University of Southampton

Faculty of Social Sciences
School of Mathematical Sciences

Robust Optimisation in Network Revenue Management

by

Simos Zachariades
MSc, BSc
ORCiD: 0000-0002-1916-0018

A thesis for the degree of Doctor of Philosophy

July 2022

University of Southampton

Abstract

Faculty of Social Sciences
School of Mathematical Sciences

Doctor of Philosophy

Robust Optimisation in Network Revenue Management

by Simos Zachariades

Network revenue management is used extensively, particularly within the airline industry, to allocate dependent resources between different products. This work focuses on the situation where demand is uncertain and the aim is to determine booking limits that are robust to fluctuations in demand. Expanding on the work of Perakis and Roels (2010), we developed a genetic algorithm that finds booking limits that either minimize the maximum regret or maximize the minimum revenue for a number of different booking control policies: partitioned booking limits, nested booking limits and bid prices. We present results that demonstrate how these booking limits outperform those obtained via local descent methods and other traditional network models. Furthermore, we consider the uncertainty set for demand to be ellipsoidal further to the polyhedral as originally proposed. Finally, we introduce the formulation on network cruise revenue management application. We present the robust formulation for the cruise network setting and present numerical results that show that the robust control measures outperform standard approximation methods.

Contents

List of Figures

List of Tables

Nomenclature

Indices

i	A network resource i such that $i = 1, ..., m$
j	A product j such that $j = 1, ..., n$
t	A time period such that $t = 1, ..., T$

Functions

$r(v, w)$	A real valued function defined on $V \times W$ specifying the payoffs associated with the decisions V and states W under consideration
$R(v, w)$	A real valued function defined on $V \times W$ specifying the regret associated with decision v and state w
$V_t(\boldsymbol{c})$	The maximum expected revenue to go, given remaining capacity $\boldsymbol{c}$ in period t
$V_t^M(\boldsymbol{c})$	An approximation method M that yields an estimate to $V_t(\boldsymbol{c})$

Vectors

$\boldsymbol{f}$	Revenue coefficients vector for each product j such that $\boldsymbol{f} = [f_j \mid j = 1, ..., n]$
$\boldsymbol{q}^\pi$	Accepted booking requests vector when policy π is in use such that $\boldsymbol{q}^\pi = [q_j^\pi \mid j = 1, ..., n]$
$\boldsymbol{x}$	Booking limits decision vector such that $\boldsymbol{x} = [x_j \mid j = 1, ..., n]$
$\boldsymbol{c}$	Capacity vector for resources i such that $\boldsymbol{c} = [c_i \mid i = 1, ..., m]$
$\boldsymbol{d}$	Total observed demand vector such that $\boldsymbol{d} = [d_j \mid j = 1, ..., n]$
$\boldsymbol{p}$	Vector of partitioned protection levels for products such that $\boldsymbol{p} = [p_j \mid j = 1, ..., n]$
$\boldsymbol{b}$	Vector of nested booking limits for products such that $\boldsymbol{b} = [b_s \mid s = j, ..., n$
$\boldsymbol{y}$	Vector of dual prices such that $\boldsymbol{y} = [y_j \mid j = 1, ..., n]$

$\boldsymbol{P}(t)$	Vector of request for products $j = 1, ..., n$ at time t. If $P_j(t) > 0$ then a request for product j at time t has occurred and $P_j(t) = 0$ otherwise

Matrices

$\boldsymbol{A}$	Incidence matrix, $[a_{ij}]$, denoting the network resources a product requires such that $a_{ij} = 1$ when product j requires resource i and $a_{ij} = 0$ otherwise
$\boldsymbol{B}$	Occupancy matrix, $[b_{ij}]$, denoting the number of lifeboat seats a product j requires such that $b_{ij} = o_j$ where o_j is the occupancy under product j that utilises resource i of the network and 0 otherwise

Sets

$\mathcal{B}$	The set of all booking limits $\boldsymbol{b}$, such that $\boldsymbol{b} \in \mathcal{B}$
$\mathcal{D}$	The set of all demand realisations $\boldsymbol{d}$, such that $\boldsymbol{d} \in \mathcal{D}$
$\mathcal{J}$	The set of all products j such that $j \in \mathcal{J}$
$\mathcal{K}$	The set containing all the different cabin types on a cruise itinerary such that $k \in \mathcal{K}$
$\mathcal{L}$	The set containing all the possible legs l in a cruise route where a leg starts at a port of embarkation and ends in a port of debarkation such that $l \in \mathcal{L}$
$\mathcal{O}$	The set of different occupancy levels o available in a cruise cabin such that $o \in \mathcal{O}$
$\mathcal{S}$	The set of all buckets of products s, such that $s \in \mathcal{S}$
$\mathcal{U}$	An uncertainty set
$\mathcal{X}$	The set containing all the possible itineraries in a cruise network where an itinerary can span a single to multiple legs of the network such that $\chi \in \mathcal{X}$
$\mathcal{Z}$	The set of different fare price tiers available for a cruise product such that $\zeta \in \mathcal{Z}$
Π	The set of non-anticipating booking policies π such that $\pi \in \Pi$
V	The set containing all the possible alternative decisions available to the decision maker, the decision space.
W	The set containing all the possible states under consideration, the state space.

Genetic Algorithm

t	An iteration of the algorithm
g	A gene, an element of a chromosome
$\boldsymbol{c}$	A column vector representing a chromosome such that $\boldsymbol{c} = [g_j \mid j = 1, ..., n]$

$\mathcal{G}_t$	The set of of chromosomes such that $\mathcal{G}_t = \{\boldsymbol{c}_i \mid i = 1, ..., N\}$
$f(\boldsymbol{c}_i)$	A real valued function specifying the fitness score associated with the chromosome $\boldsymbol{c}_i$
p_c	The proportion of new chromosome to be created by the new crossover operator
m_c	The proportion of new chromosome to be created by the new crossover operator

Declaration of Authorship

I, Simos Zachariades, declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. None of this work has been published before submission

Signed:.. Date:..................

Acknowledgements

With tremendous gratitude, I would like to express my gratefulness to all the people who have supported me in any capacity over all these years to make this thesis possible.

First and foremost, I owe deep gratitude to my supervisors, Joerg and especially Christine. You were responsible for my first encounter with operational research back in my bachelor degree days and then inspired me to continue down a research path during my Master. Your continuous and unwavering support and guidance have been of paramount importance throughout these years. Your endless patience and encouragement drove me forward and have been the solid pillar of this project, which would not have been possible without you!

I also want to thank my research colleagues for the great environment in the office, which made my workdays fun these past years! Ruthy, Martina and Alex, it was a pleasure sharing this journey with you. I could not have hoped to embark on this journey with a better company! Your drive, energy and passion inspired me so many times, and I will always cherish our time together! I have been to a few offices since, but none compares to La Famiglia. A big thank you to Marton, Walton, Karl, and Laura for the countless tea breaks with crossword puzzles, random discussions about anything and everything, the fun NBA fantasy competitions that still go on, and every other fun activity we've shared! And, of course, my Cypriot friends, colleagues and roommates! Andria, I deeply appreciate your friendship during the whole PhD journey, Damiane, thank you for being a great roommate and a person to discuss anything with, including research. Fani and Sofocle, thank you for making my days in Southampton remind me more of home. And Stephanie, thank you for helping me push through at the very end. Our study sessions were of paramount importance to finishing this thesis!

And, of course, to my family, my parents and brothers, I am forever grateful for all of your sacrifices and your unwavering belief in me. I couldn't have completed this journey without your support. Thank you!

Last but not least, I want to wholeheartedly thank Tania, who is the ever calm presence by my side. Your love and patience pushed me forward every step of the way. You kept up with me being abroad for so many years, and you were my solid rock when things did not go my way. You are more than I have ever hoped for. Thank you for making me better.

To my friends and family

Chapter 1

Introduction

1.1 Background and Motivation

Revenue management originates in the post-deregulation era of the US airline industry when a handful of competing airlines aimed to maximise their revenue by altering the seat allocation or prices of the different products they offered. Parallel to the innate capacity of the aviation industry for revenue management, academia embraced the field and helped to advance the methodologies and practices implemented considerably. Academics across operational research, statistics, economics, and computer science are actively engaged in revenue management research. Today, the field has risen to a mainstream business methodology practised by a growing list of industries, ranging from transportation and hospitality to sport and music. Its effect on business is highlighted by Talluri and Van Ryzin (2005) who report that "the economic impact of revenue management is significant, with increases in revenue of 5% or more reported in several industry applications". The most common characteristics of revenue management were summarised by Kimes (1989) as follows:

1. Fixed capacity
2. Perishable inventory
3. Market/Customer segmentation
4. Advanced sales/reservations
5. Time-variable demand
6. Low marginal costs

In this thesis, we focus our investigation on the quantity-based revenue management for multiple resources problem, commonly referred to as network revenue management (NRM). The aim is to optimally allocate the capacities of multiple resources to the different classes of demand.

The reason this problem is of interest is threefold. First, networks arise naturally in some of the world's most profitable industries. The airline, railway and cruise industries, hotel long stays, and car rentals are a few applications that can be modelled as network revenue management problems. Optimising these networks based on mathematical models and data science can lead to a significant increase in revenue.

Secondly, the problem presents a mathematical and logistical challenge of its own accord. The curse of dimensionality, the complexity of the network and the difficulty in accurately forecasting demand for the numerous products on offer deem the problem extremely challenging. Talluri and Van Ryzin (2004) even stated that "In the network case, exact optimisation is for all practical purposes impossible". Hence the use of heuristics or approximation models is necessary to solve the problem in a reasonable time.

Thirdly, the emergence of research into robust optimisation and its applications to revenue management result in some interesting research questions. For example, determining the trade-off between expected revenue and risk in a network setting, quantifying and optimising risk in networks, and determining the appropriate performance measures for robust network revenue management models to determine how they compare with each other and traditional risk-neutral approaches.

Mathematically, the solution to the problem is to find a booking policy, $\pi \in \Pi$ where Π is the set of non-anticipating booking policies, that will maximise the expected revenue. the policy π is assumed to be non-anticipating; that is, any decisions under policy π are taken only using information gained up to the time t when the decision is taken. Let $\boldsymbol{f} \in \mathbb{R}^n$ be the revenue coefficients vector where n is the number of products in offer while $\boldsymbol{q}^\pi \in \mathbb{R}^n$ is the vector of accepted booking requests when policy π is in use. Let $\boldsymbol{A}$ be the matrix that indicates how each of the products uses the resources' capacities, $\boldsymbol{c}$, in the network. That is $\boldsymbol{A}\boldsymbol{q}^\pi \leq \boldsymbol{c}$ while the number of accepted requests must be positive and less than the total observed demand, $\boldsymbol{d}$, that is $0 \leq q^\pi \leq \boldsymbol{d}$. Putting this together, we can formulate the problem as,

$$\begin{aligned} \sup_{\pi \in \Pi} \quad & \boldsymbol{f}'\mathbb{E}[\boldsymbol{q}^\pi] \\ \text{s.t} \quad & \boldsymbol{A}\boldsymbol{q}^\pi \leq \boldsymbol{c} \\ & 0 \leq \boldsymbol{q}^\pi \leq \boldsymbol{d} \end{aligned} \tag{1.1}$$

The data used to construct the collection $(\boldsymbol{f}, \boldsymbol{A}, \boldsymbol{c}, \boldsymbol{d})$ are often uncertain. Data might not be known exactly. They can be given a nominal figure that drifts around their true value or follow a random process. Data uncertainty results from several reasons, most commonly:

1. Forecast entries (e.g. future demands, returns, etc.) are subject to prediction errors.
2. Data are subject to measurement errors.
3. Decision variables that cannot be exactly implemented as computed are subject to implementation errors.

These sources of uncertainty are particularly apparent in the context of revenue management. There are two main sources of uncertainty in revenue management systems. Firstly, it is the inherent uncertainty due to the arrival of customers into the system being a random process. Future demand for products is uncertain, and it is usually treated in one of two distinct ways; *Dynamic* models assume a stochastic process for the arrival of customers, and *Static* models assign a probability distribution to the aggregate number of future customers. Both methods heavily rely on historical data and forecasting for their effectiveness.

This leads to the second source of uncertainty. Errors in the data entries and measurements, systematic changes in behaviour, bias in reporting and poor forecast models yield the risk of data uncertainty. In applications where rich historical data exist, more sophisticated quantitative forecasting methods can be employed, and their results can be relied on with increased confidence but are still uncertain. On the other hand, new or non-stationary applications often apply simple forecasting techniques that are, by their nature, associated with large error margins. Since accurate forecasting is vital to RM models, there is a need to protect such models against their inherent data uncertainty.

While in traditional Linear Optimisation (LO) methodology, it is often assumed that small data uncertainties do not significantly affect the feasibility or optimality of the solution, Ben-Tal and Nemirovski (2000) challenged and proved this notion wrong. Using examples, they show that even small data uncertainties can significantly distort the optimal solution from its nominal value. At the same time, Bertsimas et al. (2011) arrive at a conclusion that "In applications of LO, there exists a need of a technique capable of detecting cases when data uncertainty can heavily affect the quality of the nominal solution, and in these cases to generate a 'reliable' solution, one that is immunised against uncertainty." To that end the *Robust Optimisation* (RO) methodology has been developed where the data collection $(\boldsymbol{f}, \boldsymbol{A}, \boldsymbol{c}, \boldsymbol{d})$ is considered uncertain and belongs to an uncertainty set $\mathcal{U}$ and thus the uncertain mathematical program solved is:

$$\begin{aligned} \text{maximise} \quad & \boldsymbol{f}'\boldsymbol{q}^{\pi} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{q}^{\pi} \leq \boldsymbol{c} \\ & 0 \leq \boldsymbol{q}^{\pi} \leq \boldsymbol{d} \\ & (\boldsymbol{f}, \boldsymbol{A}, \boldsymbol{c}, \boldsymbol{d}) \in \mathcal{U} \end{aligned} \tag{1.2}$$

In this thesis, we are only concerned with the uncertainty in vector $\boldsymbol{d}$, the right-hand side of problem 1.2, as we can not know the demand to be realised for products in the future, but we can only estimate it. On the other hand, we do not consider $\boldsymbol{A}$, $\boldsymbol{c}$ as uncertain as in the settings we investigate, airline and cruise networks, changes in the number of resources a product consumes and the capacity of the vessels are fixed and predetermined. Thus we can safely assume that both $\boldsymbol{A}$ and $\boldsymbol{c}$ are certain. Lastly, the treatment of vector $\boldsymbol{f}$ is of higher interest as we only *assume* to be certain in our setting. In reality, the fares of products can and are often changed during the booking horizon, a practice known as dynamic pricing. Hence the problem we are concerned with is:

$$\begin{aligned} \text{maximise} \quad & \boldsymbol{f}'\boldsymbol{q}^{\pi} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{q}^{\pi} \leq \boldsymbol{c} \\ & 0 \leq \boldsymbol{q}^{\pi} \leq \boldsymbol{d} \\ & \boldsymbol{d} \in \mathcal{U} \end{aligned} \tag{1.3}$$

Of course as an extension to our treatment, one can model vector $\boldsymbol{f}$ as uncertain and construct its associated $\mathcal{U}$ from the historical data of prices set to products. Such treatment would also introduce uncertainty in the objective function of the problem rather than only to the right-hand side. In such a case, one could rewrite problem 1.2 using the epigraph form to migrate the objective function uncertainty to the left-hand side of the problem but also introducing an extra variable, $\epsilon \in \mathbb{R}$, to optimise over:

$$\begin{aligned} \text{maximise} \quad & \epsilon \\ \text{subject to} \quad & \boldsymbol{f}'\boldsymbol{q}^{\pi} - \epsilon \geq 0 \\ & \boldsymbol{A}\boldsymbol{q}^{\pi} \leq \boldsymbol{c} \\ & 0 \leq \boldsymbol{q}^{\pi} \leq \boldsymbol{d} \\ & (\boldsymbol{f}, \boldsymbol{d}) \in \mathcal{U} \end{aligned} \tag{1.4}$$

It is clear that $(\boldsymbol{q}^{\pi}, t)$ is optimal for 1.4 if and only if $\boldsymbol{q}^{\pi}$ is optimal for 1.3 and $\epsilon = \boldsymbol{f}'\boldsymbol{q}^{\pi}$.

Nonetheless, the new problem 1.3 under discussion, is about deciding on a reasonable course of action on the basis of incomplete information. An action for a decision maker is choosing the $\boldsymbol{q}^{\pi}$. Any action the decision maker takes has an associated consequence, in this case the revenue to be generated from the choice of under $\boldsymbol{q}^{\pi}$ the uncertainty of $\boldsymbol{d}$. We refer to the unknown revenue to be generated as a *state*.

In the Robust decision framework there are two main approaches to dealing with this uncertainty. The maximin revenue and the minimax regret principles. These are explained in finer detail in sections 2.5.1.1 and 2.5.1.2 respectively. The maximin revenue principle selects the booking policy limits that maximise the minimum revenue generated by the worst demand realisation under consideration,

$$\phi^* = \max_{\boldsymbol{b} \in \boldsymbol{\mathcal{B}}} \min_{\boldsymbol{d} \in \mathcal{D}(\boldsymbol{b})} \boldsymbol{f}(\boldsymbol{b}, \boldsymbol{d}) \tag{1.5}$$

The rationale of the model is to guarantee a minimum revenue over all future demand realisations. This approach only considers the worst-case scenarios and thus is extremely risk averse and may not perform well on average. Hence it often leads to conservative decisions where the trade off of robustness against revenue is heavily skewed towards robustness.

To overcome the overly conservative results of the maximin revenue principle, Savage (1951) introduced the minimax regret principle. Under this principle the decision maker chooses the decision x that minimises the maximum regret function over all possible states. The regret, $R(\boldsymbol{b}, \boldsymbol{d})$, is defined as the difference between the payoff of the best booking limits we could have chosen for demand realisation $\boldsymbol{d}$ and the actual booking limits we have chosen for the demand realisation $\boldsymbol{d}$. Hence the maximum regret is the additional revenue that could have been obtained with perfect information over all demand processes under consideration and mathematically is given as,

$$\rho = \max_{\boldsymbol{d} \in \mathcal{D}} R(\boldsymbol{b}, d) \tag{1.6}$$

The aim of the decision maker is to minimise the regret associated with demand realisation d,

$$\begin{aligned} \rho^* &= \min_{\boldsymbol{b} \in \mathcal{B}} \max_{\boldsymbol{d} \in \mathcal{D}} R(\boldsymbol{b}, \boldsymbol{d}) \\ &= \min_{\boldsymbol{b} \in \mathcal{B}} \max_{\boldsymbol{d} \in \mathcal{D}} \{\max_{\boldsymbol{b}^*} r(\boldsymbol{b}^*, \boldsymbol{d}) - r(\boldsymbol{b}, \boldsymbol{d})\} \end{aligned} \tag{1.7}$$

Both of these equations have an inner and outer optimisation problem to solve. What we refer to as the inner optimisation, is the quantification of the minimum revenue ϕ and maximum regret ρ. We quantify these robust measures by employing the MILP formulation of Perakis and Roels (2010) which is presented in section 3.2.1. What we refer to as the outer optimisation problem is altering the booking limits repeatedly to find the booking limits that yield the best robust measure value. The outer problem is usually solved by a heuristic method. Perakis and Roels (2010) use a gradient descent algorithm, a brief overview is given in section 2.6, while we employ a genetic algorithm for the task.

1.2 Aims and Contributions

This research project develops and tests algorithms that find robust solutions to network revenue management problems. Initially we extend the algorithms developed by Perakis and Roels (2010) for an airline network, before going on to develop robust optimisation algorithms for cruise networks. While working towards achieving our objectives, we make the following contributions, which we present in the different chapters of this document.

In our Literature Review chapter, we collect a significant body of work in revenue management and identify the recent trend of research shifting away from traditional risk-neutral methodologies towards risk-averse and robust modelling formulations. We identify two areas of *future research*: a) *robust customer-choice* and b) *robust/risk-averse network revenue management*. The latter forms the focus of the remainder of the thesis.

In Chapter 3 we introduce a comprehensive and universal mathematical notation for all the booking control policies, seat-allocation models, booking acceptance algorithms and demand simulation models. We expand the methodology in the literature by providing a more thorough explanation of the robust seat allocation models formulation and introducing ellipsoidal uncertainty sets. Using an ellipsoidal uncertainty set introduces non-linearity to the problem formulation and consequently increases the computation time. Furthermore, we suggest a genetic algorithm to optimise the outer problem.

In Chapter 4 we present numerical results and compare them to Perakis and Roels (2010) work which initially introduced the robust seat allocation models we built upon. We show that our genetic algorithm performs better in simulated revenue than their local gradient algorithm. Furthermore, in a similar fashion to de Boer et al. (2002) who compared two traditional seat-allocation models, we built a comparison of five seat-allocation models, of which two are the robust optimisation algorithms we develop in Chapter 3. We implement three different booking control policies and simulate the booking horizon to evaluate the performance of each model under these policies. We also report our models' summary statistics on the simulated revenue. To conclude the chapter, we compare our results and discuss the limitations of our study.

