoming arcs for node i, respectively.
- $\mathcal{K}$: the set of commodities, indexed by k.
- $w_O(a)$ and $w_D(a)$: the origin and destination WAs associated with arc a.
- $t_O(a)$ and $t_D(a)$: the origin and destination times associated with arc a.

- $ID^+(w, w', t) := \{a | w_O(a) = w, w_D(a) = w', t_O(a) > t, t_D(a) = t_O(a) + 1\}$: a set of all the arcs originating at WA w and finishing at w', originating at a time later than t. This is used in defining constraints to enforce the indirect steams.

- $ID^-(w) := \{a | w_O(a) \neq w, w_D(a) = w\}$: a set of all the arcs originating at different WAs, and finishing at WA w. This is also used to define the indirect stream constraints.

Parameters

- p_s: the probability of scenario s.

- c_w: the number of staff members required to operate one processing unit in WA w.

- b_i^{ks}: the demand for commodity i and node k in scenario s. For the majority of nodes and commodities, this will be 0 - that is, flow simply passing through here. At the source nodes, this will be positive, as this is where flow enters the network. For sink nodes, this will be negative, indicating where flow leaves the network.

- v_a^k: the commodity specific capacity for arc a and commodity k. These are used to enforce that the commodities follow the correct mappings.

- $\rho_{w,w'}$: the proportion of total flow passing from WA w to WA w' from indirect mapping $(w, w') \in \mathcal{I}$.

- u_w: the total WA processing capacity per time period per processing unit for WA w.

- C_{wt}: the processing unit capacity for WA w in time t.

- α: the weighting given to the first stage objective. We set $0 \leq \alpha \leq 1$.

Decision variables

- First-stage decisions:
  - y_{wt}: number of processing units rostered in WA w for time t. Here, we use the term 'processing unit' to describe one person or machine that is used to sort the commodity in a WA.
  - g_j: auxiliary continuous variable, giving the number of workers required for shift j.
  - S_{wt} and F_{wt}: auxiliary binary variables indicating if WA w has started and finished in or before time t, respectively.
- Second-stage decision variables:
  - x_a^{ks}: amount of commodity k sent along arc a in scenario s.
  - d_w^s: The amount of r priority delayed mail generated by WA w in scenario s.

Formulation

$$\min \alpha \sum_{j \in \mathcal{J}} g_j + (1-\alpha) \sum_{s \in \mathcal{S}} p_s \left(\sum_{a \in \mathcal{A}_{All}} \sum_{k \in \mathcal{K}} x_a^{ks} + \sum_{w \in \mathcal{W}_R} d_w^s \right)$$

$$\text{s.t. } g_j \geq \sum_{w \in \mathcal{W}} c_w y_{wt}, \ \forall t \in \mathcal{T}_j, \forall j \in \mathcal{J}$$

$$\sum_{a \in \delta^+(i)} x_a^{ks} - \sum_{a \in \delta^-(i)} x_a^{ks} = b_i^{ks}, \ \forall k \in \mathcal{K}, i \in \mathcal{N}, s \in \mathcal{S}$$

$$x_a^{ks} \leq v_a^k, \forall a \in \mathcal{A}, \ \forall k \in \mathcal{K}, s \in \mathcal{S}$$

$$\sum_{k \in \mathcal{K}} \sum_{a | (w, W_d(a)) \in \mathcal{E}} x_a^{ks} \leq y_{wt} u_w, \forall w \in \mathcal{W}, s \in \mathcal{S}$$

$$\sum_{a \in ID^+(w,w',t)} x_a^{ks} = \rho_{w_1 w_2} \sum_{a \in ID^-(w)} x_a^{ks}, \ \forall k \in \mathcal{K}, \ \forall (w, w') \in \mathcal{I}, \ \forall t \in \mathcal{T}, s \in \mathcal{S}$$

$$y_{wt} \leq C_{wt} S_{wt}, \ \forall w \text{ s.t. } \exists \, w' \in \mathcal{W} \mid (w', w) \in \mathcal{H}$$

$$y_{wt} \leq C_{wt}(1 - F_{wt}), \ \forall w \text{ s.t. } \exists \, w' \in \mathcal{W} \mid (w, w') \in \mathcal{H}$$

$$S_{wt} \leq S_{w,t+1}, \ \forall t \in 2, ..., T,$$

$$F_{wt} \leq F_{w,t+1}, \ \forall t \in 2, ..., T$$

$$F_{wt} \geq S_{w't+1}, \ \forall t \in 2, ..., T, \ (w, w') \in \mathcal{H}$$

$$\sum_{t < T} \sum_{a \in \mathcal{A}_{wt}} x_a^{ks} + d_w^s \geq r_w, w \in \mathcal{W}_R, \forall s \in \mathcal{S}$$

$$y_{wt} \leq C_{wt}, \ \forall w \in \mathcal{W}, t \in \mathcal{T}$$

$$x_a^{ks} \geq 0, \ \forall a \in \mathcal{A}, k \in \mathcal{K}, s \in \mathcal{S}$$

$$y_{wt} \in \mathbb{N}_0, \ \forall w \in \mathcal{W}, t \in \mathcal{T}$$

$$S_{wt}, F_{wt} \in \{0, 1\}, \ \forall w \in \mathcal{W}, t \in \mathcal{T}$$

5.B Performance of scenario reduction methods with MC problem

Here, we examine the performance of the different scenario reduction methods when applied to the MC problem. We do this by examining the sampling error from the sets, done with the following procedure:

1. Generate M candidate decisions $y_m, m = 1, ..., M$.
2. For k trials:
   (a) For each y_m:
      i. Sample scenario set $\mathcal{S}_k$ using the chosen method, with scenarios ϕ_s and associated probabilities r_s, $sin\mathcal{S}_k$
      ii. Compute the error:

$$err_{km} = |\sum_{s\in\mathcal{S}_k} r_s Q(y_m;\phi_s) - \sum_{s\in\mathcal{S}} p_s Q(y_m;\phi_s)|$$

   (b) Calculate the average error over the candidates $e\bar{r}r_k = \frac{1}{M}\sum_{m=1}^{M} err_{km}$

This gives k average errors for each method.

We calculated the average errors for the three scenario reduction methods, with $M = 10$ candidate solutions, and $k = 10$ trials. The results from this are shown in Figure 5.4.5.

Chapter 6

Conclusions

The courier, express, and parcel delivery industry is a vast industry worldwide, with sequential sorting facilities being a key part of it. The sequential nature of these facilities means that determining the correct staff levels is especially important, to stop upstream delays propagating through the system. Previous models for staffing these facilities were either lacking sufficient detail, set on longer-term timescales, or did not allow for randomness in daily material volumes. The work in this thesis addresses these shortcomings.

Here we summarise the contributions and limitations of each chapter, and provide some final remarks.

Chapter 3 In Chapter 3, we outlined a deterministic model to determine a workplan for a day in the mail centre. This model is based on a network design model. However, additional complexity has been added in the form of indirect streams and tethered WAs. Sorting deadlines are dealt with by using a time-expanded network at much finer-grained time periods than previous work.

We applied the developed model to three months of data provided by the UKMC. This data gave information necessary to build the network, daily mail volumes for the different streams, and workplans suggested by the UKMC's staffing algorithm. We used

this data to build the network, optimise it for each day, and compare our workplans to those suggested by the UKMC. We then used the data to simulate different conditions (i.e. changes in the mail volumes) to see how the model would respond. We tested four different approaches - minimising each objective (maximum workers and changes) on their own, and lexicographically minimising each one with the other as a secondary objective.

As expected, we found that approaches that focused on a specific objective were the best performing on that objective. For example, the MMWT and Lex 1 approaches worked best regarding the maximum workers objective. However, we found that the Lex 2 approach consistently outperformed the UKMC staff levels on both objectives. When increasing the mail, we found that the proportional increase in required workers was less than the proportional increase in the mail. The number of workers required also increased as the proportion of parcels vs letters was increased. This may foreshadow future staffing decisions giving the long term trends of increasing parcels and decreasing letter volumes. Finally, we noticed different behaviours in the approaches when the time granularity is decreased. Lex 2 and MC showed a trade-off between the two objectives as time granularity decreases (finer granularity results in more changes, but fewer workers). Lex 1 and MMWT improves both objectives as the number of time periods decreases.

This contributes to the literature by demonstrating the value of using a network model for staffing in mail centres (and other sequential sorting facilities). This work also highlights the effects that the priorities of different objectives can have on different results. From a managerial perspective, decision makers can use our model to determine workplans for day-to-day operations.