In Chapter 5 we make a twofold contribution. First, we introduce a compact cruise network capacity control formulation. Our approach is novel because, in the available cruise literature, authors present multi-resource capacity control formulations where the constraints considered are built on lifeboat seats and available cabins of each type and the number of passengers per individual cabin. In our treatment, we further consider multiple ports of embarkation and debarkation, a cruise network built on multiple legs of travel. As far as we are aware, our treatment is a novel formulation because of this extra consideration. Secondly, we robustify the introduced network formulation by introducing polyhedral uncertainty sets, solve the resulting problems by modifying our genetic algorithm to treat both the cabin and lifeboat constraints, and present a simulation study of the revenues for the different models developed.

Table 1.1 summarises the aims and contributions of this research project up to this time.

Chapter		Description
Literature Review	Aim	Acquire a comprehensive knowledge regarding the theory, methodology and models
	Contribution	Identify gaps in literature
Robust Capacity Control	Aim	Apply practically the acquired knowledge of methodologies and models
	Contribution	Improve the exposition of model formulations.
	Contribution	Introduce ellipsoidal uncertainty to robust formulations
	Contribution	Implement a genetic algorithm tailored to the robust formulations that outperforms previous heuristics results
Numerical Results	Aim	Evaluate current methodologies
	Contribution	Present an extended simulation study of revenues for the several booking policies investigated
	Contribution	Identify limitations of deployed methodologies and propose future work
Cruise Capacity Control	Aim	Apply practically the acquired knowledge of methodologies and models
	Contribution	Present a compact mathematical description of the *network* problem.
	Contribution	Present a mathematical formulation of the *robust* network problem.

TABLE 1.1: Research project aims and contributions

1.3 Outline

Chapter 2 reviews relevant literature within the revenue management field. We begin with a short introduction on revenue management and its history, describe what booking control policies are, and then present the important published works on price and quantity-based models. We review the classical mathematical programming approximation models, the Deterministic Linear Programming (DLP), Expected Marginal Revenue (EMR) and Randomised Linear Programming (RLP), and the more recent area of research, customer choice modelling. In the final sections of the chapter, we explore risk-averse and robust capacity control literature and review the robust decision framework (section 2.5.1) that we use in our formulations.

In Chapter 3 we state the robust formulation for the network revenue management problem as presented by Perakis and Roels (2010), and introduce ellipsoidal uncertainty sets to the formulation. Finally, we provide a brief overview of genetic algorithms and then detail the design choices made and implementation procedure followed to tailor the algorithm to the problems under investigation in this thesis.

Chapter 4, describes the simulation model and the booking acceptance algorithms enforcing the booking limits derived by the network approximation models. We test resulting booking limits by emulating the booking horizon and present the results of our numerical experiments, which show an improved performance of the robust controls via the genetic algorithm compared to the heuristics employed in the original implementation.

Chapter 5 details the application of robust controls to the cruise network formulation also presented in the chapter. We also discuss the alterations made to the genetic algorithm to facilitate the added complexity of the cruise network compared to the airline network. Subsequently, we present a simulation study of the revenue generated via the different booking controls.

Finally, Chapter 6 is a critical discussion of the work presented in this thesis and the future work that can be carried out on this stream of work.

Chapter 2

Literature Review

In all cases, the objective of Revenue Management is selling the right product to the right customer at the right time [for the right price]
— Robert Cross

This literature review is structured as follows: In Section 2.1 a brief introduction to the history of revenue management is given, introducing the classical risk-neutral approach of maximising expected revenue and presenting the classification of models into price or quantity based as well as single or multiple resources. Section 2.2 introduces the various booking controls, i.e. the mechanisms used to control the availability of resources. Such mechanisms are used across the field, both in price and quantity based models and single or network applications. Section 2.3 includes formulations tackling single resource problems. Both price and quantity based models are presented, and we also give an overview of customer choice models. Section 2.4 follows the same format as the previous section but focuses on models that utilise multiple resources. Section 2.5 discusses two more recent research streams: risk-averse and robust revenue management.

2.1 Introduction

Revenue Management (RM) is a field in which academics and practitioners are actively engaged, resulting in several formulated definitions. Perhaps the most well-known purpose of the field is the one given by Cross (1997) "*In all cases, the objective of Revenue Management is selling the right product to the right customer at the right time [for the right price]*". This statement accurately describes the main objectives of the field but fails to encompass the methodology behind it. To this end, the definition by Talluri and Van Ryzin (2006) is more explicit, "*RM is the collection of techniques, strategies and tactics employed by firms to manage demand for their products and services scientifically*".

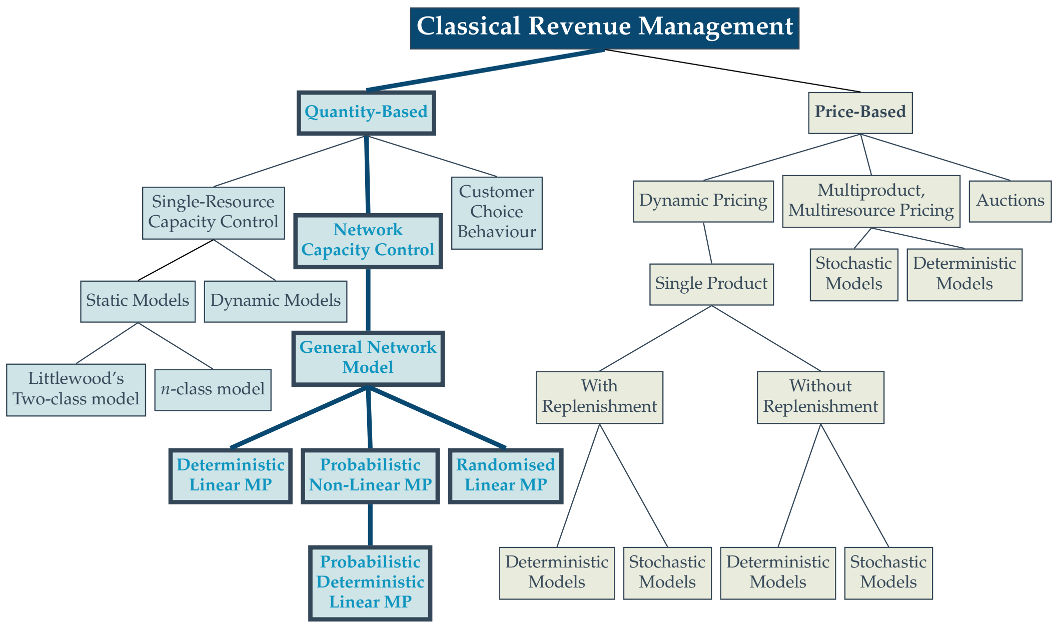


FIGURE 2.1: The Field of Revenue Management

In general, the revenue management field can be separated into two broad categories, *Quantity* and *Price-based* models. Quantity-based models describe situations where the decision-maker manages the demand by varying the quantity of the product on offer. In contrast, Price-based models describe the process where the decision-maker alters the products' price to manage the demand. Another major classification of models differentiates products that utilise single or multiple resources.

In the first 30 years of research in the field, the 'classical' revenue management techniques and methods were developed. The first mathematical models were introduced by Rothstein (1971) who proposed a simple overbooking model for airline companies and Littlewood (2005) with his seminal work on the single-leg, two-fare capacity allocation problem. In the last 20 years, research has shifted from myopic policies focused on the vendor's actions to modelling customers' behaviour, incorporating risk measures and robustifying optimisation models. McGill and Van Ryzin (1999) give an overview of the field at the time, Bitran and Caldentey (2003) focus their review on dynamic pricing models, while Talluri and Van Ryzin (2006) and Chiang et al. (2006) give comprehensive overviews of classical revenue management in their respective books. More recent efforts focus on specific areas of the field such as Gonsch (2017) who concentrates on the rise of risk-averse and robust revenue management, Strauss et al. (2018) describe recent developments in customer choice modelling while Klein et al. (2019) demonstrate the broad use of revenue management on industry applications.

In our review of the literature, we aim to encompass both the classical and risk-averse RM streams of work. Thus we divide the RM overview into four subsections: First, in section 2.2 we discuss the various types of booking controls that control the availability of resources. Section 2.3 introduces works on single-resource RM and similarly in section 2.4 the multiple-resource models are discussed. Section 2.5 overviews the risk-averse RM that introduces risk measures and robust optimisation. This leads to the final section of the chapter where gaps in literature are identified.

2.2 Types of Control

There are various ways to control the availability of resources subject to capacity and demand constraints. The objective is to find an effective and easy to implement policy or, as it is often referred to in this context, 'control', to allocate the available resources to booking requests for the different products, $j \in \mathcal{J}$. In order to choose the control, several criteria must be considered, such as technological constraints imposed by distribution and reservation systems, how profitable the method is, and the overall robustness of the control. The following subsections describe the most commonly used controls, namely booking limits, protection levels, and bid prices.

2.2.1 Booking Limits and Protection Levels

Booking limits are controls that limit the amount of capacity that can be sold to any particular booking class at a given point in time. *Protection levels* specify an amount of capacity to reserve (protect) for a particular booking class. Both Booking Limits and Protection Levels can be either partitioned or nested.

A *partitioned booking limit* is equivalent to a *partitioned protection level* and it essentially divides the available capacity in each leg of the network into a number of distinct buckets, one for each booking class. Demand for each product j can only access the allocated capacity to its booking class and none other. Let s be a set of products, or a bucket as it is often called, such that $s \in \mathcal{S}$. Then for the partitioned booking limits, the number of buckets is equal to the number of products and we have that,

$$\mathcal{S} = \bigcup_{j=1}^{n} \{j\} \tag{2.1}$$

This control often leads to an undesirable situation. Consider the case where a high revenue yielding booking class has utilised all of its allocated capacity and there is still demand for this booking class. If any lower booking class still has unsold capacity it would have been desirable to sell the unsold capacity to the higher revenue yielding customers. However under the partitioned booking limits control this is not possible. Furthermore, in the network setting dividing the available capacity to each product results into a very large number of small allocations. This is especially undesirable when demand has a high variability because it often leads to products being allocated a very small number or even fractions of seats. Hence, for these reasons partitioned booking limits are not usually used in practice, even though they are often used for theoretical results such as providing bounds or in approximate models. Furthermore, there are situations where the structure of the system dictates that partitioned booking limits are needed, for example if the seat type changes.

2.2.2 Nested Controls

A *nested booking limit* overcomes the limitations of its partitioned counterpart by allowing booking classes access to capacity in a hierarchical manner. Booking classes are ranked according to some measure (usually fare value) with the highest-ranked booking class having access to all capacity and the lowest-ranked booking class having access only to its allocated capacity.

Let f represent fare value and considered an ordered list of the products based on their fare value, such that $f_1 \leq f_2 \leq ... \leq f_n$. Then a nesting policy on a bucket s containing products $\{j, ..., n\}$ limits the number of accepted requests for any product in the bucket, $s = \{j, ..., n\}$. Hence we have,

$$\mathcal{S} = \bigcup_{j=1}^{n} \{j, j+1, ..., n\} \tag{2.2}$$

Nesting Heuristics were developed to address the problem presented when applying partitioned booking limits calculated by optimization models to networks. Generally speaking the higher the number of legs and fares in the network the more complex the problem becomes. The issue presented is twofold. Firstly, the number of seats allocated to each product by Mathematical Programming formulations becomes very small, often less than 1 for hub and spoke networks, and secondly, the demand for those seats is characterised by large uncertainty. To overcome this issue, a number of different nesting methods were proposed. Four of them are presented here. Nesting by fare class, fare value, shadow prices and bid prices. The main idea behind all of these methods is to never refuse a higher-valued passenger when seats originally allocated to a lower-valued passenger are still available. These methods are well explained in Williamson (1992), while Talluri and Van Ryzin (2006) give explicit background and mathematical formulations for these controls.

Consider a product j. Let a partitioned protection level for the product be denoted by p_j and a nested booking limit by b_j. If the capacity is c then the relationship between these two controls is given by,

$$b_j = c - p_{j-1}, \qquad j = 2, ..., n \tag{2.3}$$

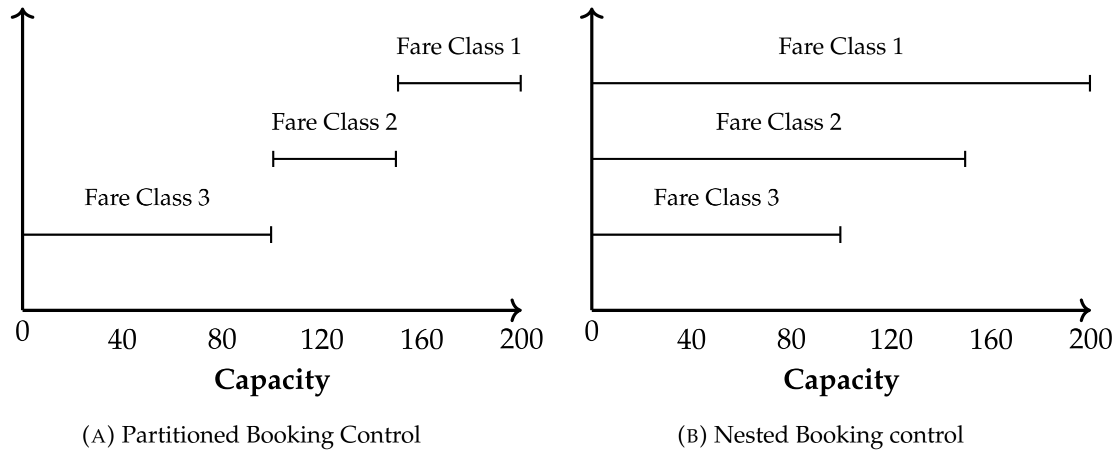


FIGURE 2.2: Control Policies

That is given that the booking classes are ranked hierarchically, the booking limit for class j is the capacity minus the protection levels of lower classes. Figure 2.3 resembles this relationship with a simple example. Consider three booking classes ranked by fare value, where Fare Class 1 $\geq$ Fare Class 2 $\geq$ Fare Class 3. The partitioned booking limits for each class are given in brackets. The protection level for Fare Class 1, p_1, is 12 seats that is only Fare Class 1 has access to these seats. Furthermore, the booking limit of Fare Class 1, b_1, is equal to 30, that is Fare Class 1 customers can access all the capacity. Similarly, the protection level for Fare Class 1 and 2, p_2, equals 22 seats. That is only customers of Fare Classes 1 and 2 can access these seats while b_2 equals 18, that is Fare Class 2 customers can access the seats allocated to Fare Class 2 and 3 customers.

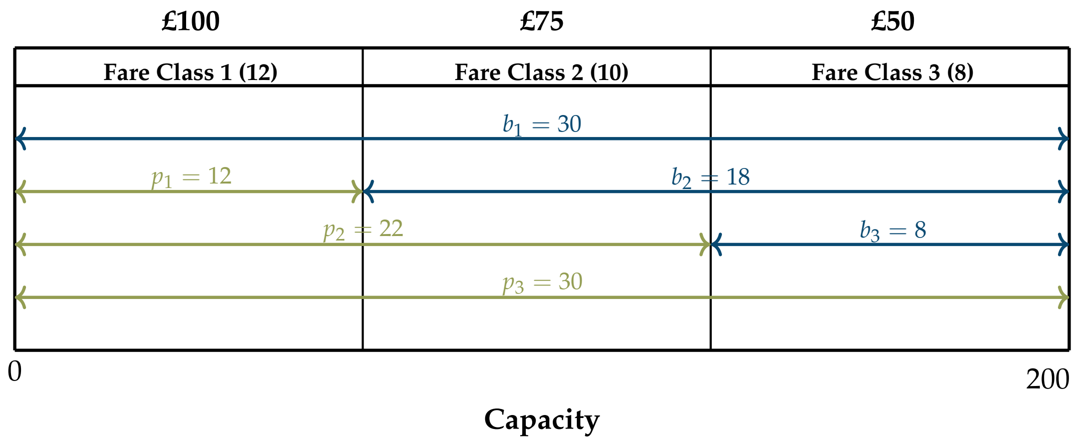


FIGURE 2.3: Relationship between booking limits b_j and protection levels y_j

Nesting by Fare Class

This is the simplest method proposed. The number of seats allocated to each fare class by the network optimisation model are summed together in one bucket. Hence these buckets are then used as protection levels. Booking limits can then be easily calculated by subtracting the aggregated protection levels of the higher classes from the leg capacity.

Class	Fare Value	No. Booked	Partitioned Allocation	Seats Available
1	250	50	50	0
2	125	44	50	6
3	75	77	100	23

(A) Discrete Fare Allocation

Class	Fare Value	No. Booked	Nested Allocation	Seats Available
1	250	50	200	29
2	125	44	150	29
3	75	77	100	23

(B) Nested Fare Booking Limits

TABLE 2.1: Small Partitioned Vs Nested Booking Limits Example

Nesting by Fare Nominal Value

This method is also simple. It ranks the different products by fare value. The product with the highest fare value is ranked at the top and the product with the lowest fare value is ranked at the bottom of the hierarchy. Different products can be can be grouped together into inventory buckets by specifying a range of fare prices. Then the individual seat allocations calculated by the network optimisation model are aggregated together and are used as the protection level for the bucket. Hence booking limits can then be calculated as in the case of nesting by fare class.

Nesting by Shadow Prices

This nesting method was first introduced by Williamson (1988) and aimed at capturing the value that each product added to the network. Shadow prices are defined as the additional revenue that would be generated if an additional seat was allocated to a given product, all else held constant. They are calculated as the dual prices of the network optimisation model solution.

2.2.3 Bid-Price Controls

Another simple method to perform capacity control is bid prices. Suggested initially by Simpson (1989) bid prices can be estimated as shadow prices of the capacity constraints and are concerned with the marginal value of a seat for a leg of travel. The bid price is a threshold price calculated for each product. It equals the sum of the shadow prices of the resources combined in that product. The customer's booking request is accepted if the fare he is willing to pay exceeds this threshold price. The central assumption is that the relationship between revenue and the remaining capacity is linear.

This control considers each product individually hence

$$\mathcal{S} = \bigcup_{j=1}^{n} \{j\}$$

but there is no booking limit, $b_j = \infty$, if the the fare value of the product is greater than the displacement cost of the resources it consumes, $f_j \geq \boldsymbol{y}'\boldsymbol{A}$, where $\boldsymbol{y}$ is the vector of dual prices, otherwise the booking class is closed $b_j = 0$.

While this method is easy to understand and implement, Talluri and Van Ryzin (1998) proved that bid prices are not generally optimal and do not guarantee correct acceptance or denial decisions. Particularly they note that selling one unit of capacity might result in a significant change in the capacity of multiple resources simultaneously. Thus, the interpretation of the bid prices as the marginal value of *one* unit of additional capacity may not be correct. Secondly, they challenge the assumption that revenue depends on the remaining capacity in a linear fashion. However, they prove that bid-price controls are asymptotically optimal when leg capacities and sales volumes are sufficiently large.

2.3 Single-Resource Revenue Management

2.3.1 Quantity Based

Littlewood's two-class, single-leg model received considerable attention in the literature due to its intuitive ease and wide applicability. Littlewood proposed the simple rule that an airline should accept a request for a discounted seat on the plane as long as its fare value exceeds the *expected* value of a full fare seat. Mathematically this is given as,

$$f_2 \geq f_1 Pr[d_1 > p_1], \tag{2.4}$$

where f_i is the fare value or aggregate revenue for the ith fare class d_1 is the demand for the higher fare class and p_1 is the protection level for the higher fare class.

Belobaba (1987a,b, 1989) extended Littlewood's model to incorporate multiple fare classes by proposing two sub-optimal but widely used heuristics. EMSR-a and EMSR-b, both use the notion of the Expected Marginal Seat Revenue and are simple to understand and implement. EMSR-a calculates protection levels for pairs of classes using Littlewood's rule assuming that only the classes in a pair exist. That is considering classes j and $j+1$, EMSR-a sets the protection level for class j to

$$f_{j+1} = f_j Pr[d_j > p_j]. \tag{2.5}$$

When all the pair-wise protection levels are calculated, the protection level for each class is calculated as the summation of the protection levels of all lower level classes. Mathematically, this is given as,

$$p_j = \sum_{k=1}^{j} p_k^{j+1} \tag{2.6}$$

where p_k^{j+1} is the protection level for class k assuming only classes k and $j+1$ exist. Of course this approach has the limitation that it ignores the effect on average demand by lumping together multiple classes, especially when these classes have similar fares. This leads to over-conservatism.

To overcome this problem EMSR-b again considers pairs of classes $j+1$ and j where j is an artificial class combining all fare classes $1, 2, ..., j$. The demand for this artificial class is equal to the aggregate demand of all fare classes included inside the artificial class and its fare value is equal to their weighted-average fare values. Hence, the fare value is given by,

$$\bar{f}_j = \frac{\sum_{k=1}^{j} p_k E[d_k]}{\sum_{k=1}^{j} E[d_k]} \tag{2.7}$$

and demand is given by

$$S_j = \sum_{k=1}^{j} d_k \tag{2.8}$$

where D_k is the mean demand for each fare class j. Then the EMSR-b heuristic sets protection level p_j equal to,

$$f_{j+1} = f_j Pr[S_j > p_j], \tag{2.9}$$

Further work on single-leg capacity control was done with the introduction of *nesting allocation policies*. Nesting heuristics allow higher class customers to have access to unsold seats initially allocated to lower class customers. They are discussed in greater detail in sections 2.2.2.