A limitation of this work is the assumption of known mail volumes. While in keeping with the literature, this assumption proves to be unrealistic in practice.

Chapter 4 We addressed this limitation in 4, by developing a stochastic programming model of the mail centre. This model is an extension of the deterministic model from

Chapter 3. We showed the effectiveness of this model by comparing results from the model to results obtained from the EV solution, as well as EV solutions with increased demand. We also examined different weightings between the two objectives, to account for the differences in magnitude between them. Finally, we added more flexibility to the model, allowing for the workplan to be changed (for a penalty) in the second stage of the model.

When changing the weights between the objectives, we found that only a weighting of $\alpha = 0.99$ for the first stage objective, and 0.01 for the second stage objective appropriately balanced the two objectives. When applying these weights, we find two observations. We show that the stochastic programming model with as few as 3 scenarios resulted in better (i.e. lower cost) solutions than the EV solutions (and the gap between the two increased as the number of scenarios increased). We noticed that increasing m (the proportional increase in expected demand) did not uniformly result in better EV_m solutions, with a variety of values of m giving the best results for the different weekdays. However, the stochastic program model still out-peformed the EV solutions when 20 or more scenarios were chosen.

Finally, we allowed the model to move workers around the mail centre in each time period in the second stage. This, as expected, showed clear benefits in both the first stage (primarily if there were no penalties to these changes) and the second stage objectives. However, if no penalty was imposed on the second-stage changes, then the model made an impractical number of changes, suggesting a penalty of $\kappa = 0.5$ gave more appropriate results.

The main drawback of this approach was the additional computational burden from solving the stochastic programs. This is especially important when we appropriately weight the two objectives, where we see we need scenario sets of 20 scenarios or more to out-perform the EV_{60} or EV_{80} solutions. However, we note that the scenario sets in this chapter were chosen with random sampling. A more sophisticated technique could

result in better scenario sets, reducing the computational burden.

Chapter 5 We explored other methods of scenario reduction in Chapter 5. Here, we discussed previous work showing the advantage of using problem-based scenario reduction methods over distribution-based. However, we also recognised that many specific problem-based methods could be generalised under a unifying framework.

The framework consisted of three steps. Firstly, a number of candidate decisions were found. These could be identified in a number of ways, including heuristics or solving smaller stochastic programs (of various sizes). Secondly, a deterministic model was solved with each scenario and each candidate decision and the recourse was found, to transform the scenarios into the recourse space. Finally, the recourse space was used to perform scenario reduction with existing distribution-based methods.

We applied this framework to a number of different approaches to finding candidate solutions and reduction methods, and tested this on three problems. While there was no one approach or method which worked best in every situation, we observed that using multi-scenario candidates performed better than the other spaces for most methods and most problems.

A limitation of our work on scenario reduction is that the framework cannot be used for scenario generation methods. This is because we need the selected scenarios to be a subset of the full scenario set. This excludes the many current scenario set selection methods. Therefore, a direction for future research would be to modify the framework to include scenario generation methods.

Final remarks This work represents an important contribution to the literature. It both shows a practical solution to a real-world problem (demonstrated on real-world data) and advances methods for network models, stochastic programming and scenario reduction. The deterministic sequential sorting facility model has a new level of detail and time-granularity that gives more control over the staffing levels. This can be used

by shift managers to make lower cost (and more efficient) staffing decisions, as we have shown. Furthermore, our analysis also showed the solutions the model would give if various aspects of the mail volumes were to change, showing the versatility of the model.

The stochastic model extends this, accounting for the randomness in the mail volumes. If large variations in volumes occur between days, and accurate forecasts are unavailable (as we see for our mail centre example), this is an important addition to the model. We saw that (given an appropriate weighting between the objectives is set) stochastic programs with enough scenarios out-performed the deterministic models, showing the value of this in facilities where accurate forecasts are not available. In addition, allowing flexibility to the workplan in the second stage also results in improvements. This gives the shift managers even more tools to select appropriate staff levels.