Important works on single-leg nested seat allocations were developed by Curry (1990), Robinson (1995), Wollmer (1992) and Brumelle and McGill (1993) who find the optimal nested allocation employing different methods. A very important assumption made by all these authors is the sequential arrival of booking requests where lower class customers always arrive first. This is a strong assumption that is still adopted today.

Ball and Queyranne (2009) propose an online algorithm for the single-leg problem using the competitive ratio notion. That is they compare the performance of the online algorithm against the optimal of the offline policy. Lan et al. (2008) consider the classical multi-fare, single-resource problem with the added complication that demand information is limited. They introduce a competitive analysis approach as well, which guarantees a certain performance level under all possible demand scenarios.

2.3.2 Price Based

Lee and Hersh (1993) develop a discrete-time dynamic programme (DP) model for finding optimal booking policies that does not depend on assumptions regarding customer arrival process. Gallego and Van Ryzin (1997) construct an upper bound for the optimal expected revenue by analysing a deterministic version of the DP problem. Furthermore they propose two heuristics for the stochastic formulation of the problem and show that they are asymptotically optimal as the expected sales volume tends to infinity.

Other academics tried approximation methods such as approximating the revenue-to-go function. Bertsimas and Popescu (2003) and Adelman (2007) propose such dynamic programming formulations.

2.3.3 Customer Choice Modelling

Another very important modelling aspect of RM problems is the behaviour of customers. Its intuitive importance to RM is obvious while it also provides a better description of a real life system where a vendor tries to maximise revenue and customers choose products according to their preferences. Such models are often characterised by increased mathematical complexity that makes the use of approximation methods necessary to solve them.

Talluri and Van Ryzin (2004) proposed a general discrete customer choice model for single-leg capacity problem introducing the notion of “efficiency” that can be solved exactly. The problem formulation could easily be adapted to a range of applications. Gallego et al. (2004) extended the Deteministic linear programming formulation to the customer choice setting indtroducing the *choice-based linear program*.

2.4 Network Revenue Management

Network revenue management refers to the quantity-based allocation of multiple resources. This class of problems relates to situations where customers buy a bundle of resources in combination under various terms and conditions. Typical applications include airline, cruise and train networks or sequential reservations for a hotel or car rental service. Network revenue management poses significant implementation and methodological challenges because it vastly increases the complexity and volume of data collected, stored and managed. It also requires demand forecasts for every network product at each point in the booking process. Lack of detailed data can cause severe numerical and estimation problems.

2.4.1 Price Based

Although our focus in this project is on quantity-based revenue management, multi-period models from the price-based branch of the revenue management tree are also of interest because of their close resemblance of the network capacity problem. For example the single-leg revenue management problem with two-fare classes is essentially equivalent to the much studied single-period or newsvendor problem, while the multi-product, multi-resource problem is equivalent to the network capacity problem.

Dynamic Pricing is of interest because when a discrete customer arrival process is assumed, a network revenue management problem can be formulated as a DP problem. We present the classic DP network revenue management problem as described by Talluri and Van Ryzin (2006).

2.4.1.1 The Dynamic Program

Talluri and Van Ryzin (2006) formulate the dynamic program for specifying the optimal booking control. Given the current time t, the current remaining capacity $\boldsymbol{c}$, and the current request vector for products, $\boldsymbol{P}(t)$ where products are denoted by $j = 1, ..., n$, the vendor must decide whether to accept or deny requests. Let the $n -$ vector $\boldsymbol{q}^{\pi}$ denote the vendor's decision. If the vendor accepts a request for product j in period t then $q_j^{\pi}(t) = 1$ and $q_j^{\pi}(t) = 0$ otherwise. The decision to accept, is a function of the remaining capacity vector $\boldsymbol{c}$ and the price f_j of product j, that is $q_j^{\pi}(t) = (t, \boldsymbol{c}, f_j)$, and hence $\boldsymbol{q}^{\pi}(t) = (t, \boldsymbol{c}, \boldsymbol{f})$. Since the vendor can accept at most one request at any period t and resources cannot be oversold, if the current capacity is $\boldsymbol{c}$, then $\boldsymbol{q}^{\pi}(t)$ is restricted to the set $Q^{\pi}(\boldsymbol{c}) = \{\boldsymbol{q}^{\pi} \in \{0,1\}^n : \boldsymbol{A}\boldsymbol{q}^{\pi} \leq \boldsymbol{c}\}$.

Let $\boldsymbol{q}^{\pi *}(t, \boldsymbol{c}, \boldsymbol{f})$ denote the optimal control and let $V_t(\boldsymbol{c})$ be the value function of the maximum revenue expected to go given the remaining capacity $\boldsymbol{c}$ and the time t. Then $V_t(\boldsymbol{c})$ must satisfy the Bellman equation,

$$V_t(\boldsymbol{c}) = \mathbb{E}\left[\max_{q^{\pi} \in Q^{\pi}(\boldsymbol{c})} \{\boldsymbol{P}(t)'\boldsymbol{q}^{\pi}(t, \boldsymbol{c}, \boldsymbol{f}) + V_{t+1}(\boldsymbol{c} - \boldsymbol{A}\boldsymbol{q}^{\pi})\}\right] \tag{2.10}$$

with the boundary condition,

$$V_{T+1}(\boldsymbol{c}) = 0, \ \forall\, \boldsymbol{c}. \tag{2.11}$$

Therefore an optimal control $\boldsymbol{q}^{\pi *}(\cdot)$ satisfies,

$$q_j^{\pi *}(t, \boldsymbol{c}, f_j) = \begin{cases} 1, & if\ f_j \geq V_{t+1}(\boldsymbol{c}) - V_{t+1}(x - \boldsymbol{A}_j) \text{ and } \boldsymbol{A}_j \leq \boldsymbol{c} \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

Equation 2.12 represents the notion that an optimal control will accept a booking request for product j at price f_j if and only if there exists efficient remaining capacity and its price exceeds the opportunity cost of the resource capacities the request utilises.

While the structure of an optimal control is given by the above dynamic program, computing the value function $V_t(\boldsymbol{c})$ exactly for any network of realistic size is not possible because of the curse of dimensionality. Consider a network with 10 resources and capacities of $C_k = 200$ for each resource k. Then the state space of the program will have 200^{10} states. Instead simpler approximations of networks are often used to acquire an estimate of its value function. The two main approaches are: a) decomposing the network into a collection of single-resource problems and b) reformulating the problem as a static mathematical program.

We are concerned with the latter case and present three classical mathematical programming models used to optimise networks to acquire booking limits and displacement costs, namely the Deterministic Linear Programming (DLP) model, the Expected Marginal Revenue (EMR) model, and the Randomised Linear Programming (RLP) model. These models are discussed in section 2.4.2.

2.4.2 Quantity Based

The primary aim of any network approximation model is to produce an accurate estimation of the value function $V_t^M(\boldsymbol{C})$ of the network. This is important not only because the solutions of such mathematical programs can be applied directly as partitioned capacity allocations to the different products but more importantly because good estimations of the displacement costs can also be produced.

Further to accuracy, the speed of computation of the model is equally important. In practice decisions are often made in small time spans and the static approximation models need to be recomputed frequently given new capacities and time inputs. Therefore a particularly accurate model with high computation time is not appealing to practitioners. Thus both the accuracy and speed of the approximation models are essential factors in evaluating their performance.

Mathematical Programming (MP) was one of the first approaches to be used to tackle the problem. Again due to typical network size, optimisation usually only occurs once. Such models are known as static. Glover et al. (1982) were the first to propose such a model. They introduced a network flow formulation where demand was deterministic instead of stochastic.

The Deterministic Linear Programming model (DLP)

The Deterministic Linear Programming method (DLP) was originally suggested by Williamson (1992) and can be considered as a simplification of the above. Williamson maximised revenue over the expected value of future demand, $E[d_j]$, for each product $j = 1, ..., n$ on sale. Her model produces partitioned booking limits but the dual prices can be used for bid price control. The model is easy to understand and fast to implement but it only yields an upper bound for the expected revenue (using Jensen inequality) that has been proven to be infeasible in practice (Cooper (2002)).

The DLP is a deterministic mathematical programming problem that consists of finding the 'optimal' product allocation, $\boldsymbol{x} = [x_j \mid j = 1, ..., n]$, such that the capacity and demand constraints are satisfied. Mathematically the problem is formulated as follows,

$$\begin{aligned} V_t^{DLP}(\boldsymbol{c}) \ = \text{maximize} \quad & \sum_{j=1}^{n} f_j x_j \\ \text{subject to} \quad & \sum_{j \in A_i} x_j \leq c_i && \forall \quad i = 1, ..., m \\ & x_j \leq \mathbb{E}[d_j] && \forall \quad j = 1, ..., n \\ & 0 \leq x_j && \forall \quad j = 1, ..., n \end{aligned} \tag{2.13}$$

de Boer et al. (2002) argue that the linear relaxation of the DMP (DLP) is tight; i.e. it yields an integer solution without the need to include an integrality constraint in the formulation. Due to the faster speed of computation, solving the DLP formulation is preferred in practice. The solution found by the DLP gives rise to partitioned booking limits for every j while the displacement costs can be calculated as the shadow prices of the capacity constraints. The DLP is easy to implement in a MATLAB environment and solve using a commercial solver like CPLEX or GUROBI.

Even though the DLP is easy to understand and implement it yields partitioned booking limits that are a sub-optimal policy. To demonstrate this, consider the case where demand for product j is lower than the allocated seats calculated by the DLP. Then by the definition of the partitioned booking policy, the remaining unsold seats cannot be bought by any other customer, and thus empty seats remain in the flight, therefore, losing revenue. To overcome this problem, booking control heuristics are used as described in Section 2.2. Further to the limitation of partitioned booking limits, DLP also disregards uncertainty in demand and assigns a deterministic value of demand to each product; thus, it is not robust.

The Probabilistic Mathematical Programming model (PMP)

In a static mathematical program the goal is to maximise the expected revenue of the network. Revenue is generated by customers willing to buy the products on offer. Let x_j be the number of seats to be allocated on a plane for product j. Then the revenue will be maximised only if all of the allocated seats x_j are sold. In cases where demand for any product j is less than the allocated seats, i.e $d_j < x_j$, then the seats sold are only equal to the demand d_j that has materialised. Hence the value of the general network capacity control seat allocation problem can be approximated by the following mathematical formulation,

$$\begin{aligned} V_t^{PMP}(\boldsymbol{c}) &= \text{maximise} \quad \mathbb{E}\left[\sum_{j=1}^{n} f_j \min\{x_j, d_j\}\right] \\ &\text{subject to} \quad \sum_{j \in A_i} x_j \le c_i \qquad \forall \quad i = 1, ..., m \\ &\qquad\qquad 0 \le x_j \le d_j \qquad \forall \quad j = 1, ..., n \end{aligned} \tag{2.14}$$

This formulation is a Non-linear program where the objective function is concave and separable. To overcome this the Deterministic Mathematical Program is introduced.

The Expected Marginal Revenue model (EMR)

DLP's disregard for the uncertainty in demand, is a very strong simplification assumption. Several attempts were made to incorporate the stochasticity of demand in the mathematical optimisation models. Wollmer (1986) proposed a model where demand was probabilistic and optimised the Expected Marginal Revenue (EMR) of the network. EMR is a notion introduced in the paper where each seat in the network is associated with a potential revenue if sold to a specific Origin-Destination-Fare (ODF) combination. His model is still widely used because of the probabilistic nature of the demand. Mathematically it is expressed as follows,

$$\begin{aligned} V_t^{EMR}(\boldsymbol{C}) &= \text{maximize} \quad \sum_{j=1}^{n} \sum_{k=1}^{M_j} f_j \, P(d_j \ge k) \, x_j(k) \\ &\text{subject to} \quad \sum_{j \in A_i} \sum_{i} x_j(i) \le c_i \qquad \forall \quad i = 1, ..., m \\ &\qquad\qquad x_j(k) \in \{0, 1\} \qquad \forall \quad j = 1, ..., n \end{aligned} \tag{2.15}$$

where $x_j(k)$ is a binary variable representing whether $k = 1, ..., M_j$ seats are allocated to the product and $M_j = \max_i \{\frac{c_i}{j} \in A_i\}$. Again the LP relaxation of the above formulation is tight. While the model is intuitively attractive, the large number of decision variables limits its practical applicability. The EMR is implemented in MATLAB environment and solved using the CPLEX 17.1 solver. The probabilities $P(d_j \ge k)$ are calculated assuming demand follows a normal distribution and uses the mean and standard deviation values as reported in de Boer et al. (2002).

The Randomised Linear Programming model (RLP)

The RLP was first introduced by Talluri and Van Ryzin (1999) as an extension to the DLP method. It aimed to compute better bid prices by incorporating the stochastic information about demand into the procedure of calculating bid prices. The method consists of simulating a variety of future realizations of the itinerary demand and solving deterministic linear programs to allocate capacities to each realization. The dual prices are then averaged to form a bid price approximation. In essence the method replaces the expected value of future demand $E[\boldsymbol{d}]$ as seen in equation 2.13 with the random variable d_j itself, i.e. the random variable characterising future demand. Mathematically this is given by,

$$\begin{aligned} V_t^{RLP}(\boldsymbol{c}) \;=\; &\text{maximize} && \sum_j f_j\, x_j \\ &\text{subject to} && \sum_{j \in A_i} x_j \le c_i && \forall \quad i = 1, ..., m \\ &&& x_j \le d_j && \forall \quad j = 1, ..., n \\ &&& 0 \le x_j && \forall \quad j = 1, ..., n \end{aligned} \tag{2.16}$$

Then we can simulate N independent samples of the demand vector $\boldsymbol{d}_j = [d_{j1}, ..., d_{jn}]$ and solve problem 2.13 for each realisation. We can then estimate the gradient by taking the average,

$$\frac{1}{N} \sum_{k=1}^{N} \mu(x_j, d_{jk}) \tag{2.17}$$

where the vector $\boldsymbol{\mu}(x_j, d_{jk})$ is either the partitioned booking limits calculated for each product or the bid price vector.

Talluri and Van Ryzin conclude their paper by suggesting that the method is a simple extension to the DLP method that provides an admittedly small but important improvement to the expected revenue. Furthermore, they suggest that revenue performance might be improved further by using variance reduction technique in the simulation steps of the method and comment on the future research that might be conducted on simulation-based optimisation. The RLP method is easily implemented in MATLAB environment and solved using the CPLEX 17.1 solver. As in the case for the EMR model the random variable D_j is assumed to follow a normal distribution and the mean and standard deviation values are as reported in de Boer et al. (2002).

2.4.3 Customer Choice Modelling

Liu and Van Ryzin (2008) extended the "efficiency" notion from single-leg to a network setting and applied it to the choiced-based linear program of Gallego et al. (2004). Using the same notation as in section 2.4.1.1, their deterministic approximation to the dynamic programme is given by,

$$\begin{aligned} \mathrm{V}_t^{\mathrm{CDLP}}(\boldsymbol{c}) = \text{maximise } & \sum_{S \subseteq N}^{T} \boldsymbol{f}(t, \boldsymbol{d}(t)) \\ \text{subject to } & \sum_{t=1}^{T} \boldsymbol{A}\boldsymbol{d}(t) \leq \boldsymbol{c} \\ & \boldsymbol{d}(t) \geq 0, \qquad t = 1, \ldots, T \end{aligned} \tag{2.18}$$

In a network setting, Van Ryzin and Vulcano (2008) develop a customer choice model and propose a stochastic approximation in conjunction with virtual nesting to solve it. Zhang and Adelman (2009) extend the work of Liu and Van Ryzin (2008) on the choice-based linear programming that solves the multinomial logit customer choice problem by introducing an extension to the approximate dynamic programming formulation of Adelman (2007) to the customer choice setting.

Rusmevichientong and Topaloglu (2012) consider Robust formulations of assortment optimization problems under the multinomial logit choice model for both the static and dynamic settings. They assume that the true parameters of the logit model are completely unknown and are represented by a compact uncertainty set. The innovation to their approach is the structure of the proposed uncertainty set that is characterized by a radius parameter ϵ that enables the decision maker to control the trade-off between increasing the average revenue and protecting against the worst-case scenario. Kunnumkal and Talluri (2016) present a comparison of the approximation techniques proposed by other authors to calculate upper bounds to the value function of the network customer choice management dynamic program.

2.5 Robust and Risk-Averse Revenue Management

In this section articles where the risk-neutrality of the decision maker assumption is challenged but demand is still modeled via known probabilities are presented. Such treatments fall under the umbrella of risk-averse revenue management. We further review models where no or only limited demand distribution information is available. In such cases, formulations aim to optimise revenue for some worst-case scenario. A second objective is to minimise the variability in revenue. Such models consist the robust revenue management stream.

As mentioned in Section 2.1, revenue management can be divided into Quantity-based and Price-based models. This classification is also valid in robust settings. An interesting outcome of this distinction was noted by Gabrel et al. (2014) who stated that "Quantity-based problems often lead to LP formulations while Price-based models lead to Nonlinear problems". Gonsch (2017) provide an extensive review of this research stream.

2.5.1 Robust Decision Framework

In this section we introduce the general *maximin revenue* and *minimax regret principles* and show how they can be utilised in network revenue management problems. The maximin and minimax principles were introduced by the works of Wald (1945, 1950) and Savage (1951) respectively and are used specifically for the treatment of non-probabilistic uncertainty. They are also well established models in the fields of decision theory and are special versions of the maximin paradigm of game theory (Von Neumann and Morgenstern, 1944) where nature is assigned the role of one of the players. In a Wald maximin model nature must represent *uncertainty*.

The general maximin setting is characterised by a *decision space* V, a set that contains all the possible alternative decisions available to the decision maker and a *state space* W, a set consisting of all possible states under consideration. A real-valued function r defined on $V \times W$ specifies the payoffs associated with the decisions and states under consideration and is given by $r(v, w)$.

2.5.1.1 *Maximin Revenue Principle*

The generic Wald's model seeks the best alternative whose performance under the worst-case scenario is at least as good as the worst-case performance of all other alternatives. Mathematically, using the notation given above, it is expressed as,

$$v^* := \max_{v \in V} \min_{w \in W} r(v, w) \tag{2.19}$$

In revenue management terms, the decision maker chooses her booking policy limits, $\boldsymbol{b}$, and when demand realisation $\boldsymbol{d}$ occurs revenue is generated, given by the payoff function, $\boldsymbol{f}(\boldsymbol{b}, \boldsymbol{d})$. The aim of the decision maker is to maximise the minimum revenue associated with booking limits $\boldsymbol{b}$ over all demand realisations $\boldsymbol{d} \in \mathcal{D}$. The minimum revenue is given by,

$$\phi = \min_{\boldsymbol{d} \in \mathcal{D}(\boldsymbol{b})} \boldsymbol{f}(\boldsymbol{b}, \boldsymbol{d}) \tag{2.20}$$

Hence the maximin principle selects the booking policy limits that maximise the minimum revenue generated by the worst demand realisation under consideration,

$$\phi^* = \max_{\boldsymbol{b} \in \boldsymbol{\mathcal{B}}} \min_{\boldsymbol{d} \in \mathcal{D}(\boldsymbol{b})} \boldsymbol{f}(\boldsymbol{b}, \boldsymbol{d}) \tag{2.21}$$

The rationale of the model is to guarantee a minimum revenue over all future demand realisations. This approach only considers the worst-case scenarios and thus is extremely risk averse and may not perform well on average. Hence it often leads to conservative decisions where the trade off of robustness against revenue is heavily skewed towards robustness.

2.5.1.2 *Minimax Regret Principle*

To overcome the problem of overly conservative decisions the minimax regret principle was introduced by Savage (1951) with the aim of improving the control decisions on average given the uncertainty in demand. This is achieved by introducing the notion of regret, or loss as Savage called it. Let $R(v, w)$ to denote the regret associated with decision v ans state w. The regret is the difference between the payoff that the optimal decision v^* could have generated at state w and the payoff that the taken decision v has generated at state w. Hence,

$$R(v, w) = \max_{v^*} r(v^*, w) - r(v, w) \tag{2.22}$$

The minimax regret principle chooses the decision x that minimises the maximum regret function over all possible states, that is,

$$\begin{aligned} v^* :&= \min_{v \in V} \max_{w \in W} R(v, w) \\ &= \min_{v \in V} \max_{w \in W} \{\max_{v^*} r(v^*, w) - r(v, w)\} \end{aligned} \tag{2.23}$$

In revenue management terms, the decision maker again chooses his booking policy limits, $\boldsymbol{b}$. The payoff function gives the revenue that is associated with booking limits $\boldsymbol{b}$ and demand realisation $\boldsymbol{d}$ denoted by $r(\boldsymbol{b}, \boldsymbol{d})$. The regret, $R(\boldsymbol{b}, \boldsymbol{d})$, is defined as the difference between the payoff of the best booking limits we could have chosen for demand realisation $\boldsymbol{d}$ and the actual booking limits we have chosen for the demand realisation $\boldsymbol{d}$.

$$R(\boldsymbol{b}, \boldsymbol{d}) = \max_{\boldsymbol{b}^*} r(\boldsymbol{b}^*, \boldsymbol{d}) - r(\boldsymbol{b}, \boldsymbol{d}) \tag{2.24}$$

Hence the maximum regret is the additional revenue that could have been obtained with perfect information over all demand processes under consideration and mathematically is given as,

$$\rho = \max_{\boldsymbol{d} \in \mathcal{D}} R(\boldsymbol{b}, d) \tag{2.25}$$

The aim of the decision maker is to minimise the regret associated with demand realisation d,

$$\begin{aligned}\rho^* &= \min_{b\in\mathcal{B}} \max_{d\in\mathcal{D}} R(\boldsymbol{b},\boldsymbol{d}) \\ &= \min_{b\in\mathcal{B}} \max_{d\in\mathcal{D}}\{\max_{b^*} r(\boldsymbol{b}^*,\boldsymbol{d}) - r(\boldsymbol{b},\boldsymbol{d})\}\end{aligned} \tag{2.26}$$

2.5.2 Single-Resource

Birbil et al. (2009) propose robust formulations for both the static and dynamic setting of the single-leg seat allocation problem. By conducting simulation experiments they come to the conclusion that their robust formulations considerably reduce variability compared to the classical formulations, while the decrease in average revenue is negligible.

2.5.3 Network

Perakis and Roels (2010) propose robust formulations for the capacity allocation problem in RM. More specifically they use the minimax and maximin regret criteria, while assuming general polyhedral uncertainty sets. They also consider a number of booking control policies like partitioned booking limits, nested booking limits, displacement-adjusted virtual nesting and bid prices. Using numerical analysis they show that minimax regret control can outperform the classical heuristics for settings where future demand is censored or correlated. We draw heavily from this work in Chapter 3.

Thiele (2004) in her PhD thesis also applies data-driven robust formulations for both the single-leg and network settings. She considers an approach, which builds directly on the historical realizations of uncertainty, without requiring any estimation. In her model, a fraction of the best cases are removed to ensure robustness, and the system is optimized over the sample average of the remaining data. This leads to tractable mathematical programming problems.

As an extension to Thiele's PhD work (Thiele, 2004), Bertsimas and Thiele (2006) propose an approach that takes into account the uncertainty of the demand in the supply chain setting without assuming a specific distribution. Their approach is highly tractable while it also allows a trade-off between solution robustness and protection against uncertainty. Another important feature of the proposed approach is that the robust problem is of the same difficulty as the nominal problem computationally. Perakis and Sood (2006) propose a robust formulation for a dynamic pricing under uncertainty setting. Adida and Perakis (2010) present a computational study that compares robust against stochastic optimisation approaches for dynamic pricing and inventory control.

2.6 Gradient Descent Algorithms

In this section we give an overview of the gradient descent algorithm. Any textbook on nonlinear optimization mentions that the gradient method is due to Cauchy et al. (1847). We are interested in these algorithms since they have been employed to solve equations 2.21 and 2.26 in the current literature. A more recent overview of gradient descent algorithms is given by Ruder (2016).

Gradient descent is an iterative first-order optimisation algorithm used to find a local minimum or maximum of a given function. Gradient descent is one of the most popular algorithms to perform optimization, especially deployed in machine learning and deep learning fields. It is often used to to minimise the cost function of the chosen machine learning model. They are however used as black-box optimizers, as practical explanations of their strengths and weaknesses are hard to come by.

In general for a gradient descent algorithm to work, the objective function, F, needs to be *convex* and *differentiable*. The algorithm iteratively calculates the next point $\boldsymbol{x}_{n+1}$. In each iteration the step direction and size must be determined. The direction is determined by the gradient at the current position, $\nabla F(\boldsymbol{x}_n)$, scaled by the step size (or *learning rate*) γ_n. If the aim is to minimize (or maximise) the objective function, $F(\boldsymbol{x}_n)$, a step is made by subtracting (or adding) the obtained value from the current position $\boldsymbol{x}_n$ respectively. This is mathematically expressed as:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma_n \nabla F(\boldsymbol{x}_n),\ n \geq 0. \tag{2.27}$$

The sequence is terminated when conditions are met such as reaching the maximum number of iterations N or the step size being smaller than the tolerance T. This results to a monotonic sequence $F(\boldsymbol{x}_0) \geq F(\boldsymbol{x}_1) \geq F(\boldsymbol{x}_2) \geq \cdots$, that in turn, yields the sequence $(\boldsymbol{x}_n)$ that converges to a local minimum.

The learning rate γ_n is an important parameter because it scales the gradient and thus controls the step size. The step size in turn has a strong influence on the performance of the algorithm since the lower the learning rate, the longer it takes to converge meaning that it may reach the maximum number of iterations before finding the optimum. On the other hand, if the learning rate is too big the algorithm may not converge to the optimum (jump around) or even diverge completely. A further limitation of the first-order gradient descent algorithm is the challenge it faces when there exists a saddle point in the objective function. When this is the case, obtaining a global minimum is not guaranteed.

There are many variants of gradient descent that aim to improve upon the limitations the algorithm faces. Nesterov (2003) proposed modifications to the method that enables faster convergence for convex problems while Second-order algorithms such as the Newton-Raphson method, deal with situations concerning saddle points better.

Algorithm 1 provides a simple outlook of the gradient descend method

Algorithm 1: Pseudo-code for a simple gradient descend algorithm

Inputs : Initialisation Point x_n, Gradient Function $F(x_n)$, Step Size γ_n, Maximum iterations N, Tolerance T

Outputs:

```
Choose $x_n$
while $n \leq N$ or $\gamma_n < T$ do
    Calculate $\nabla F(x_n)$
    Choose $\gamma_n$
    Calculate $x_n + 1 = x_n - \nabla F(x_n)$
end
```

2.7 Conclusion

We have looked at classical revenue management, both on single and multiple-resource settings and quantity and price based models. A considerable amount of research has been conducted on such risk-neutral models for the past 30 years.

More recent works focus on challenging the assumption of risk-neutrality of the decision-maker leading to robust and risk-averse model formulations. Section 2.5 introduces some of the works in the area but such approaches are almost exclusively focused on the single-leg example. As Gonsch (2017) suggests, the volume of research in this area is still scarce, while the potential impact of such models on RM practitioners is extremely high.

The second stream of recent RM developments is customer choice modelling. As the name suggests, the aim is to model customers' behaviour and their choice of products over a set of products available to each type of customer. While the field is not particularly new, it has gained considerable attention over the past ten years as businesses are drifting towards more customer-centric policies and aiming to deliver 'personalised' experiences to customers. This has led to an increased interest in the area leading to new formulations for assortment optimisation problems as Strauss et al. (2018) suggest. Customer choice models developed so far for the network setting are based on the DLP model to create the set of offered products to customers while there is no published work on robust customer choice network revenue management.

Hence we identify two areas of *future research*: a) *robust/risk-averse network revenue management* and b) *robust customer-choice modelling*. In this thesis we restrict our attention to robust/risk-averse network revenue management methodologies.

Chapter 3

Robust Capacity Control

In this chapter, we state the formulation of the network revenue management problem and reconsider the robust formulation introduced by Perakis and Roels (2010). The novelty of our treatment is twofold; first, we use ellipsoidal uncertainty sets to characterise demand, in addition to the polyhedral uncertainty sets proposed initially. Ellipsoidal sets lead to increased revenue but at the expense of computation times. Second, we design and implement a genetic algorithm to optimise the robust controls booking limits. The genetic algorithm results surpass the revenue gained by the local gradient descent initially suggested Perakis and Roels (2010), but it also significantly increases computation time.

The chapter is organised into three sections. First, a brief introduction to the network revenue management problem is provided. Then the mathematical formulation by Perakis and Roels (2010) and the extension of using ellipsoidal uncertainty sets is stated. In the third section, we describe the genetic algorithm we utilise to solve this problem. We split this part into two subsections. First, we explain our rationale for using a genetic algorithm and provide a detailed description of the design and implementation of the genetic algorithm operators we employ. In the second, we describe the procedure we carried out to hyper-optimise the algorithm's parameters and provide diagnostic plots showing how the algorithm converges. A complete set of results from the methodology presented here is given in Chapter 4.

3.1 Introduction

As mentioned in Chapter 2 *network* revenue management problems arise when customers request to buy a bundle of resources in combination under various terms and conditions. The classical paradigm of network revenue management is the airline industry. It was the first industry to adopt revenue management techniques and it is easy to conceptualize the examples. Nevertheless, the models described in this chapter can be applied to other industries such as the railway and hospitality.

Airlines typically offer hundreds of products as Origin, Destination and Fare combinations (ODF). Products might span multiple legs (resources) of the airline's network. Booking requests for such ODFs arrive simultaneously over the booking horizon. The objective is to find a booking control policy that will optimally allocate the capacity of the resources to the various product requests. This allocation has to be done dynamically as demand materializes. Modelling demand is another important aspect of network revenue management problems since future demand is often assumed to be stochastic and independent among the different products, a strong assumption that has been challenged lately in literature.

3.1.1 Problem Statement

Consider a network with m resources indexed by i (e.g. flights, night stays) and n products indexed by j, where each product is a bundle of resources sold under various terms and conditions. Heterogeneous customers stochastically arrive over a finite time interval $(0, T]$ and request to buy products. There are $\boldsymbol{c} = [c_i \mid i = 1, ..., m]$ available units of resources. Let $a_{ij} = 1$ if resource i is used by product j and $a_{ij} = 0$ otherwise. Then, the incidence matrix $\boldsymbol{A} = [a_{ij}]_{m \times n}$ denotes the resources that a product requires. Let $\boldsymbol{f} = [f_j \mid j = 1, ..., n]$ be the column vector denoting the fare revenue generated by selling product j and $\boldsymbol{d} = [d_j \mid j = 1, ..., n]$ denote the random total aggregate demand associated with each product j. A booking policy is a rule for accepting or rejecting booking requests and is denoted by $\pi \in \Pi$ where Π is the set of all non-anticipating policies, that is policies that take into consideration the information available up to time t where $t \in (0, T]$.

The objective is to find a policy π that maximizes the expected revenue, $\boldsymbol{f}'\mathbb{E}[\boldsymbol{q}^\pi]$, where $\boldsymbol{q}^\pi$ is the vector of total number of accepted requests per product on sale when policy π is in use. The policy needs to satisfy the capacity constraints, $\boldsymbol{A}\boldsymbol{q}^\pi \leq \boldsymbol{c}$, while the accepted requests must be non-negative and no greater than the total demand, i.e., $\boldsymbol{0} \leq \boldsymbol{q}^\pi \leq \boldsymbol{d}$. The problem can then be formulated as follows:

$$
\begin{aligned}
\underset{\pi \in \Pi}{\text{supremum}} \quad & \boldsymbol{f}'\mathbb{E}[\boldsymbol{q}^\pi] \\
\text{subject to} \quad & \boldsymbol{A}\boldsymbol{q}^\pi \leq \boldsymbol{c} \\
& \boldsymbol{0} \leq \boldsymbol{q}^\pi \leq \boldsymbol{d}
\end{aligned}
\tag{3.1}
$$

The control policies described in section 2.4.2 try to maximise revenue, $\boldsymbol{f}'\mathbb{E}[\boldsymbol{q}^\pi]$, by assuming that the decision maker possess enough information to model and confidently predict future demand and therefore a risk neutral approach towards risk is justified.

The resulting solutions can be used directly as booking limits or their dual values can be used to compute bid prices.

However the assumption that the decision maker is risk-neutral is not always true. Consider a newly established business that launches a new product, or events that happen only a few times in a year, where there is not enough information or data to accurately forecast future demand. Then the decision maker is more likely to adopt a risk-averse approach and be more interested in ensuring that at least a minimum revenue is achieved. It is in these cases, a robust formulation has the potential to generate useful results.

3.2 Mathematical Formulation

In this section we discuss further the two robust models as introduced by Perakis and Roels (2010) that use the *maximin revenue*, ϕ^* and *minimax regret*, ρ^*, measures described in section 2.5.1.

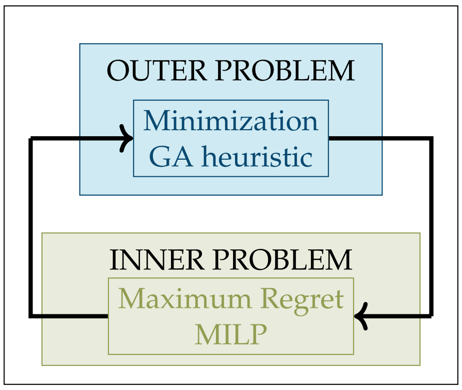
The framework specifies that the maximum regret ρ or the minimum revenue ϕ generated by a decision $d \in D$ that results in a state-space $s \in S$ are calculated for a given solution to the problem. The mixed-integer programme described in section 3.2.1 yields these values. The central assumption of this formulation is that only partial information about demand is known. Our first contribution in this chapter is utilising an ellipsoidal uncertainty set to describe demand uncertainty.

The second step in the robust decision framework is to compute the optimal value for the measure at hand, i.e. to maximise minimum revenue, ϕ^*, or to minimise maximum regret ρ^*. This is achieved via a heuristic method such as a gradient descent algorithm by Perakis and Roels (2010). Our second contribution in this chapter is our proposed use of a genetic algorithm for the optimisation of the outer problem.

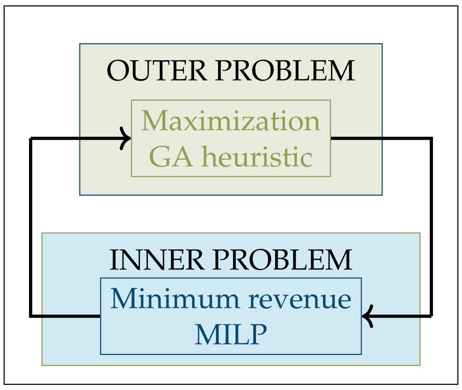
Hence, the overall hybrid optimisation procedure we employ to calculate the final booking limits utilising these robust controls is as follows:

1. Compute initial booking limits using any suitable method (section 2.4.2)
2. Compute the robust measure value, ϕ or ρ, using the exact MIP method (Inner Problem)
3. Compute the optimal robust measure value, ϕ^* or ρ^*, using an approximation method (Outer Problem)

In the first step a policy maker can choose any method to compute the initial booking limits. Step 2 refers to the inner optimisation, where we quantify the robust measures



(A) Minimizing the Maximum Regret



(B) Maximising the minimum revenue

FIGURE 3.1: An Exact Method within a Heuristic

by employing the MILP formulation of Perakis and Roels (2010) which is presented in the following section. Finally, step 3 refers to the outer optimisation where the booking limits are altered repeatedly by a heuristic method to find the booking limits that yield the best robust measure value.

Our second contribution to the formulation is the design and implementation of a genetic algorithm to perform the outer optimisation part. We discuss in detail the design elements and hyper-optimisation of the heuristic's parameters in section 3.3. In chapter 4 we showcase our simulation results that suggest that our heuristic outperforms the local gradient descent algorithm employed by Perakis and Roels (2010).

3.2.1 Robust MIP formulation

Let $\boldsymbol{x}$ be the decision vector for sales while $\boldsymbol{z}$ be the perfect information hindsight sales. Let $\boldsymbol{d} \in \mathcal{U}$ be the vector of realised demand for each product j that belongs in some uncertainty set $\mathcal{U}$. Π is the feasible decision set of booking control policies, assumed to be compact and π is a booking control policy. We denote a booking limit for a bucket s as b_s such that $\boldsymbol{b} = [b_s \mid s \in \mathcal{S}]$ and $R(\boldsymbol{b}, \boldsymbol{d})$ is the revenue associated with the booking limits $\boldsymbol{b}$ derived under booking policy $\pi \in \Pi$ when the demand process $\boldsymbol{d}$ is realised.

The objective is to find the maximum difference in revenue between the perfect hindsight information $\boldsymbol{f}'\boldsymbol{z}$ and the realised sales $\boldsymbol{f}'\boldsymbol{x}$ as shown in 3.2a. To find this value, the sales must be feasible. There are three conditions that must be satisfied to achieve this. First, the decision vector for sales, $\boldsymbol{x}$, or the hindsight perfect information sales $\boldsymbol{z}$ must not exceed capacity. This is shown by constraints 3.2b and 3.2c for $\boldsymbol{x}$ and $\boldsymbol{z}$ respectively. Secondly, demand must be characterised. The demand to be realised must belong to an uncertainty set $\mathcal{U}$ as shown by constraint 3.2f. Furthermore, the realised sales for a product j cannot exceed the demand for that product while also be non-negative. This must be true for both $\boldsymbol{x}$ and $\boldsymbol{z}$ as shown by constraints 3.2d and 3.2e. Thirdly, the booking limits, b_s, must not be exceeded. Constraint 3.2g enforces that realised sales satisfy the booking limits passed. From these three conditions only one can be in effect at any given time. The sales for product j must equal the demand for the product d_j unless the booking limits are exceeded, or the capacity is exceeded. To enforce this the binary variables $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and γ are introduced in constraints 3.2h - 3.2j. Constraint 3.2k ensures that only one of the three scenarios can occur at once.

To calculate the robust controls ϕ or ρ, the following mixed integer linear programming formulation must be solved. For any uncertainty set $\mathcal{U}$ and any booking control policy π, the maximum regret $\rho(\pi)$ is equal to the optimal value of the following MILP:

$$
\begin{aligned}
\underset{\boldsymbol{z},\boldsymbol{x},\boldsymbol{d},\boldsymbol{\alpha},\boldsymbol{\beta},\gamma}{\text{maximise}} \quad & \boldsymbol{f}'\boldsymbol{z} - \boldsymbol{f}'\boldsymbol{x} & & \text{(3.2a)}\\
\text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{c} & \forall\, M \quad & \text{(3.2b)}\\
& A\boldsymbol{z} \leq \boldsymbol{c} & \forall\, M \quad & \text{(3.2c)}\\
& 0 \leq \boldsymbol{x} \leq \boldsymbol{d} & & \text{(3.2d)}\\
& 0 \leq \boldsymbol{z} \leq \boldsymbol{d} & & \text{(3.2e)}\\
& \boldsymbol{d} \in \mathcal{U} & & \text{(3.2f)}\\
& \sum_{j\,\in\, s} x_j \leq b_s & s\, \in S \quad & \text{(3.2g)}\\
& \boldsymbol{d} \leq \boldsymbol{x} + M(\boldsymbol{1} - \boldsymbol{\alpha}) & & \text{(3.2h)}\\
& \sum_{j\,\in\, s} x_j \geq \beta_s b_s & s\, \in S \quad & \text{(3.2i)}\\
& \boldsymbol{A}_k x_j \geq c_k \gamma_k & k = 1, ..., m \quad & \text{(3.2j)}\\
\sum_{k=1:\, a_{kj}>0}^{m} \gamma_k + \alpha_j + & \sum_{s: j\,\in\, s} \beta_s \geq 1 & j = 1, ..., n \quad & \text{(3.2k)}\\
& \boldsymbol{\alpha} \in \ \{0,1\}^n & & \text{(3.2l)}\\
& \boldsymbol{\beta} \in \ \{0,1\}^{|S|} & & \text{(3.2m)}\\
& \gamma \in \ \{0,1\}^m & & \text{(3.2n)}
\end{aligned}
$$

where $M \geq \{\max_j d_j : \boldsymbol{d} \in \mathcal{U}\}$. Similarly, for any uncertainty set $\mathcal{U}$ and any booking policy π, the minimum revenue $\phi(\pi)$ is equal to the negative of the optimal value of Equation (3.2a) when $\boldsymbol{z} = \boldsymbol{0}$.

3.2.2 Constructing Uncertainty sets

The first documented attempt to protect the solution of a linear optimisation problem was developed by Soyster (1973), who proposed an optimisation formulation where the constraints elements are strictly members of a convex set. The resulting solution was deemed too conservative as too much optimality from the nominal problem was sacrificed to achieve robustness.

Almost a quarter of a century later the independent efforts of Ben-Tal and Nemirovski (Ben-Tal and Nemirovski, 1998, 1999, 2000), El-Ghaoui and Lebret (El Ghaoui and Lebret, 1997) and El-Ghaoui et al. (El Ghaoui et al., 1998) expanded the work of Soyster to include more general conic problems with robust formulations that were computationally tractable yielding results that were not as conservative. This was achieved mainly by the introduction of the ellipsoidal uncertainty sets.

Uncertainty sets are an integral part of the robust optimisation theory. A number of authors have suggested ways to form different uncertainty sets. Standard uncertainty sets are the so called *box, ellipsoidal, polyhedral, cone* and *convex*. The choice of structure for the uncertainty set is dependent on the problem being solved, but a number of possible structures exist that have been proven computationally tractable. Some such sets are:

1. Discrete sets $\mathcal{U} = \{\hat{\boldsymbol{x}}_i \mid i = 0, ..., n\}$.
2. Interval sets $\mathcal{U} = \{\boldsymbol{x} \mid \underline{\boldsymbol{x}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{x}}\}$.
3. Ellipsoid sets $\mathcal{U} = \{x \mid \|Ax\|_2 \leq \delta\}$.

Bertsimas and Brown (2009), Bertsimas and Sim (2004, 2003) and Ben-Tal et al. (2002) construct more complex uncertainty sets such as intersections of ellipsoidal sets, polyhedral uncertainty sets and sets with budgets of uncertainty. They explore their properties and formulate problems that yield computationally tractable robust counterparts. A brief mathematical overview of the most important uncertainty sets is given in Appendix B.3.