Given that a minimum number of scenarios is needed to see the benefit of the stochastic programs, our scenario reduction framework is useful to determine better solutions at lower computational costs. The generality of the framework means that many existing (and future) scenario reduction methods can be improved by this, and many approaches can be combined in new ways. This could include extensions to methods we did not test here, such as the moment-matching method of Zhang et al. (2023), or the k-means-like method of Bertsimas and Mundru (2023). Furthermore, more sophisticated methods could be used to find candidate solutions. Currently, when stochastic programs are solved, the scenario sets are found by random sampling. It may be beneficial to use more sophisticated scenario generation techniques at this stage as well.

Bibliography

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1994). *Network flows: Theory, algorithms and applications*, volume 50. Elsevier B.V.

Andreatta, G., De Giovanni, L., and Monaci, M. (2014). A fast heuristic for airport ground-service equipment–and–staff allocation. *Procedia, Social and Behavioral Sciences*, 108:26–36.

April, J., Glover, F., Kelly, J. P., and Laguna, M. (2003). Practical introduction to simulation optimization. In *Proceedings of the 2003 Winter Simulation Conference, 2003*, volume 1, pages 71–78 Vol.1. IEEE.

Arpón, S., Homem-de Mello, T., and Pagnoncelli, B. (2018). Scenario reduction for stochastic programs with conditional value-at-risk. *Mathematical Programming*, 170(1):327–356.

Balachandran, K. (1977). Optimal design of a post office. *Omega*, 5(2):185–191.

Bard, J. F., Binici, C., and deSilva, A. H. (2003). Staff scheduling at the United States Postal Service. *Computers and Operations Research*, 30(5):745–771.

Bard, J. F., DeSilva, A., Feo, T. A., and Wert, S. D. (1993). Design of semi-automated mail processing facilities. *IIE Transactions*, 25(4):88–101.

Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

Bertsimas, D. and Mundru, N. (2023). Optimization-based scenario reduction for data-driven two-stage stochastic optimization. *Operations Research*, 71(4):1343–1361.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Brucker, P. and Knust, S. (2012). *Complex scheduling*. Springer Berlin Heidelberg.

Brunner, J. O., Bard, J. F., and Kolisch, R. (2009). Flexible shift scheduling of physicians. *Health Care Management Science*, 12(3):285–305.

Busaker, R. and Gowen, P. J. (1961). A procedure for determining minimal-cost network flow patterns. Technical Report 15, Operational Research Office, John Hopkins University.

Calfa, B., Agarwal, A., Grossmann, I., and Wassick, J. (2014). Data-driven multi-stage scenario tree generation via statistical property and distribution matching. *Computers & Chemical Engineering*, 68:7–23.

Charnes, A. and Cooper, W. (1959). Chance-constrained programming. *Management Science*, 6(1):73–79.

Crainic, T. G., Rei, W., Hewitt, M., and Maggioni, F. (2016). Partial benders decomposition strategies for two-stage stochastic integer programs. Technical Report 37, CIRRELT.

Cruz-Mejía, O. and Letchford, A. N. (2023). A survey on exact algorithms for the maximum flow and minimum-cost flow problems. *Networks*, 82(2):167–176.

Dantzig, G. and Wolfe, P. (1961). The decomposition algorithm for linear programs. *Econometrica*, 29(4):767.

Dantzig, G. B. (1954). A comment on Edie's "Traffic delays at toll booths". *Journal of the Operations Research Society of America*, 2(3):339–341.

Dantzig, G. B. (1955). Linear programming under uncertainty. *Management Science*, 1(3-4):197–206.

Dantzig, G. B. and Glynn, P. W. (1990). Parallel processors for planning under uncertainty. *Annals of Operations Research*, 22(1):1–21.

De Bruecker, P., Van den Bergh, J., Beliën, J., and Demeulemeester, E. (2015). Workforce planning incorporating skills: State of the art. *European Journal of Operational Research*, 243(1):1–16.

Devesse, V. A. P. A., Akartunalı, K., Arantes, M., and Toledo, C. F. M. (2022). Linear approximations to improve lower bounds of a physician scheduling problem in emergency rooms. *Journal of the Operational Research Society*, 74(3):888–904.

Dorneles, A. P., de Araujo, O. C., and Buriol, L. S. (2017). A column generation approach to high school timetabling modeled as a multicommodity flow problem. *European Journal of Operational Research*, 256(3):685–695.

Dupacova, J., Growe-Kuska, N., and Romisch, W. (2003). Scenario reduction in stochastic programming an approach using probability metrics. *Mathematical Programming*, 95(3):493–511.