In our treatment, we are interested in two different kinds of uncertainty sets, polyhedral, denoted by $\mathcal{U}_{\mathcal{P}}$ and ellipsoidal, denoted by $\mathcal{U}_{\mathcal{E}}$. These are the most commonly used sets because they form robust counterparts that can be treated in reasonable computational times by standard commercial software as well as the fact that ellipsoidal uncertainty sets are linked with risk-averse measures as shown by Natarajan et al. (2009).

The general polyhedral uncertainty is described using the 1-norm of the uncertain data vector,

$$\mathcal{U}_{\mathcal{P}} = \{\xi \mid \|\xi\|_1 \leq \Gamma\} = \left\{\xi \mid \sum_{j \in J_i} |\xi_j| \leq \Gamma\right\} \tag{3.3}$$

where Γ is the adjustable parameter controlling the size of the uncertainty set. In our treatment we use the polyhedral uncertainty set $\mathcal{U}_{\mathcal{P}}$, defined by

$$\boldsymbol{d} \in \mathcal{U}_{\boldsymbol{P}} = \{d_j \mid l_j \leq d_j \leq u_j,\ j = 1, ..., n\} \tag{3.4}$$

which is the interval of lower and upper bounds for the demand of any product. When the lower and upper bounds are equal then we assume that our certain parameters have no perturbations and hence the uncertain optimisation problem reduces to its Deterministic Linear Programming relaxation (see equation 2.13).

In the Ellipsoidal $\mathcal{U}_{\mathcal{E}}$ case the demand for all products is characterised by the following equation,

$$\boldsymbol{d} \in \mathcal{U}_{\mathcal{E}} := (\boldsymbol{d}-\boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{d}-\boldsymbol{\mu}) \leq \lambda \tag{3.5}$$

where Σ^{-1} is the covariance matrix and μ is equal to the mean value of demand for each product $\bar{d}_j$. The scalar λ determines the size of the uncertainty set. More specifically, this set considers all possible realisation of demand within a radius of λ from the mean demand vector, where the ellipsoid is tilted and stretched by the covariance. When $\lambda = 0$, this set is just the singleton $\boldsymbol{\mu}$.

Using an ellipsoidal uncertainty set is different to using a polyhedral set because of the fact that we are adding a non-linear term to the formulation instead of a linear constraint. Specifically we introduce a second-order cone constraint which prevents $\boldsymbol{x}$ from being large in directions with considerable uncertainty in demand d. The impact of this constraint is the added complexity to the formulation that leads the increased computational cost. It also adds flexibility to the shape of uncertainty by changing it from a cylinder to a cone and also allowing altering its size via the scalar λ. These alterations aim to provide a better return on the risk vs revenue tradeoff. Results in Chapter 4 show the impact of these changes.

By varying the size of the $\mathcal{U}$ we can get a frontier on the behaviour of the two measures. To construct the uncertainty sets we assume that demand for each product follows a Normal distribution, $\boldsymbol{d} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma})$. For the polyhedral uncertainty set, the lower and upper bounds of demand for each product can be calculated as quantiles from sample generations of demand. The percentile of the lower and upper bound, q_l and q_u, are given on the third and fourth columns of Table 3.1 respectively.

For the ellipsoidal uncertainty set, the assumption that demand follows a normal distribution means that the value of λ for each level of uncertainty can be determined from the χ^2_p probability tables. The second column of Table 3.1 presents these values.

3.2.3 Constructing the Constraint Matrix

In this subsection we provide details on constructing the constraint matrix for problem 3.2a. More specifically we focus on constraints,

%	λ	q_l	q_u
100.00	64.00	0.000	100.000
99.95	44.43	0.025	99.975
99.90	42.31	0.050	99.950
99.50	37.16	0.250	99.750
99.00	34.81	0.500	99.500
97.50	31.53	1.250	98.750
95.00	28.87	2.500	97.500
92.50	27.22	3.750	96.250
90.00	25.99	5.000	95.000
87.50	25.00	6.250	93.750
85.00	24.20	7.500	92.500
82.50	23.42	8.750	91.250
80.00	22.76	10.000	90.000
77.50	22.16	11.250	88.750
75.00	21.60	12.500	87.500
70.00	20.60	15.000	85.000
65.00	19.70	17.500	82.500
60.00	18.87	20.000	80.000
55.00	18.10	22.500	77.500
50.00	17.34	25.000	75.000
45.00	16.60	27.500	72.500
40.00	15.89	30.000	70.000
35.00	15.20	32.500	67.500
30.00	14.44	35.000	65.000
25.00	13.70	37.500	62.500
20.00	12.86	40.000	60.000
15.00	12.00	42.500	57.500
10.00	10.86	45.000	55.000
7.50	10.21	46.250	51.250
5.00	9.39	47.500	52.500
2.50	8.23	48.750	51.250
1.00	7.02	49.500	50.500
0.50	6.27	49.750	50.250
0.10	4.91	49.950	50.050
0.05	4.44	49.975	50.025
0.00	0.50	50.000	50.000

TABLE 3.1: Parameters for constructing ellipsoidal and interval uncertainty sets of different sizes

$$\boldsymbol{Ax} \leq \boldsymbol{C} \tag{3.6}$$

$$0 \leq \boldsymbol{x} \leq \boldsymbol{d} \tag{3.7}$$

$$\sum_{j \in s} x_j \leq b_s \qquad s \in \mathcal{S} \tag{3.8}$$

Inequality 3.6 represents the capacity constraint, inequality 3.7 the demand constraint, and 3.8 the booking limits constraints for realised sales $\boldsymbol{x}$. We first describe the simple case of partitioned booking limits and then detail the construction of the matrix for the nested booking limits control policy.

Partitioned booking limits

As seen in section 2.2 the partitioned booking limits control policy creates a booking bucket for each products thus each booking request concerns only the booking bucket of the product in request. As seen in section 3.2.1, the capacity constraint $\boldsymbol{A}$ is defined by $a_{kj} = 1$ when product j uses resource k and 0 otherwise. The capacity constraint matrix for the example seen in Figure 4.3 is presented in Appendix C.1. The demand constraint matrix is a simple $n \times n$ diagonal matrix with 1 along the diagonal and 0 everywhere else. The simplicity of partitioned booking limits is in the booking limit constraint matrix which has the exact same structure as the demand constraint matrix, since each product is a booking bucket on its own.

Nested booking limits

For the nested booking limits the matter is more complicated. Specifically, products j are grouped together according to their itinerary $\mathcal{I}$ and are ranked according to their fare value. Let $j \in s_{\mathcal{I}}$ denote the set of products j in the booking bucket s for itinerary $\mathcal{I}$. For each bucket s we rank products $j = 1, ..., n \in s_{\mathcal{I}}$ according to their fare value such as the product with the highest fare value, denoted by j_1, is ranked on the top and the product with the lowest fare value, j_n is ranked on the bottom. Their respective booking limits b_j are then characterised by ,

$$\begin{aligned} b_{j_1} &\geq b_{j_2} \\ b_{j_2} &\geq b_{j_3} \\ &\vdots \\ b_{j_n} &\geq 0 \end{aligned} \tag{3.9}$$

The matrix characterising the above relationship is found in Appendix C.2.

Given the above equations the relationship between the booking limits of all the products in offer in a booking bucket is characterised. More specifically we always have Fare Class 1 products on the top rank of the booking buckets. We then concern ourselves with the capacity constraints of these booking buckets. It is sufficient to characterise what resources k of the network Fare Class 1 products are utilising, since booking limits of products that ranked lower and use the same itinerary $\mathcal{I}$ will be smaller by equations 3.9. This incidence matrix is found in Appendix C.2.

Hence we only now need to specify the nested booking limits, N_j are less than or equal to the capacity of the network resources they utilise. To achieve this we utilise the above relationship between booking limits of products and the matrix identifying what legs k of the network, Fare Class 1 products are utilising. Let $\mathcal{I}_k$ $k = 1, ..., m$ denote the itineraries of the products that only use a resource k of the network. It is sufficient to specify the capacity constraints only for Fare Class 1 products $j_1 \in \mathcal{I}_k$,

$$N_{j_1} \leq C_k - \sum_{j \in s_{\mathcal{I}} : j \leq j_1} b_j \qquad \forall\, k = 1, ..., m \tag{3.10}$$

3.3 Genetic Algorithm

This section gives is an overview of the genetic algorithm that was designed and implemented to perform the outer optimisation to the robust formulations described in section 3.2.1 in this chapter and Chapter 5 which forms the basis of the optimisation for the cruise application, as described in section 5.2.2. First, we present an overview of the genetic algorithm procedure, and then we detail the design features and modelling choices we have made and implemented on the genetic operators to tailor the algorithm to the problems we are investigating. In the final part, we present the hyper-optimisation of the parameters we have conducted to fine-tune the algorithm's performance.

3.3.1 Motivation

Genetic Algorithms (GA) are search-based meta-heuristics for solving both constrained and unconstrained optimisation problems, classified under the broader class of Evolutionary Algorithms. They were introduced and developed by Holland et al. (1992) and his students in the 1960s and 70s. As the name suggests, these algorithms are inspired by biological evolution and are designed to mimic the natural selection process. They are unique among heuristics because of their defining characteristic of using a group, called a generation or a population, of candidate solutions, called individuals or chromosomes, instead of a single point, as is the case with other heuristics.

Our motivation for using a genetic algorithm arises from the method's ability to deliver a 'good enough' solution 'fast enough'. It can perform better than random local search as it exploits historical information while it provides a list of possible solutions instead of a single answer as seen in our results in table 4.18. In general, it is a faster and more efficient method than traditional search algorithms as it does not require any derivative information and can handle complex real-life problems with a vast search space and a high number of parameters. Furthermore, it can optimise both continuous and discrete functions while also multi-objective problems.

On the other hand, we note some of the genetic algorithm's drawbacks. First, it is not suited for all kinds of problems. Other methods more efficiently solve straightforward problems or when derivative information is available. A second theoretical issue the genetic algorithm exhibits is that there are no guarantees on the optimality or the quality of the solution because it is a stochastic heuristic. Further to the theoretical hindrances, the genetic algorithm also poses problems if not implemented correctly, as the algorithm may not converge to the optimal solution. Even in cases where it is implemented meticulously, the fitness value is calculated repeatedly, which might be computationally expensive.

3.3.2 Overview

Evolutionary algorithms have been thoroughly researched since their introduction, and several articles and books have been written on genetic algorithms. In this section, we do not aim to present an overview of the field but merely outline the algorithm's

Term	**Description**
Fitness function	The objective function to optimise.
Individual, Genome, Chromosome	Any point to which you can apply the fitness function.
Fitscore	The value of the fitness function for an individual is its score.
Genes	The vector entries of an individual.
Allele	The value a gene takes for a particular chromosome.
Population, Generation	An array of individuals. An individual can appear multiple times in a population.
Parents	Certain selected individuals in the current population used to create individuals in the next generation.
Offspring, Children	Individuals created from parents.

TABLE 3.2: Genetic Algorithm terminology

implementation to the problems we are investigating. The interested reader can find some excellent overviews of the field by Mitchell (1998), Sivanandam and Deepa (2008) and Kramer (2017).

In general, genetic algorithms must make use of three 'genetic' operators; *selection*, *crossover* and *mutation*. At each iteration, these operators are employed to create the next generation of candidate solutions from the previous population. The *selection* operator selects the individuals to act as parents, the *crossover* operator then dictates how these parents combine to form children, and the *mutation* operator applies random changes to any individual crossover offspring to form the children of the new generation. Table 3.2 presents the most commonly used terms and their respective explanation.

Algorithm 2 presents the conceptual idea of our genetic algorithm. The algorithm produces a pool of candidate solutions by repeatedly modifying the current population of individuals. At each iteration, individual solutions are selected from the current population to be parents and are used to produce the next generation's children. These new solutions are evaluated, and the selection, creation and evaluation cycle is repeated. Over successive generations, the population 'evolves' towards an optimal solution.

The genetic algorithm has five main stages. In the *Initialisation* stage, the algorithm creates the initial generation of chromosomes. This generation is randomly sampled or evaluated using a good approximation technique such as a constructive heuristic or the network approximation methods discussed in section 2.4.2. The goal is to construct an initial well-distributed generation in the solution space. A well-distributed initial population means that the algorithm receives information from the whole solution space and is more likely to avoid local optima.

Then we evaluate the fitness of each candidate solution in the initial population. Each chromosome in the current generation is assessed in this stage of the algorithm. The evaluation of solutions is carried out via a fitness function f that always depends on the application modelled. This stage is of extreme importance for two reasons; firstly, it links the genetic algorithm to the original problem that it is attempting to solve, and

Algorithm 2: Outline of our simple genetic algorithm

```
Initialization;
Create random initial population;
while Termination conditions are not met do
    for Evaluation do
        Score each individual of the current population by computing its fitness score;
    end
    Generation;
    for Selection do
        Scale the raw fitscores;
        Select elite individuals to be passed to the next population;
        Select individuals to enter the mating pool;
    end
    for Crossover do
        Select parents from mating pool via the roulette wheel method,
          $\Pr(X = x) = \frac{f(x)}{\sum_{i=1}^{n} f(x_i)}$;
        Create offspring from parents;
    end
    for Mutation do
        Mutate offspring genes to create children;
        Mutate parents genes to create children;
    end
    New Generation = [elite individuals; mutated crossover offspring; mutated parents];
end
```

secondly, it is the algorithm's operator that usually consumes the most computational time. This is also true in our investigation. The fitness functions that we employ are the minimax regret and maximin revenue as seen in formulations 3.2a and 5.2. As discussed, these are functions of high complexity and are considered NP-hard.

Once the chromosomes of the current generation are evaluated, we are ready to proceed to the creation of the next generation. The new generation is created at the *Generation* stage. This happens via the three genetic operators; selection, crossover and mutation.

First, the associated fitness values of each individual are used to determine the *mating pool*. The mating pool is a subset of the current population of individuals chosen by the *selection operator*. Members of the mating pool can be used in two ways; they can pass intact directly to the new generation or be chosen as parents to be transformed by the *crossover operator*. To choose the candidate solutions to be the parents of the new generation, one can simply choose the solutions with the highest fitness scores, referred to as the 'elitist' approach, or choose stochastically by assigning a higher probability of being chosen to the solutions with higher fitness scores, the 'roulette wheel' approach. These methods are discussed in further detail in section 3.3.3.3.

Once we have the selected solutions that will act as parents to the new generation, we move to the *Crossover* operator. Here we cross subsets of parents to create new offspring. The crossover operator combines schemata of different chromosomes to create

Term	Symbol	Description
Bit/Gene	g	An element of a chromosome $\boldsymbol{c}$
Candidate solution/ Chromosome/ Individual	$\boldsymbol{c} = \{g_j \mid j = 1, ..., n\}$	A column vector with n genes representing a candidate solution to the problem
Generation/ Population	$G_t = \{\boldsymbol{c}_i \mid i = 1, ..., N\}$	A set of $\boldsymbol{c}$ at iteration t
Fitness score	$f(\boldsymbol{c}_i)$	Fitness score of the i^{th} $\boldsymbol{c}$
Parent	$\boldsymbol{p}_i$	A $\boldsymbol{c}$ selected by the `selection function` to pass intact in G_{t+1} and can also be used as a parent of new chromosomes
Offspring	$\boldsymbol{o}_i$	A $\boldsymbol{c}$ produced by the `crossover function` but not evaluated by the `mutation function`
Child	$\boldsymbol{c}_i^{new}$	A $\boldsymbol{c}$ produced for G_{t+1} that has undergone both `crossover` and `mutation`
Crossover Rate	p_c	The proportion of new $\boldsymbol{c}$ to be created by the `crossover operator`
Mutation Rate	p_m	The probabibilty that a $g_j \in \boldsymbol{c}_i$ is randomly adjusted

TABLE 3.3: Genetic algorithm basic terminology

new chromosomes with better fitness scores, a process inspired by the natural reproduction phenomenon. There are several options as to carry out the crossing of the parents, and these are discussed in more detail in section 3.3.3.4.

Once this step is completed, we perform the *Mutation* stage where genes of offspring are randomly altered. The mutation operator serves many purposes, including ensuring the feasibility of potential solutions. The design of the mutation function can be as free as the developer's choosing, but it will always aim to mutate offspring genes in some probabilistic way. We discuss our mutation function implementation in section 3.3.3.5.

Once the mutation operations are complete, we have a new generation of candidate solutions. The fitness function then evaluates these in the *Evaluation* stage, and the termination conditions are checked. Multiple conditions can terminate the algorithm, often referred to as *stopping criteria*. Common criteria are a maximum number of iterations to be reached, the fitness value improvement rate is lower than the accepted tolerance, and the run time exceeds the allocated limit. The *Termination* stage checks if these conditions are satisfied. If any of the stopping criteria is true, then the algorithm terminates. Otherwise, the evaluation and generation of new individuals are repeated until any of the termination conditions are met.

3.3.3 Implementation of genetic operators

In this section we present a more detailed description of the genetic operators. To further aid the narrative, we introduce the notation in Table 3.3. The table maps key notation to terminology.

Algorithm 3: Pseudo-code for creating initial population

Inputs : options

Outputs: initialPopulation, initialPopulationScores

```
if options.initialPop is empty then
    initialPopulation = zeros(options.populationSize, options.genomeLength);
end
for $i = 1$:options.populationSize do
    for $j = 1$:options.genomeLength do
        sampleDemand(:, j) = normrnd(options.mean(j), options.std(j), 1, 1000);
        demandPoint(:, i) = quantile(sampleDemand(:, j), randi([25,75])/100);
    end
end
for j = 1:size(sampleDemand, 1) do
    $d$ = demandPoint(j, :);
    Solve DLP with demand $d$;
    initialPopulation = DLPsolutions;
    initialPopulationScores = DLPobjfun;
end
```

3.3.3.1 Initialisation

The problem we are employing the genetic algorithm to solve is a seat inventory control problem. Therefore the initial generation is constructed from column vectors representing booking limits for products on offer. Each gene, $g_j \in \boldsymbol{c}_i$, represents the booking limit for product j where a product is an Origin-Destination-Fare combination. Let $\boldsymbol{c}_i$ denote a chromosome in the generation, $G = \{\boldsymbol{c}_i \mid i = 1, .., N\}$ where N is the generation size. The initial booking limits population can be created using any of the approximations to the network models we described in section 2.4.2.

To initialise the genetic algorithm, we need to pass an initial population matrix where each member of the population is a candidate solution. It is generally recommended that the initialisation procedure randomly covers the whole solution space or models and incorporates expert knowledge. Our treatment considers the measure of robustness being optimised (maximum regret or minimum revenue) and constructs a different initial population accordingly.

The minimax regret principle is a measure of robustness that concerns itself with the average performance of the booking limits. Therefore we adopt an approach where we construct a randomly generated initial matrix intending to cover as much of the solution space as possible. We produce booking limits by solving the Randomised Linear Programming (RLP) model (see section 2.4.2), given the generated random demand realisations from the associated distributions to each product on sale. The demand distribution parameters for our airline network example are given in Appendix A.

On the other hand, the maximin revenue evaluates the performance of booking limits at the worst possible scenarios, i.e. the lowest demand points. Hence, an initial matrix with booking limits generated from optimistic demand realisations provides no helpful information to the algorithm and might direct the search towards local optima away from the true global optimum. Therefore, the initial matrix is generated using the DLP solutions to random demand realisations that are within the lower 50%. We allow demand for each product to vary up to the 50th percentile of its associated distribution.

Thus the initial population matrix is generated by the following steps; for each allele (product on sale) of each chromosome (candidate solution), we draw a random sample of 1000 demand points. The random sample is drawn from the normal distribution using the product's mean and standard deviation. To create an instance to solve, we need to assign a demand point value to each product. This value is calculated as a random quantile from the uncertainty interval chosen by the modeller; in our case, we chose the interval $[0.25, 0.75]$ for the minimax Regret and $[0, 0.5]$ for the maximin Revenue. Once a demand point estimate has been assigned to every product, we truncate any possible negative values to 0. The final step is to solve a deterministic DLP for each demand realisation drawn. Algorithm 3 shows how the initialisation procedure was coded.

Our default options set the generation size, N, to 170. This value was chosen after performing the hyper-optimisation of the genetic algorithm's parameters. Detailed description of the procedure followed, are given in section 3.3.4.1. Therefore 170 booking limits are produced using algorithm 3 for each case and considered candidate solutions.

3.3.3.2 The Evaluation operator

The evaluation operator utilises the fitness function to assess the quality of the solutions, c_i, the genetic algorithm has generated. There are three main ways to evaluate chromosomes. The simplest one assigns the fitness score the value calculated by the fitness function employed. This strategy underperforms when the chromosomes' fitness values are close together as the best and worst candidate solutions are likely to produce the same number of offspring. Two standard techniques used are windowing and linear normalisation to avoid this situation. In our design, we opt for normalisation of the scores as it is a well-understood and straightforward procedure that works well in practice.

Algorithm 4: Pseudo-code for a population evaluation

```
Inputs  : Evaluation formula, Current/Last population, Last fitscores
Outputs: Current fitscores
1 Separate new and past seen individuals;
2 Identify unique newly seen individuals;
3 Evaluate fitness of unique newly seen individuals;
4 Assign the scores of newly and past seen individuals to the fitscores array;
```

Several treatments of infeasible solutions are available,

1. ***Penalty functions*** - penalise the fitness score of infeasible solutions. This allows an infeasible solution that is close to optimal to still be considered for selection, albeit with a reduced score and therefore reduced probability of being selected as a parent.

2. ***Death penalty*** - If a solution is infeasible, assign the minimum score so that it has no chance of being reproduced or carried over to the next generation.

3. ***Repair Function*** - a third option for the treatment of infeasible solutions is to repair them. While strictly speaking, this is not done in the evaluation function; it is worth mentioning here. Introducing a repair operator that will enforce constraints is another option to eliminate infeasible solutions.

Minimising the number of fitness function calls is very important, especially if each call to the function is expensive, as is the case when solving a robust exact formulation. In our treatment we use both the death penalty and the repair function. If an offspring is strictly infeasible then it is assigned a minimum score so that it is eliminated from future generations. However, if the offspring violates a constraint partially, that is capacity is exceeded in one resource of the network and there is still room in another resource then we repair it with the aid of the mutation function. A more detailed description is given in section 3.3.3.5

3.3.3.3 The Selection operator: *Survival and Mating pool choices*

The best offspring solutions are selected to be parents in the new parental population to allow convergence towards optimal solutions. A surplus of offspring can be created, and the best of them can be chosen as members of the new generation. The selection can be performed based on a variety of rules:

1. ***Elitist*** - Selects the best solutions of the offspring solutions as parents.

2. ***Comma*** - Selects the μ best solutions from λ offspring solutions. This method has the limitation that good parents can be forgotten.

3. ***Plus*** - Selects the μ best solutions from λ offspring solutions and the μ old parents that led to their creation.

4. ***Roulette Wheel*** - Also known as fitness proportional selection selects parental solutions randomly. The probability of being selected depends on the fitness of a solution; the relative fitness is normalised with the sum of all fitness values in the population. This means that every solution has a chance of being selected while stronger solutions are more likely to move on, but good solutions can be forgotten.

5. ***Tournament Selection*** - A subset of solutions is selected randomly, and within this competition subset, the best solutions are selected as new parents. The second step can be implemented with proportional fitness selection as well.

It is worth mentioning that methods that allow good solutions to be forgotten are not necessarily wrong. The reason for this apparent contradiction is that forgetting some good solutions may allow the algorithm to escape from local optima.

Furthermore is important to distinguish between the two uses of the selection operators. When using the selection as a mechanism to choose the parents of the new generation, it is called ***survival selection***. The selection operator determines which solutions survive and which solutions die. This perspective directly implements Darwin's principle of survival of the fittest. The second use of selection operators is to determine the specific parents from the mating pool used to produce offspring in the crossover operator. This operator is referred to as ***mating selection***. It makes sense to consider different mating selection criteria compared to survival selection criteria.

Algorithm 5: Pseudo-code for *Elitist* selection operator

```
Function gaSurvivalSelection()
    Inputs: children, fitScores, fitEvalTime, options
    Outputs: parents, parentScores, childless, worstScores
    nParents = round((1 - options.crossoverFraction) *
     options.populationSize);
    [sortedIndex] = sort(fitscores(:), 'descend');
    bestIndividuals = children(sortedIndex(1:nParents), :);
    bestScores = fitscores(sortedIndex(1:nParents));
    worstIndividuals =
     children(sortedIndex(nParents+1:options.populationSize), :);
    worstScores = tempScores(sortedIndex(nParents+1:options.populationSize));
    parents = bestIndividuals;
    childless = worstIndividuals;
    parentScores = bestScores;
end
```

We assume that the initial population passed is a good guess for the optimal solution; therefore, we aim to exploit the initial population as much as possible. Hence, in our treatment, we opted for the elitist method for the survival selection and the roulette wheel method for the mating selection. Algorithms 5 and 6 describe how those methods were implemented.

The number of individuals, $\boldsymbol{c}_i$, selected to pass intact to the next generation is implicitly determined by the crossover rate, p_c. Let O_{sel} denote the set of such $\boldsymbol{c}_i$. The set has size equal to the proportion of the current generation not affected by crossover, $|O_{sel}| = (1 - p_c) * N$. Further to that p_c determines the number of offspring to be created by the `crossover function`. Let O_{cross} denote the set of $\boldsymbol{c}_i$ that are created by crossover, then $|O_{cross}| = p_c N$.

The set O_{sel} is created via a simple elitist approach based on the individuals' fitness value as seen in algorithm 5. Given the current generation G_t, each chromosome, $\boldsymbol{c}_i$, in the generation is evaluated and assigned a fitness value, $f(\boldsymbol{c}_i)$. Then the chromosomes are sorted in descending order so that the chromosome with the highest score, $f^1_{\boldsymbol{c}_i}$ is placed first and the individual with the lowest score, $f^N_{\boldsymbol{c}_i}$, is placed last. The first $|O_{sel}|$ chromosomes are selected to pass into the next generation, G_{t+1}, intact.

Given that the elitist approach has created the set O_{sel}, the chromosomes in the set O_{cross} must now be created. To create $\boldsymbol{c}_i \in O_{cross}$ the crossover function described in section 3.3.3.4 is used. The crossover operator requires parent chromosomes $\boldsymbol{p}_i$ to be selected from the mating pool so that crossover between those parents can be carried out. The crossover parent selection is made via the roulette wheel strategy.

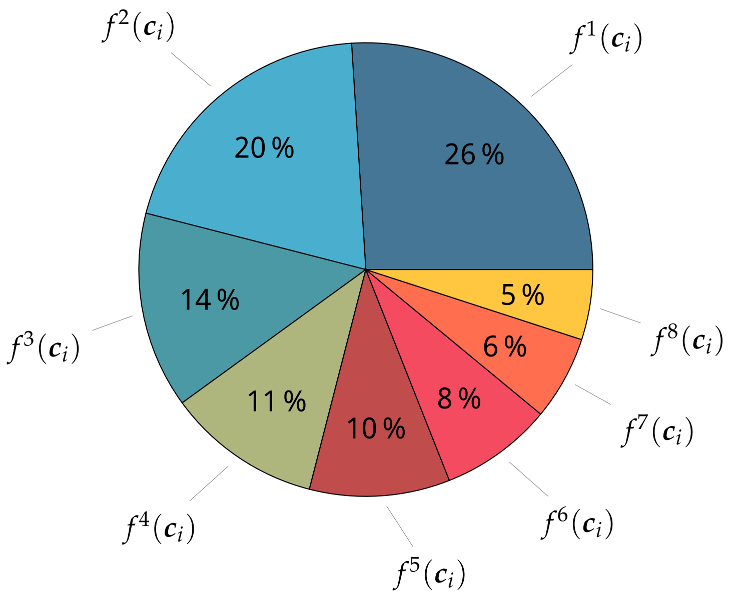


FIGURE 3.2: Graphical representation of Roulette Wheel parent selection for crossover operation

Roulette Wheel is a random parent selection procedure that favours $\boldsymbol{c}_i$ with high fitness score, $f(\boldsymbol{c}_i)$. The chance of a $\boldsymbol{c}_i$ to be selected as parent is directly proportional to its $f(\boldsymbol{c}_i)$. The procedure first calculates the total fitness of the potential parents, $f_{tot} = \sum_{i=1}^{|O_{sel}|} f(\boldsymbol{c}_i)$. Then fitness bins, b_k, are created such that the first bin covers the range between zero and the fitness score of the first chromosome in O_{sel}, the second covers the range between the first and second chromosome in O_{sel} and so on. $\boldsymbol{c}_i$, $[0, f(\boldsymbol{c}_i)]$. Then a random number, r, between $[0, f_{tot}]$ is generated. The first chromosome $\boldsymbol{c}_i$ that has bin value greater than the random generated number, $b_k > r$, is selected as parent. The procedure is repeated until the necessary number of parents are selected. The chance of a chromosome to be selected is equal to the ratio of its fitness score over the total fitness, $\frac{f(c_i)}{f_{tot}}$ hence chromosomes with higher fitness scores, have higher probability of being selected as parents. Algorithm 6 presents this procedure.

Algorithm 6: Pseudo-code for Parent Selection strategy: Roulette Wheel

```
1: Calculate total fitness $f_{tot} = \sum_{i=1}^{|O_{sel}|} f_{c_i}$
2: Create fitness bins $b_k = \sum_{i=1}^{k} f_{c_i}$ where $k = 1, ..., |O_{sel}|$
3: repeat
4:     Generate a random integer $r \in [0, f_{tot}]$
5:     Select the first individual $\boldsymbol{c}_i$ that has $b_k > r$ as parent $\boldsymbol{p}_i$
6: until Required number of parents are selected
```

3.3.3.4 The Crossover operator

The crossover operator allows the combination of the genetic material of two or more candidate solutions. There exist a number of ways to perform recombination of the parents. The options include:

1. ***N-point*** - the classic approach where each parent is split at n-points, and the parts are then swapped to form offspring.
2. ***Arithmetic*** - the mean value of the parents is taken for each gene.
3. ***Uniform*** - 0.5 ratio to randomly choose a gene from each parent.
4. ***Dominant*** - successively chooses each component from one of the parents based on a dominance rule.

We have implemented methods 1,2, and 3 and compared their performance after 25 runs of 100 generations each. In our setting, the Uniform crossover seems to perform best. The algorithms below present pseudo-code on how we have implemented each method.

The first method we present is the traditional *N-point* method. We set N to equal the number of fare classes as intuition dictates that the segments of the candidate solution that represent a different type of product should be grouped together. It is important to note that this method is particularly effective when using bit representation. In our setting, we are using a vector representation where elements are positive integers, and thus the performance of the method might not be as successful.

The second method we implemented, algorithm 8, is the Arithmetic crossover or intermediate crossover as it is also known. Here we take the component-wise arithmetic mean of the parents. The method allows for more than two parents to be used. Two variants were implemented: first, only two parents were chosen from the mating pool,

Algorithm 7: Pseudo-code for *Uniform* crossover

```
Function gaUniformCrossover()
    Inputs: parents, fitScores, options
    Outputs: offspring
    // Calculate # offspring to create and preallocate matrices for computational speed
    offspringToCreate = (options.populationSize - size(parents, 1));
    offspring = zeros(options.populationSize, options.genomeLength);
    offspring(1:size(parents, 1), :) = parents;
    // Create fitness bins
    scores = abs(scores);
    TotalFitness = floor(sum(scores));
    FitnessBins = [0; cumsum(scores)];
    // Initialise offspring creation index
    iNewGeneration = size(parents, 1) + 1;
    while offspringToCreate >= 1 do
        // Choose parents from mating pool using Roulette Wheel
        r = randi([1, TotalFitness], 2, 1);
        NextBin = interp1(FitnessBins, FitnessBins, r, 'next');
        index1 = find(FitnessBins==NextBin(1))-1;
        index2 = find(FitnessBins==NextBin(2))-1;
        Parent1 = parents(index1, :);
        Parent2 = parents(index2, :);
        // Create new offspring by calculating the component wise mean of the parents
        for genome = 1:options.genomeLength do
            prob = randi([0, 1], 1);
            if prob == 1 then
                offspring(iNewGeneration, genome) = Parent1(genome);
            else
                offspring(iNewGeneration, genome) = Parent2(genome);
            end
        end
        // Update offpsring counter and index
        offspringToCreate = offspringToCreate - 1;
        iNewGeneration = iNewGeneration + 1;
    end
end
```

and second, four parents were chosen. We opted for four parents in the second variant as this would allow more input and avoid the problem of averaging out between solutions with significant variance.

A two-point crossover function is used to create new chromosomes, $\boldsymbol{c}_i^{new} \in O_{cross}$, from selected chromosomes with high fitness scores, $\boldsymbol{c}_i \in O_{sel}$, in the hopes of creating even better solutions. To create these new chromosomes we first choose the parent chromosomes using algorithm 6 as described in section 3.3.3.3.

A two-point crossover splits the parent chromosomes at two points, creating three sections. We choose a two-point crossover because we can determine cross-sections equivalent to the fare classes of the seat inventory problem. This is desirable because we are

Algorithm 8: Pseudo-code for two parent arithmetic crossover

```
Function gaArithmeticCrossover()
    Inputs: parents, fitScores, options
    Outputs: offspring
    // Calculate # offspring to create and preallocate matrices for computational speed
    offspringToCreate = (options.populationSize - size(parents, 1));
    offspring = zeros(options.populationSize, options.genomeLength);
    offspring(1:size(parents, 1), :) = parents;
    // Create fitness bins
    scores = abs(scores);
    TotalFitness = floor(sum(scores));
    FitnessBins = [0; cumsum(scores)];
    // Initialise offspring creation index
    iNewGeneration = size(parents, 1) + 1;
    while offspringToCreate >= 1 do
        // Choose parents from mating pool using Roulette Wheel
        r = randi([1, TotalFitness], 2, 1);
        NextBin = interp1(FitnessBins, FitnessBins, r, 'next');
        index1 = find(FitnessBins==NextBin(1))-1;
        index2 = find(FitnessBins==NextBin(2))-1;
        Parent1 = parents(index1, :);
        Parent2 = parents(index2, :);
        // Create new offspring by calculating the component wise mean of the parents
        offspring(iNewGeneration, :) = round(mean([Parent1; Parent2],1));
        // Update offpsring counter and index
        offspringToCreate = offspringToCreate - 1;
        iNewGeneration = iNewGeneration + 1;
    end
end
```

exploiting the structure of the problem to most likely preserve the capacity constraint of the problem.

The new generation, G_{j+1}, is created by considering all possible combinations $\perp Sk$ where $k = 2$ of the set S of parent candidate solutions. For each pair of parents, the `crossover function` divides each candidate solution at n points (in our case 2) creating $n + 1$ fragments. It then crosses the fragments of the parental solutions to create new candidate solutions as seen in Figure 3.3.

3.3.3.5 The Mutation operator

Mutation operators change a solution by disturbing them based on random changes. The strength of this disturbance is called the mutation rate.

In the `mutation function` we check the offspring created by our crossover function for feasibility. Any offspring that are feasible are left as they are. Offspring that are infeasible are checked to identify candidate solutions that can be turned feasible either

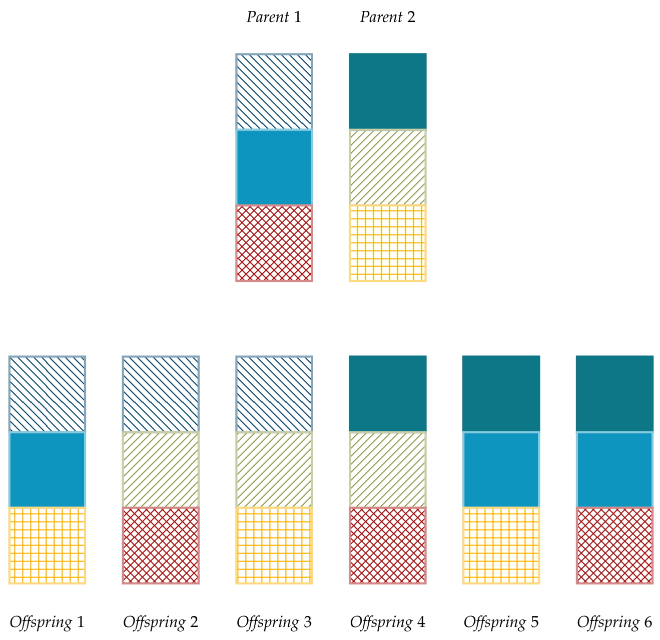


FIGURE 3.3: Representation of the crossover operation

Algorithm 9: Pseudo-code for the mutation operator

```
index1 = any(infeasible offspring);
if $sum(index1) > 0$ then
    cutoffspring = cut[offspring(index1, :)];
    offspring(index1, :) = cutoffspring;
end
if all offspring are optimal then
    geneSwappedoffspring = geneSwap[offspring];
    children = geneSwappedoffspring;
else
    index2 = any(optimal offspring);
    geneSwappedoffspring = geneSwap[offspring(index2,:)];
    offspring(index2, :) = geneSwappedoffspring;
    index3 = any(suboptimal offspring);
    optimisedoffspring = optimise[offspring(index3,:)];
    offspring(index3, :) = optimisedoffspring;
    children = offspring;
end
```

by swapping allocations to products or by cutting a number of seats from an allocation to a product. The swapping and cutting operations are done randomly so that new solutions are generated and checked.

In general,

$$G_{j+1} = g(O_{sel} + O_{cross})$$

where $g(\cdot)$ is the mutation function described in section 3.3.3.5.

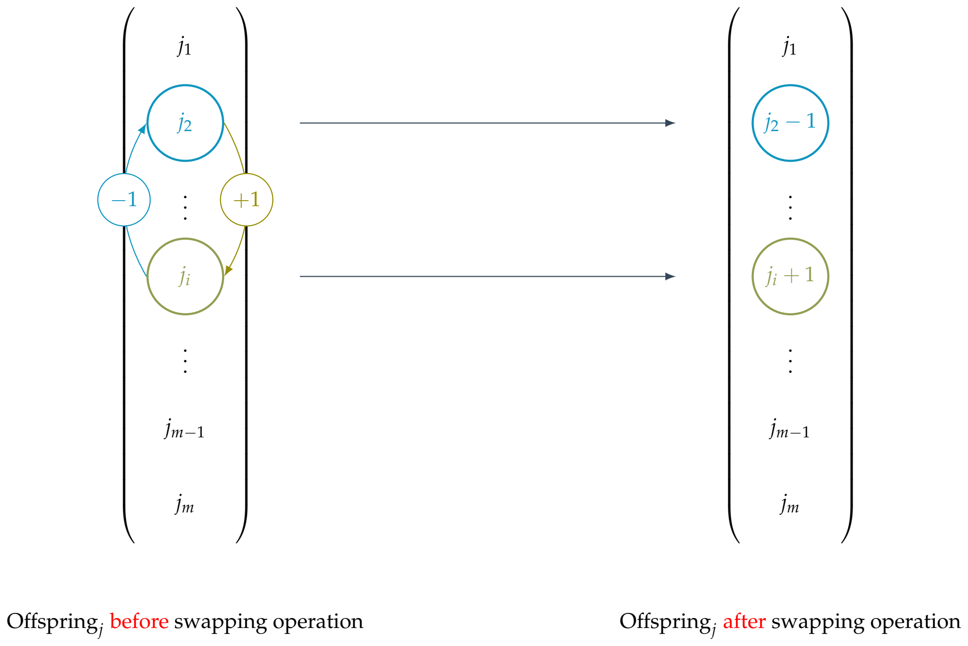

j_1
j_2
-1
$+1$
j_i
j_{m-1}
j_m
$j_2 - 1$
$j_i + 1$
Offspring$_j$ before swapping operation
Offspring$_j$ after swapping operation

FIGURE 3.4: Example of swapping in a candidate solution

3.3.3.6 Termination conditions

The termination conditions define when the main evolutionary loop terminates. Again, a number of options are available:

1. `maxIterations` - The most common termination condition is a predefined number of generations. The algorithm stops after running a predefined `maxIterations` number of iterations.
2. `maxTime` - The algorithm stops after running for an amount of time in seconds equal to `MaxTime`.
3. `MaxStallGenerations` — The algorithm stops when the average relative change in the fitness function value over a number of generations is less than Function tolerance.

As with most metaheuristics, the algorithm can terminate if any or a combination of the following criteria are met:

- A prespecified number of iterations has been completed. This is set to 60.
- A prespecified number of iterations has been completed without improvement in the fitness function. This is set to 20 iterations where the change in the best fitness function is less than the function tolerance which is set to 0.000001. Improvement in the function is taken from the average

$$f_{\max} - \bar{f}$$

3.3.4 Hyper-optimisation of the genetic algorithm's parameters

To optimise the performance of the algorithm we pay attention to the parameters of the genetic operators. Fine tuning these hyper-parameters can help make the algorithm more efficient in terms of computational speed, accuracy and performance. As expected, a single optimal value for each of these parameters for any type of problem does not exist. The optimal value of these parameters depends on the application in question.

Meta-Genetic algorithms are heuristics designed to optimise these parameter values. Such algorithms are computationally expensive as they employ a genetic algorithm to optimise the parameters of the genetic algorithm used to solve the original problem. Furthermore, in our specific application the fitness function is a second order conic problem which is tractable but can be computationally demanding. Hence we avoid such treatment and instead apply a simple grid search. Grid search seems appropriate because the search space is discrete and an exhaustive search in a small subset of possible values for these hyper-parameters is tractable.

To gauge the performance of the algorithm we pay attention to 4 parameters: population size N, crossover fraction p^c, crossover operator and selection operator. We optimise two measures: the objective function value and the average simulated revenue generated.

We apply a grid search approach for each parameter where we fix all other parameters to a certain value and only alter the value of the parameter in question. The initial set of parameters is indicated by bold while the range of the investigated values is also given in Table 3.4.

Typical sets are $C = \{p_c | 0.5 \leq p_c \leq 1\}$ for the crossover parameter and $M = \{p_m | 0.001 \leq p_m \leq 0.05\}$ for the mutation function. In this section we present the experiment design we employed and the results it yielded in optimising the parameters of the genetic algorithm.