Easwaran, G. and Uster, H. (2009). Tabu search and Benders decomposition approaches for a capacitated closed-loop supply chain network design problem. *Transportation Science*, 43(3):301–320.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., Owens, B., and Sier, D. (2004). An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127(1-4):21–144.

Fairbrother, J., Turner, A., and Wallace, S. W. (2022). Problem-driven scenario generation: an analytical approach for stochastic programs with tail risk measure. *Mathematical Programming*, 191(1):141–182.

Feng, Y. and Ryan, S. M. (2016). Solution sensitivity-based scenario reduction for stochastic unit commitment. *Computational Management Science*, 13(1):29–62.

Fischer, F. and Helmberg, C. (2014). Dynamic graph generation for the shortest path problem in time expanded networks. *Mathematical Programming*, 143(1-2):257–297.

Fisher, M. L. (1985). An applications oriented guide to Lagrangian relaxation. *Interfaces*, 15(2):10–21.

Ford, L. and Fulkerson, D. (1958). Constructing maximal dynamic flows from static flows. *Operations Research*, 6(3):419–433.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin (New Series) of the American Mathematical Society*, 64(5):275–278.

Growe-Kuska, N., Heitsch, H., and Romisch, W. (2003). Scenario reduction and scenario tree construction for power management problems. In *2003 IEEE Bologna Power Tech Conference Proceedings*, volume 3, pages 7 pp. Vol.3–158. IEEE.

Gurobi Optimization, LLC (2022). Gurobi Optimizer Reference Manual.

Heitsch, H. and Römisch, W. (2003). Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, 24(2-3):187–206.

Heitsch, H. and Römisch, W. (2007). A note on scenario reduction for two-stage stochastic programs. *Operations Research Letters*, 35(6):731–738.

Heitsch, H. and Römisch, W. (2009). Scenario tree reduction for multistage stochastic programs. *Computational Management Science*, 6(2):117–133.

Hewitt, M., Ortmann, J., and Rei, W. (2022). Decision-based scenario clustering for decision-making under uncertainty. *Annals of Operations Research*, 315(2):747–771.

Higle, J. L. (1998). Variance reduction and objective function evaluation in stochastic linear programs. *INFORMS Journal on Computing*, 10(2):236–247.

Hong, L. J., Yang, Y., and Zhang, L. (2011). Sequential convex approximations to joint chance constrained programs: A Monte Carlo approach. *Operations Research*, 59(3):617–630.

Houle, M. E., Kriegel, H.-P., Kröger, P., Schubert, E., and Zimek, A. (2010). Can shared-neighbor distances defeat the curse of dimensionality? In *Scientific and Statistical Database Management*, Lecture Notes in Computer Science, pages 482–500. Springer Berlin Heidelberg, Berlin, Heidelberg.

Høyland, K., Kaut, M., and Wallace, S. (2003). A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24:169–185.

Høyland, K. and Wallace, S. W. (2001). Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307.

Infanger, G. (1992). Monte Carlo (importance) sampling within a Benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research*, 39(1):69–95.

Jarrah, A. I., Qi, X., and Bard, J. F. (2016). The destination-loader-door assignment problem for automated package sorting centers. *Transportation Science*, 50(4):1314–1336.

Jarrah, A. I. Z., Bard, J. F., and deSilva, A. H. (1994a). Equipment selection and machine scheduling in general mail facilities. *Management Science*, 40(8):1049–1068.

Jarrah, A. I. Z., Bard, J. F., and deSilva, A. H. (1994b). Solving large-scale tour scheduling problems. *Management Science*, 40(9):1124–1144.

Jobst, N. J. and Zenios, S. A. (2003). Tracking bond indices in an integrated market and credit risk environment. *Quantitative Finance*, 3(2):117–135.

Kabiru, S., Saidu, B. M., Abdul, A. Z., and Ali, U. A. (2017). An optimal assignment schedule of staff-subject allocation. *Journal of Mathematical Finance*, 07(04):805–820.

Kammammettu, S. and Li, Z. (2023). Scenario reduction and scenario tree generation for stochastic programming using sinkhorn distance. *Computers & Chemical Engineering*, 170:108122.