Parameter	**Value**
Population Size	**10** : 10 : 200
Crossover Fraction	**0.1** : 0.1 : 1
Crossover Operator	**Arithmetic mean**, Uniform, N-point
Selection Operator	Elitist, Roulette Wheel

TABLE 3.4: Investigate parameter values for the GA

The experiment setting was 20 runs of the genetic algorithm on each specified combination of parameters and then the objective values as well as the simulation results of the yielding 20 optimal solutions are recorded. These simulation results and objective values are then compared to decide upon the optimal set of parameters. Sections 3.3.4.1, 3.3.4.2, and 3.3.4.3 present the results of the grid search experiment performed.

3.3.4.1 Population Size

Grefenstette (1986) argues that population size is critical to the efficiency of a genetic algorithm. He asserts that small populations might lead to only suboptimal solutions because not enough of the search space is covered, while too large populations might demand too high computation times, thus unfavourable for the algorithm's designer.

To evaluate these statements and choose the population size, N, we run the GA for multiple values $N \in \{10, 20, \ldots, 200\}$ and plot the results as demonstrated in Figure 3.5. For each value of N, 20 runs of the algorithm are performed, where each run has the same initial population. A red dash represents the objective value of each run, while the blue dot represents the mean objective value for that N.

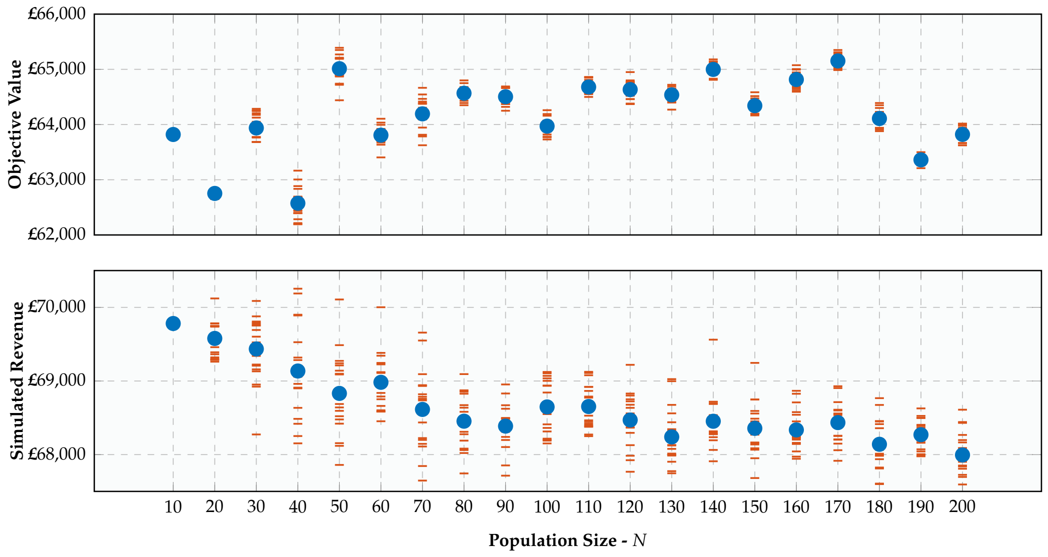


FIGURE 3.5: Performance comparison of population size

Figure 3.5 does not yield an optimal value for the population size to use in our application. We do, however, inspect the following: first, it is clear that minimal values for N (10,20) are very inflexible. They do not provide enough coverage of the solution space and converge too quickly to a suboptimal solution, as demonstrated by the smaller objective values and small variability among them. Second, slightly larger N (30-70) suffer because the algorithm does not always converge. This is evident by the high variability in the objective values, denoted as longer red tails in the graph. Third, we argue that performance is enhanced by having a population greater than 100. For $N \in [100, 170]$ we observe increasing mean values and shorter tails; that is, the algorithm is more stable and finds better optimal solutions. In the experiment, the best performance was achieved at a population size of 170 chromosomes. Finally, it is interesting to draw attention to the slight performance deterioration for values N greater than 170. This is perhaps because too much variability is added.

While with the objective value measure, we were looking for the population size that will deliver the highest and most precise results, with average simulated revenue, high returns are not the desired outcome. This is because the objective function of the GA is to maximise the minimum revenue, that is, to find the booking limits that will generate the best revenue in the worst-case scenarios of demand, not the highest revenue on average. Furthermore, small populations yield results closer to a DLP solution to a single demand realisation. These solutions are the same for all 20 runs and perform the same in the simulation, yielding higher average revenue than solutions with a higher initial population size that consider more of the solution space and are more effective in going along the trade-off line of average vs minimum revenue. Figure 3.5 demonstrates this behaviour.

3.3.4.2 Crossover Fraction

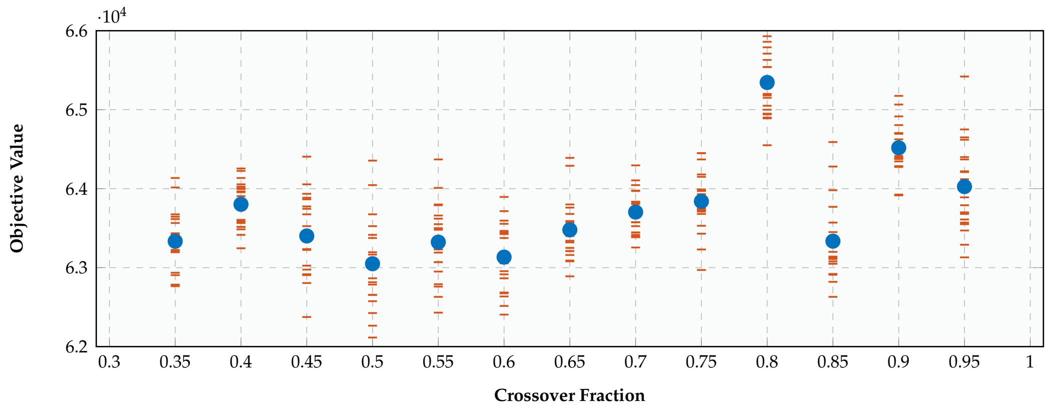


FIGURE 3.6: Performance comparison of alternative crossover fractions

Another important parameter is the crossover fraction which defines the percentage of the population that will be created via the crossover operator. It can take values in (0, 1) where having a value of 0 would mean that no crossover occurs while having a value of 1 would mean that no chromosome from the starting population survives intact to the next generation, both being undesirable properties and thus excluded by design. Figure 3.6 presents the results of our experiment. Here the value setting of 0.8 seems to be clearly outperforming the rest.

3.3.4.3 Crossover Operator

The crossover parameter is the first and perhaps most important genetic operator. Figure 3.7 clearly demonstrates that the choice of the operator has a significant effect on the objective value by the clear gaps between the objective value lines for each of the three possible operators.

It is easy to ascertain that the *uniform crossover* performs best in our experiment as in all but one of the 20 runs of the algorithm, it has achieved the highest objective value. Furthermore, on average it outperforms the *arithmetic crossover* by 1200 units while it almost yields 2900 units more compare to the *N-point crossover*.

3.3.5 Convergence

Here we present some figures that illustrate the convergence of the implemented algorithm. Figure 3.8 the minimisation of the maximum Regret, formulation 3.2a using an ellipsoidal uncertainty set. We present two algorithm runs where the parameters are set to the values defined by our grid search.

On the x-axis, we have the number of iterations the algorithm has completed, and on the y-axis, the objective value. From each generation, we record the minimum, mean and maximum scores. The figure illustrates three distinct features:

- First, there exists a clear downward trend. This is the desired behaviour as the objective was to minimise the maximum Regret. One can see that the best answer we obtain at the end of the algorithmic run is better than the best solution in the original population.
- Second, the spread of the summary scores diminishes as iterations progress. Again this is the desired effect as we want our algorithm to convergence to a single solution.
- Third, it is clear that the score lines are not monotonically decreasing. The absence of monotonicity is an indication that the algorithm successfully continues

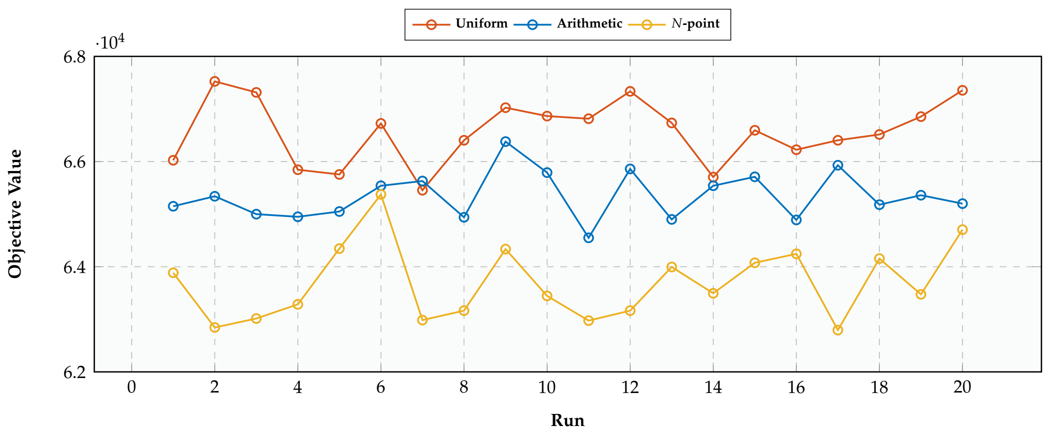


FIGURE 3.7: Performance comparison of crossover operators

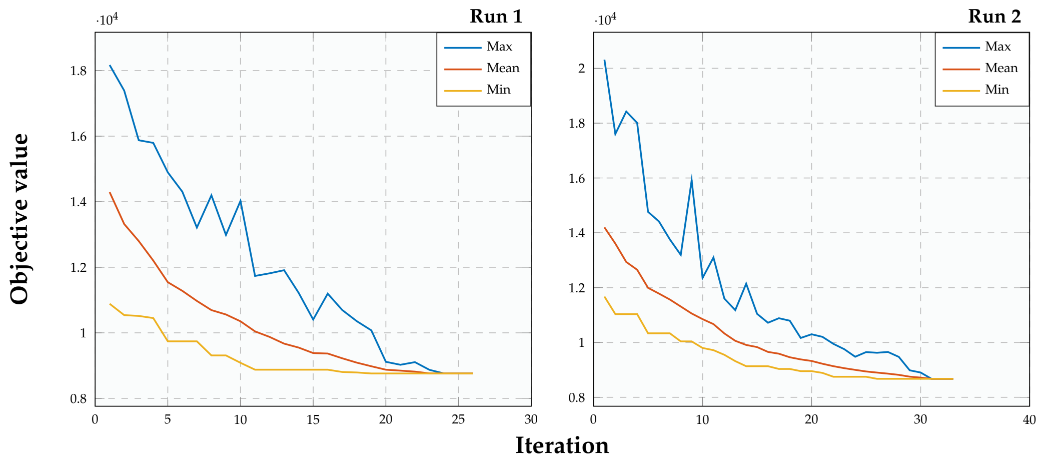


FIGURE 3.8: GA convergence of the minimax regret criterion under ellipsoidal uncertainty using the GUROBI solver

the exploration of the solution space throughout the search, and the random permutations to the individuals that might yield worse objective values than children from the previous generations might still enter the next generation to be calculated.

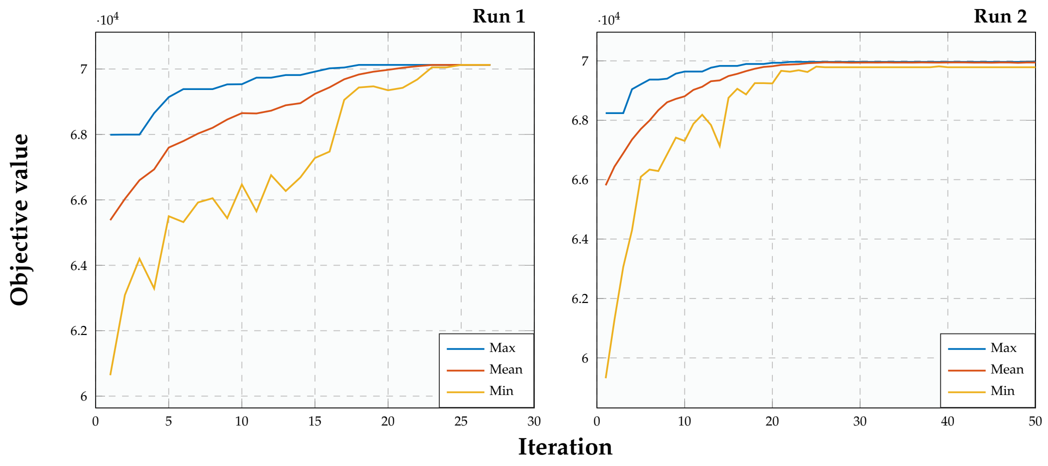


FIGURE 3.9: GA convergence of the maximin revenue criterion under ellipsoidal uncertainty using the GUROBI solver

Similarly figure 3.8 presents the maximisation of the minimum Revenue, formulation 3.2a using an ellipsoidal uncertainty set. The same observations made for the minimisation of the maximum Regret are also made here, with the difference that the trend now is upwards as expected.

3.4 Conclusion

In this chapter, we reconsider the robust formulation introduced by Perakis and Roels (2010). Our contributions to the treatment of the robust network models is twofold; first, we use ellipsoidal uncertainty sets to characterise demand, in addition to the polyhedral uncertainty sets originally proposed. The use of Ellipsoidal sets in the optimisation of the inner problem leads to increased revenue but at the expense of increased computation times. Second, we design and implement a genetic algorithm to optimise the outer problem. The genetic algorithm results surpass the revenue gained by the local gradient descent initially suggested Perakis and Roels (2010), but it also significantly increases computation time.

Chapter 4

Numerical Results

This chapter presents our numerical experiments and compares them to those presented in de Boer et al. (2002) and Perakis and Roels (2010). We compare partitioned, nested, and bid-price controls for the booking limits suggested by the DLP, RLP, EMR models described in section 2.4.2 and the two robust measures minimax regret and maximin revenue, seen in section 3.2.1, using both polyhedral and ellipsoidal uncertainty sets. To compare them, we perform a simulation of the booking horizon, which is described in detail in section 4.1 and the booking request acceptance procedure, which is detailed in section 4.2.

The chapter is organised into six sections: The first two sections describe the demand model and the booking acceptance algorithms we implemented to simulate the booking period. The third section introduces the small forward serial network on which our comparisons are built. The following three sections detail the numerical results of the seven models we investigate for each of the three booking controls we implement: partitioned, nested and bid-prices, respectively. The final section summarises the conclusions we draw from our results.

4.1 Simulating The Booking Period

Before we go on to present the allocation of seats to the different booking buckets calculated by the different models described in Sections 2.4.2 and 3.2.1, we first introduce the demand model and the algorithms that enforce the booking control policies described in Section 2.2. This provides a testing environment for the booking period and allows us to calculate the expected number of accepted and denied booking requests according to the booking control policy we wish to implement.

Demand for the different products is usually assumed to be stochastic, thus simulating the booking horizon is a very important step in solving NRM problems. It provides a powerful tool in emulating the arrival process of itinerary requests within the booking period. This is achieved by modelling the *order* and *intensity* of booking requests arrivals, while group bookings and overbooking are features that can be incorporated into the model. This is of importance because it provides a way to evaluate the effectiveness of the seat allocations produced by the approximation models and our optimisation procedure in life-like situations and it can illuminate the effect of the different booking control policies on the expected revenue.

The process of arrival of booking requests is an important factor in the simulation of the booking period. Mathematical models have been developed in an effort to emulate the intensity with which booking requests arrive. We follow Weatherford et al. (1993) approach who try to capture this variability in the booking period by simulating the arrival process using a Non-homogeneous Poisson process (NHPP). His approach has been widely used in the academic literature to simulate the arrival process of booking requests. de Boer et al. (2002), Bertsimas and De Boer (2005), Perakis and Roels (2010) are just a few of the academics that follow the same approach. Mathematically the process is characterised by the equation,

$$\lambda_j(t) = B_j(t)G_j \tag{4.1}$$

where the $B_j(t)$ follows a standardised beta distribution

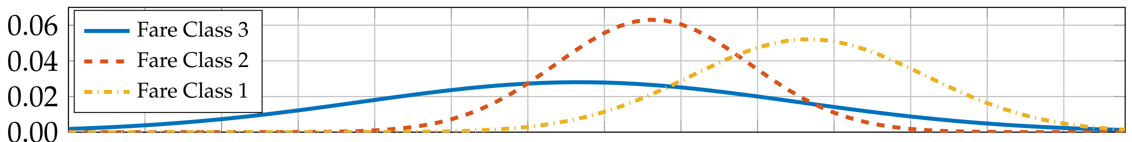
$$B_j(t) = \frac{1}{T}\left(\frac{1}{T}\right)^{\alpha-1}\left(1-\frac{t}{T}\right)^{\beta-1}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$$

where T is the length of the booking period, with scale parameter α and shape parameter β. G_j follows a gamma distribution with shape parameter p and scale parameter q.

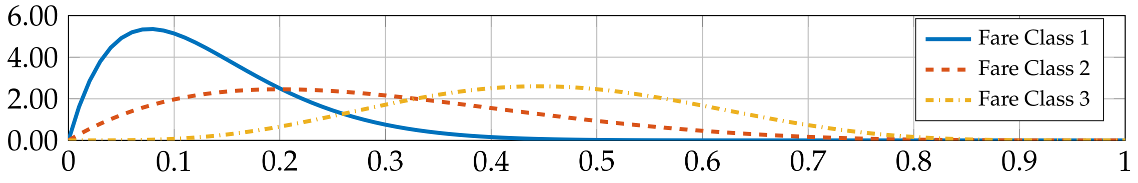
In this model the arrival intensity $\lambda_j(t)$ depends on two factors. The Expected Total number of bookings, given by G_j a gamma random variable, and the Arrival pattern, given by $B_j(t)$ a standardised beta distribution. G_j follows a gamma distribution because as de Boer et al. argue it has been shown to fit airline booking data. $B_j(t)$ is modelled by a beta distribution because of its flexible shape and its limited range (it is only valid between 0 and 1).

This model assumes that demand for each product is independent and that sales of different products do not affect future demand.

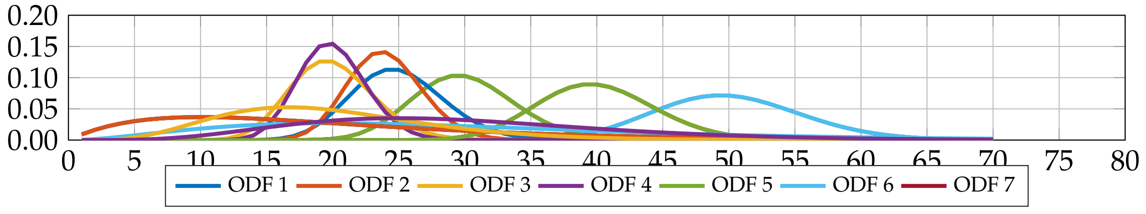
Figure 4.1 displays the plots of three distributions that can be used to model the arrival process of booking requests during a booking period. The three distributions parameters are as in Appendix A. The parameter values are identical to de Boer et al. (2002) for the Γ and B distributions while the Normal distribution parameter values, μ and σ are as in Perakis and Roels (2010).



(A) Booking requests per Fare Class modelled as Normal Distribution



(B) Pattern of arrivals per Fare Class modelled by Beta Distribution



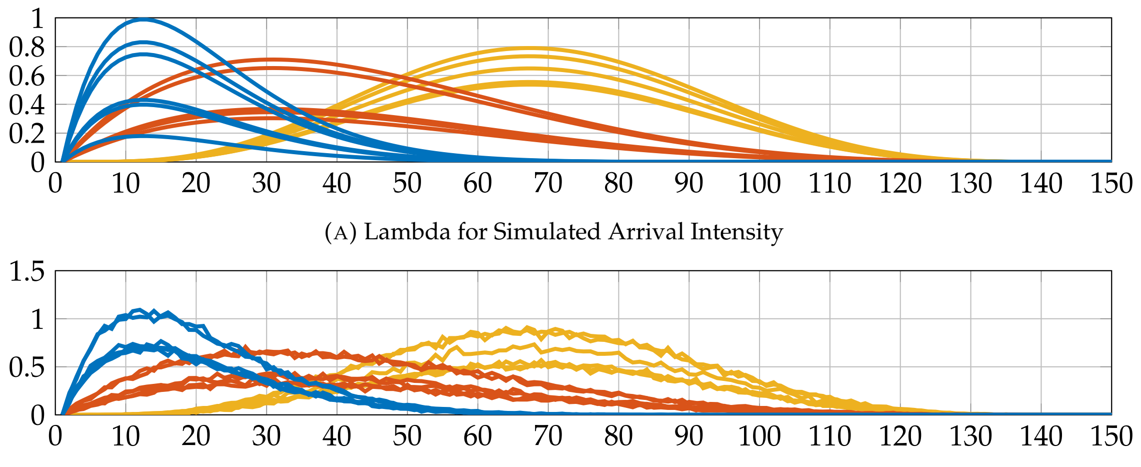
(C) Booking requests per ODF combination modelled as Gamma Distributions $G(p, y)$

FIGURE 4.1: Modelling Booking requests

It is evident from Figure 4.1a that simulating the arrival process of booking requests as Normal random variables is not a good approximation to reality. Firstly, the left tails of the distributions are taking negative values. This means that there is a small probability that a negative number of booking requests arrives in the simulation, which does not hold in reality. Secondly, the shape of Normal Distribution does not reflect the arrival pattern of the different Fare Classes according to historical data. In this application high fare customers arrive late in the booking period while low fare customers arrive early and are unlikely to book towards the end of the period. Fare Class 2 customers are spread in between the previous two categories. Figure 4.1b displays the desired behaviour which is achieved with the parametrisation given to the Beta distribution as seen in Appendix A. Finally, Figure 4.1c shows the expected number of booking requests per ODF product to be arriving in the booking period modelled as a Gamma Distribution.