Kaufman, L. (1990). Partitioning around medoids (program PAM). *Finding Groups in Data*, 344:68–125.

Kaut, M. and Lium, A.-G. (2014). Scenario generation with distribution functions and correlations. *Kybernetika*, 50(6):1049–1064.

Kaut, M. and Wallace, S. (2007). Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, 3(2):257–271.

Kaut, M. and Wallace, S. W. (2011). Shape-based scenario generation using copulas. *Computational Management Science*, 8(1–2):181–199.

Keutchayan, J., Ortmann, J., and Rei, W. (2023). Problem-driven scenario clustering in stochastic optimization. *Computational Management Science*, 20(1):13.

King, A. J. and Wallace, S. W. (2012). *Modeling with stochastic programming*. Springer Science & Business Media.

Klein, M. (1967). A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3):205–220.

Klinz, B. and Woeginger, G. J. (2004). Minimum-cost dynamic flows: The series-parallel case. *Networks*, 43(3):153–162.

Kovács, P. (2015). Minimum-cost flow algorithms: An experimental evaluation. *Optimization Methods and Software*, 30(1):94–127.

Land, A. and Doig, A. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497.

Lei, Q., Zhao, P., Jiang, D., and Ma, T. (2013). Research on the disrupted airline scheduling. In *2013 10th International Conference on Service Systems and Service Management*, pages 332–336. IEEE.

Li, Y., Gao, X., Xu, Z., and Zhou, X. (2018). Network-based queuing model for simulating passenger throughput at an airport security checkpoint. *Journal of Air Transport Management*, 66:13–24.

Lin, H. and Uster, H. (2014). Exact and heuristic algorithms for data-gathering cluster-based wireless sensor network design problem. *IEEE/ACM Transactions on Networking*, 22(3):903 – 916.

Linderoth, J., Shapiro, A., and Wright, S. (2006). The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241.

Louly, M. (2013). A goal programming model for staff scheduling at a telecommunications center. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 12(2):167–178.

Madansky, A. (1960). Inequalities for stochastic linear programming problems. *Management Science*, 6(2):197–204.

Mamer, J. W. and McBride, R. D. (2000). A decomposition-based pricing procedure

for large-scale linear programs: An application to the linear multicommodity flow problem. *Management Science*, 46(5):693–709.

Marin, A. G. and Jaramillo, P. (2009). Urban rapid transit network design: Accelerated benders decomposition. *Annals of Operations Research*, 169(1):35 – 53.

Narum, B. (2020). Problem-based scenario generation in stochastic programming with binary distributions: Case study in air traffic flow management. [Unpublished master's thesis, MSc, Norwegian University of Science and Technology].

Narum, B. S., Fairbrother, J., and Wallace, S. W. (2024). Problem-based scenario generation by decomposing output distributions. *European Journal of Operational Research*, Forthcoming.

Ni, H. and Abeledo, H. (2007). A branch-and-price approach for large-scale employee tour scheduling problems. *Annals of Operations Research*, 155(1):167–176.

Office of Communications (2012). Communications market report 2012. Technical report, Office of Communications.

Orlin, J. B., Plotkin, S. A., and Tardos, E. (1993). Polynomial dual network simplex algorithms. *Mathematical Programming*, 60(3):255–276.

Paias, A., Mesquita, M., Moz, M., and Pato, M. (2021). A network flow-based algorithm for bus driver rerostering. *OR Spectrum*, 43(2):543–576.

Papavasiliou, A. and Oren, S. S. (2013). Multiarea stochastic unit commitment for high wind penetration in a transmission constrained network. *Operations Research*, 61(3):578–592.

Pflug, G. C. (2001). Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89(2):251–271.

Pishvaee, M., Razmi, J., and Torabi, S. (2014). An accelerated Benders decomposition algorithm for sustainable supply chain network design under uncertainty: A case study of medical needle and syringe supply chain. *Transportation Research Part E: Logistics and Transportation Review*, 67:14–38.

Ponomareva, K., Roman, D., and Date, P. (2015). An algorithm for moment-matching scenario generation with application to financial portfolio optimisation. *European Journal of Operational Research*, 240(3):678–687.

Powell, W. B. (2011). *Approximate dynamic programming solving the curses of dimensionality*. Wiley series in probability and statistics. J. Wiley & Sons, Hoboken, N.J., 2nd ed. edition.

PR Newswire (2023). Global courier, express and parcel industry research report 2023-2028: CEP providers forge strategic partnerships to take advantage of online gifting trends and WFH policies. *PR Newswire*.

Prochazka, V. and Wallace, S. W. (2018). Stochastic programs with binary distributions: structural properties of scenario trees and algorithms. *Computational Management Science*, 15(3-4):397–410.

Prochazka, V. and Wallace, S. W. (2020). Scenario tree construction driven by heuristic solutions of the optimization problem. *Computational Management Science*, 17(2):277–307.

Qi, X. and Bard, J. F. (2006). Generating labor requirements and rosters for mail handlers using simulation and optimization. *Computers & Operations Research*, 33(9):2645 – 2666.

Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817.

Restrepo, M. I., Lozano, L., and Medaglia, A. L. (2012). Constrained network-based column generation for the multi-activity shift scheduling problem. *International Journal of Production Economics*, 140(1):466–472.

Royal Mail (2019). Annual Report and Financial Statements 2018-19. Technical report, Royal Mail.

Saharidis, G. K., Boile, M., and Theofanis, S. (2011). Initialization of the Benders master problem using valid inequalities applied to fixed-charge network problems. *Expert Systems with Applications*, 38(6):6627–6636.

Sen, S., Doverspike, R. D., and Cosares, S. (1994). Network planning with random demand. *Telecommunication Systems*, 3(1):11–30.

Skaugset, L. M., Farrell, S., Carney, M., Wolff, M., Santen, S. A., Perry, M., and Cico, S. J. (2016). Can you multitask? Evidence and limitations of task switching and multitasking in emergency medicine. *Annals of Emergency Medicine*, 68(2):189–195.

Skutella, M. (2009). An introduction to network flows over time. In *Research trends in combinatorial optimization*, pages 451–482. Springer.

Song, Q., Wang, X., Li, X., and Zhang, C. (2009). Optimization of postal express mail network based on swarm intelligence. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 591–596. IEEE.

Sun, M., Teng, F., Konstantelos, I., and Strbac, G. (2018). An objective-based scenario selection method for transmission network expansion planning with multivariate stochasticity in load and renewable energy sources. *Energy (Oxford)*, 145:871–885.

Sutien, K., Makackas, D., and Pranevičius, H. (2010). Multistage k-means clustering for scenario tree construction. *Informatica (Vilnius, Lithuania)*, 21(1):123–138.

Tapak, P., Kulluk, S., Özbakır, L., Bahar, F., and Gülmez, B. (2023). A constraint programming based column generation approach for crew scheduling: A case study for the Kayseri railway. *Journal of the Operational Research Society*, 74(9):2028–2042.

Tello, F., Jiménez-Martín, A., Mateos, A., and Lozano, P. (2019). A comparative analysis of simulated annealing and variable neighborhood search in the ATCo work-shift scheduling problem. *Mathematics*, 7(7).

Thorburn, H., Sachs, A.-L., Fairbrother, J., and Boylan, J. E. (2023). A time-expanded network design model for staff allocation in mail centres. *Journal of the Operational Research Society*, pages 1–16.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., and De Boeck, L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385.

Van Slyke, R. M. and Wets, R. (1969). L -shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663.

Vieira, C. L. D. S., Luna, M. M. M., and Azevedo, J. M. (2019). Minimum-cost flow algorithms: a performance evaluation using the Brazilian road network. *World Review of Intermodal Transportation Research*, 8(1):3–21.

Villani, C. (2003). *Topics in optimal transportation*. American Mathematical Society.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

Xie, L. and Suhl, L. (2015). Cyclic and non-cyclic crew rostering problems in public bus transit. *OR Spectrum*, 37(1):99–136.

Zhang, W., Wang, K., Jacquillat, A., and Wang, S. (2023). Optimized scenario reduction: Solving large-scale stochastic programs with quality guarantees. *INFORMS Journal on Computing*, 35(4):886–908.

Zhang, X. and Bard, J. F. (2005). Equipment scheduling at mail processing and distribution centers. *IIE Transactions*, 37(2):175–187.

Zhang, X., Chakravarthy, A., and Gu, Q. (2009). Equipment scheduling problem under disruptions in mail processing and distribution centres. *Journal of the Operational Research Society*, 60(5):598–610.