As Equation 4.1 dictates, we combine the Gamma distributions modelling the expected number of booking requests per ODF (*intensity*) with the Beta distributions modelling the pattern of arrivals (*order*).

Figure 4.1 presents the MATLAB simulation for the non-homogeneous Poisson arrival process in two graphs. The first graph presents the *arrival intensity*, $\lambda_j(t)$, as generated by the beta-scaled gamma distributions seen in Figures 4.1b, 4.1c. The second graph, Figure 4.2b, is the Mean Arrival Rates generated from a Poisson Arrival Process with rate $\lambda_j(t)$.



(A) Lambda for Simulated Arrival Intensity

(B) Mean Arrival Rates

FIGURE 4.2: Comparison of Mean Arrival Rates, lambda parameters

4.2 Booking Acceptance Process Algorithms

A further important factor in simulating the booking period before a plane's departure is the booking acceptance process. Here we specify the rules that govern the acceptance or decline of a booking request. These processes form part of reservation systems and are derived from the different types of booking controls examined in Section 2.2. In this section, we present three algorithms that are designed to represent partitioned booking limits, nested booking limits and bid-price controls.

Mathematical Notation To introduce the algorithms some notation is needed. Let $\boldsymbol{x} = [x_j \mid j = 1, ..., n]$ denote the vector of seat allocations to each product j. $\boldsymbol{C} = [C_k \mid k = 1, ..., m]$ is the remaining capacity for each leg k of the network. A booking request for a product j arriving to the system at time t is represented by $\boldsymbol{P}(t) = [P_j(t) \mid j = 1, ..., n]$ and the number of accepted booking requests per product is denoted by the vector $\boldsymbol{q} = [q_j \mid j = 1, ..., n]$. The itinerary of the booking request for product j is represented by I_j. Let the dual prices of the capacity constraints be denoted by π_k and the threshold prices for each product j be $\boldsymbol{T}_j$.

4.2.1 Partitioned Booking Limits

Algorithm 10 implements the partitioned booking limits control policy. It is a straightforward policy where each product is a booking bucket in its own right. Therefore there are n booking buckets. Booking requests for a booking bucket can only be allocated to the requested bucket and to no other. If a bucket is closed or sold out, then the request cannot be accepted into any other bucket.

Algorithm 10: Pseudo-code for simulating booking acceptance for partitioned booking limit control policy

```
Ensure: $C_k$ = Total Capacity $\forall\, k$, $q_j = 0 \;\forall\, j$, $b_j = x_j \;\forall\, j$
 1: if $P_j(t) > 0$ then
 2:     if $b_j >= P_j(t)$ accept booking request then
 3:         $b_j = b_j - 1$
 4:         $C_k = C_k - 1 \quad \forall\, k \in I$
 5:         $q_j = q_j + 1$
 6:     else
 7:         Decline request
 8:     end if
 9:     if New request arrives then
10:         Go to step 1
11:     end if
12: end if
```

The algorithm initialises by setting the state of the remaining capacity per leg, C_k, equal to the total capacity of the plane servicing each leg k. The number of accepted booking requests for product j, q_j, is set equal to zero for all products j. The booking limit of each bucket b_j is equal to the seat allocation x_j calculated by any of the approximation models described in section 2.4.2.

When a booking request, $P_j(t)$, for a product j arrives at time t, it must be considered for acceptance. Since the booking request can only be allocated to the requested bucket, we only have to check if there is space available in that bucket. We do so in Step 2. If seats are available in the booking bucket, the request is accepted, and we proceed to Step 3, where we adjust the quantities of booking limits, remaining capacity, and accepted requests. On the other hand, if there is not enough space in the bucket, the request is denied immediately. The process is then repeated from Step 1 when a new request arrives.

4.2.2 Nested Booking Limits

As already discussed in section 2.2.2, an intuitive extension to the partitioned booking limits policy is the nested booking limits policy. It allows higher-ranked products access to seats reserved for lower-ranked products. Below we present Algorithm 11 for nested booking limits as proposed by de Boer et al. (2002).

In Step 0, the initialisation of the algorithm takes place. The remaining capacity is set to the total capacity for each resource k. The vector of accepted booking requests is set equal to $\mathbf{0}$. In Step 1, a booking request $P_j(t)$ for a product arrives and is considered for acceptance. In Step 2, the quantity b_j is defined as the number of seats we wish to protect for each product j against lower-ranked products for the remainder of the booking horizon. Initially, $b_j = x_j \;\forall j$ where x_j is the seat allocation produced from any

Algorithm 11: Pseudo-code for the nested booking limit control

```
Ensure: $C_k$ = Total Capacity $\forall\, k$, $q_j = 0 \,\forall\, j$
 1: while $P_j(t) > 0$ do
 2:   $b_j = \max\{x_j - q_j, 0\}$
 3:   $b_k = \sum_{s \in S_k;(x_j)>(P_j)} b_j^+$ for each $k \in I$
 4:   $c_{\min} = \min\{C_k - b_k | k \in I\}$
 5:   if $c_{\min} > 0$ accept booking request then
 6:     $C_k = C_k - 1 \,\forall\, k \in \boldsymbol{I}$
 7:     $q_j = q_j + 1$
 8:   else
 9:     Decline request
10:   end if
11: end while
```

approximation or robust model formulation described previously. In Step 3, the variable b_k is introduced, which is defined as the number of seats that need to be protected at each leg k of the itinerary I for all products that are ranked higher than the product requested. Then the minimum capacity is calculated as the minimum number of seats available among all legs in the itinerary I of the requested product. If there is still capacity available, then the booking request is accepted, and the quantities $\boldsymbol{C}$ and $\boldsymbol{q}$ are adjusted accordingly in Steps 6 and 7. Otherwise, the booking request is rejected, and we return to Step 1 when a new request arrives.

4.2.3 Bid Prices Control

The final algorithm implements the control of static bid prices. The "static" in the description indicates that the displacement cost or shadow price for each leg π_k is calculated once at the beginning of the booking horizon and not recalculated thereafter. We assume that revenue depends on the remaining capacity in a linear way.

Algorithm 12: Pseudo-code for simulating booking acceptance for bid-prices booking control policy

```
Ensure: $C_k$ = Total Capacity $\forall\, k$, $q_j = 0 \,\forall\, j$, $\boldsymbol{\pi} = [\pi_k \mid k = 1, ..., m]$
 1: while $P_j(t) > 0$ do
 2:   $\boldsymbol{T}_j = \sum_{k \in I_j} \pi_k$
 3:   if $f_j \geq \boldsymbol{T}_j$ and $C_k > 1 \,\forall\, k \in I_j$ then
 4:     Accept booking request and update values:
 5:     $C_k = C_k - 1 \,\forall\, k \in I_j$
 6:     $q_j = q_j + 1$
 7:   else
 8:     Decline request
 9:   end if
10: end while
```

The algorithm initialises by setting the remaining capacity equal to the total capacity of each resource of the network, and the number of accepted requests is equal to **0**. The variables π_k are set equal to the shadow prices of each leg as calculated by any of the models described in section 2.4.2. When a booking request arrives, it is considered for acceptance. The threshold price associated with the product request $\boldsymbol{T}_j$ is equal to the summation of the shadow prices of the legs the itinerary of the product in request uses. In step 3, we check whether the requested product's fare value is greater than or equal to its threshold price and if there is enough capacity left for the requested product. If these two conditions are met, then the request is accepted, and the quantities of remaining capacity and accepted requests are adjusted accordingly. If the fare price is lower than the threshold price or there is not enough capacity, the request is denied. When a new request arrives, we repeat the process from Step 1.

4.3 Airline network example

To aid comparison, we use the example network, presented in Figure 4.3. It is the network first introduced in Talluri and Van Ryzin (2006) and was also analysed by de Boer et al. (2002) and Perakis and Roels (2010) whose work we expand. The methodology presented in Chapter 3 was implemented in MATLAB, and the optimisation carried out using the GUROBI solver.

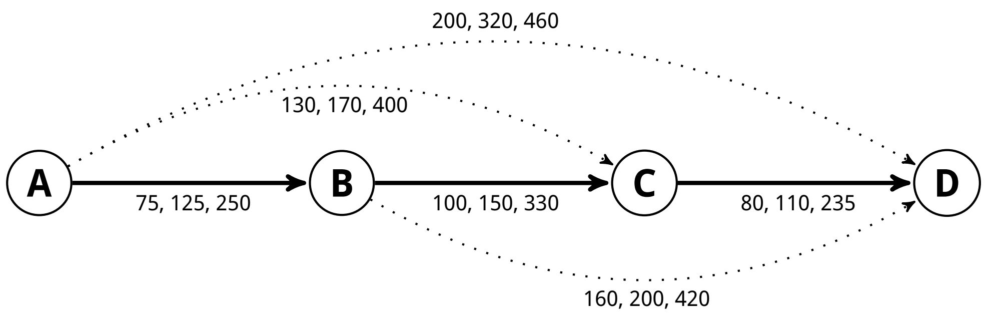


FIGURE 4.3: Small Serial Network

In this example, four cities are arranged in series, represented by nodes in the diagram. Flights are assumed to be in one direction only. This yields 6 OD pairs represented by straight and dotted arcs. Straight lines represent OD pairs A-B, B-C and C-D, which are direct itineraries that use single legs in the network. On the other hand, dotted lines represent OD pairs A-C, A-D and B-D, which are itineraries that use multiple legs of the network. For example, a customer requesting an itinerary A-C would use leg A-B and then leg B-C of the network.

There are three Fare Classes for each OD pair. Fare Class 1 has the highest value, and Fare Class 3 has the lowest. This yields 18 unique ODF combinations. Table 4.1 summarises this information and reports the fare value for each ODF combination. The numbers in brackets are used as an index for the products in what follows.

OD number	Origin-Destination	Fare class 3	Fare class 2	Fare class 1
1	A-B	£75 (1)	£125 (7)	£250 (13)
2	A-C	£130 (2)	£170 (8)	£400 (14)
3	A-D	£200 (3)	£320 (9)	£460 (15)
4	B-C	£100 (4)	£150 (10)	£330 (16)
5	B-D	£160 (5)	£200 (11)	£420 (17)
6	C-D	£80 (6)	£110 (12)	£235 (18)

TABLE 4.1: Origin-Destination-Fare combinations for linear network example

4.4 Partitioned Booking Limits

The easiest booking control policy, π, to implement is partitioned booking limits. Under this policy the number of buckets, n, is equal to the number of ODF combinations and each ODF is a bucket on its own.

4.4.1 DLP: Low, Mean and High Demand

The problem is solved initially using the DLP model, where, by definition, demand is represented deterministically. We solve the model for three demand scenarios: low, medium and high demand. We do so to highlight how extreme demand scenarios affect the allocation of resources. Even though the low and demand scenarios are unrealistic we do gain valuable insides from examining them. The low demand scenario represents an extreme worst-case instance of the problem and has a direct connection to robust formulations. The high demand scenario provides an upper bound to the expected revenue albeit a weak one as the probability of demand for all products to be at its highest level simultaneously is insignificant.

To acquire deterministic values for these scenarios, we assume that demand follows a Normal distribution as parametrised in Table A.1. We then generate 1000 random numbers from the normal distribution for each booking bucket. For the mean demand scenario we take the expected values of demand, $\mathbb{E}[D]$, for each booking bucket. For the low demand scenario, we take the lower bound of demand for each booking bucket, $l_s \; \forall \, s$, to be the 25th percentile of the random generated numbers. This is an unrealistically pessimistic scenario because it is unlikely for demand to be low for all booking buckets simultaneously. Lastly, for the high demand scenario, demand for each bucket

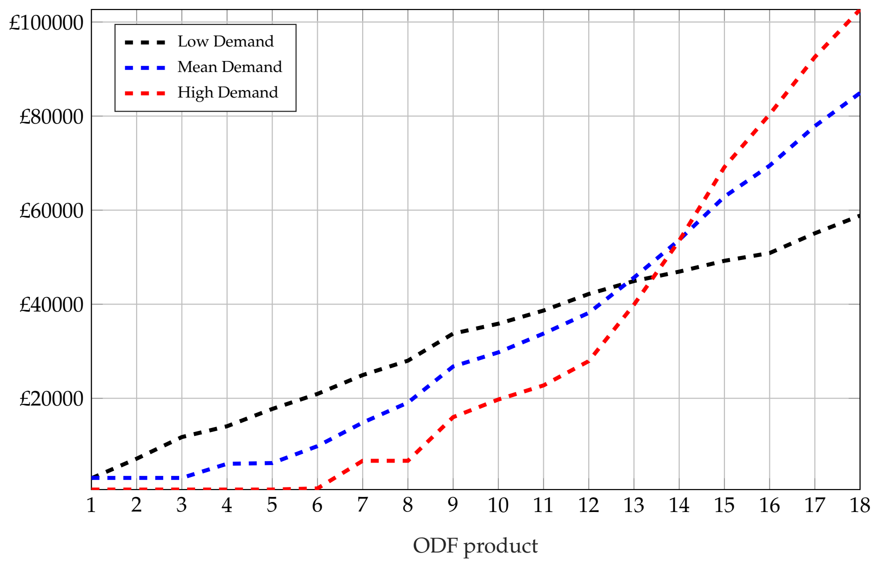


FIGURE 4.4: Expected Revenues per ODF product with Partitioned booking limits as produced by the DLP method

is set to an upper bound, $u_s \, \forall \, s$, that is equal to the 75th percentile of the randomly generated numbers. The DLP is then solved for each scenario and the partitioned booking limits calculated are presented in Table 4.2. It is worth noting that every time a new set of normal random numbers is generated, the lower and upper bounds of demand take new values leading to slightly different booking limits for these scenarios.

It is clear from Table 4.2 that for the high demand scenario, more seats are reserved for the high-priced Fare Class 1 products while almost all of the low-priced Fare Class 3 products are closed. This is an intuitively sound result since if there is a high demand from Fare Class 1 customers to fill the capacity, then the vendor will always prefer to sell Fare Class 1 customers instead of lower-valued Fare Class 2 or Fare Class 3 customers since it is more profitable.

On the other hand, when demand for all products is low, more seats are allocated to the lower-paying customers who have a higher total expected number of booking requests. High Fare Classes are still open but are allocated a significantly lower number of seats.

OD pair	Low Demand			Mean Demand			High Demand		
	Class3	Class2	Class1	Class3	Class2	Class1	Class3	Class2	Class1
1	36	29	11	41	40	30	3	50	48
2	29	12	5	0	25	20	0	0	34
3	20	16	5	0	24	20	0	31	34
4	20	12	5	30	20	20	0	27	34
5	20	13	10	1	20	20	0	11	29
6	36	29	16	45	40	30	2	50	43

TABLE 4.2: Partitioned booking limits (DLP) for Low, Mean and High Demand

Scenario	**Expected Revenue**	**Simulated Revenue**				**90% Confidence Intervals**		**5th Percentile**	**10th Percentile**	**15th Percentile**
		Min	**Average**	**Max**	**StD**					
Low	£53,585	£45,020	£51,367	£53,585	£1,121	£51,341	£51,393	£49,280	£49,845	£50,165
Mean	£84,915	£48,730	£69,767	£83,390	£5,365	£69,642	£69,892	£60,510	£62,615	£64,060
High	£103,050	£35,290	£65,146	£91,170	£8,969	£64,937	£65,355	£50,123	£53,460	£55,798

TABLE 4.3: Comparison of Expected and Simulated Revenues for Low, Medium and High Demand scenarios

The expected revenue is calculated for the partitioned booking limit control policy by multiplying the fare value for each ODF by the number of seats allocated to each booking bucket. Let $\boldsymbol{x}_s^{\pi}$ be the column vector indicating the number of seats allocated to each booking bucket, s for policy $\pi \in \Pi$. Let $\boldsymbol{f} = [f_1, f_2, ..., f_n]$ be the column vector of fare values for each ODF, $j = [1, ..., n]$. When π is the partitioned booking limits control policy, then the number of booking buckets is equal to the number of ODF products and thus, $\mathbb{E}[Revenue] = \boldsymbol{f}'\boldsymbol{x}_s^{\pi}$.

Figure 4.4 displays a plot describing the aggregated expected revenues generated by each ODF combination for the low, mean and high realisations of demand. It is important to note that even though it has been proven that the objective value for the mean demand scenario is indeed an upper bound for the expected revenue of the network (Cooper, 2002), the best case scenario is adding information to the solution space of the problem. This proves particularly useful when solving problems 3.2.1 using the genetic algorithm (see Section 3.3).

To test the seat allocation and the booking control policy, we run 5000 simulations of the booking period. To ensure a partitioned control policy, we use Algorithm 10 to accept or deny requests. The motivation for this experiment is to compare the expected and simulated revenues. The numerical results suggest that the limited use of partitioned booking limits in real-life applications is justified. We report the expected revenue as calculated using seat allocations to buckets generated by the DLP model. We also report summary statistics for the simulated revenue including 90% confidence intervals and to assess the lower end of the simulated revenue distribution we report the $5^{th}, 10^{th}$, and 15^{th} percentile. The results are displayed in Table 4.3.

OD Pair	Fare Class 3			Fare Class 2			Fare Class 1		
	Booking Limits	Sold Seats	Total Requests	Booking Limits	Sold Seats	Total Requests	Booking Limits	Sold Seats	Total Requests
1	3	3	48	50	34	34	48	10	10
2	0	0	46	0	0	20	34	12	12
3	0	0	36	31	17	17	34	34	35
4	0	0	23	27	10	10	34	18	18
5	0	0	40	11	11	27	29	13	13
6	2	2	46	50	34	34	43	30	30

TABLE 4.4: Booking Limits, Sold Seats and Total Requests for the High demand scenario

The immediate conclusion drawn is that the DLP significantly overestimates the generated revenue. It is also evident that partitioned booking limits are a suboptimal control policy. We illustrate this by a closer examination of the high demand scenario. Table 4.4 is an example of the results generated from a single simulation run. As noted before most of the Fare Class 3 buckets are closed even though there was a considerable number of requests for them. Fare Class 2 buckets are either sold out or demand is less than the allocated capacity to each bucket. For example ODF 7 has a 50 seat allocation but there was only 34 booking requests during the booking period and thus only 34 seats are sold. Here lies the limitation of partitioned booking control policy. The 16 seats left cannot be sold to any other bucket. This means that there are empty seats in the plane even though there were customers willing to fly. All booking buckets highlighted suffer from the same limitation. This also explains the significant difference between the expected revenue which assumes a full capacity of 200 passengers for every leg of the network and the simulated capacity which is only 110, 115 and 141 passengers for legs A-B, B-C and C-D respectively.

4.4.2 Comparison of Previous Approximation Models

OD number	DLP			EMR			RLP		
	Class3	Class2	Class1	Class3	Class2	Class1	Class3	Class2	Class1
1	41	40	30	42	40	40	39	38	30
2	0	25	20	0	18	22	6	17	19
3	0	24	20	0	21	17	4	23	20
4	30	20	20	23	19	27	21	19	21
5	1	20	20	15	16	22	8	18	19
6	45	40	30	38	36	35	34	39	29
Expected Revenue:	£84,915			£71,767			£82,370		

TABLE 4.5: Comparison of Partitioned booking limits from MP Models

In Table 4.5 the partitioned booking limits and expected revenue produced by the DLP, EMR and the RLP models introduced in Section 2.4.2 are compared.

Our results are identical to what de Boer et al. (2002) report for the DLP and EMR models. It is evident that the DLP model yields considerably more expected revenue, provided that the capacity of the planes is filled. This is a somewhat unexpected result in the intuitive sense since incorporating the stochasticity of demand should have resulted in higher revenue. To this end, as an extension to the de Boer et al. (2002) study, we implemented the RLP model that incorporates demand stochasticity. It performs considerably better than the EMR and only slightly worse than the DLP. In general, the RLP booking limits are almost identical to the DLP booking limits for Classes 1 and 2. The RLP however is distinctly different for Class 3 since it allocates seats to OD 2 and 3 therefore allowing bookings for these products in contrast to the other two policies. It is important to note that the RLP is seldom used to produce partitioned booking limits, but rather for calculating bid prices.

Scenario	Expected Revenue	Simulated Revenue				90% Confidence Intervals		5th Percentile	10th Percentile	15th Percentile
		Min	Average	Max	StD					
DLP	£84,915	£48,730	£69,767	£83,390	£5,365	£69,642	£69,892	£60,510	£62,615	£64,060
EMR	£71,767	£44,435	£70,757	£86,475	£6,136	£70,615	£70,900	£60,458	£62,753	£64,315
RLP	£82,370	£45,165	£68,588	£80,955	£5,272	£68,466	£68,711	£59,463	£61,773	£63,088

TABLE 4.6: Comparison of Partitioned booking limits of traditional approximation models

Another simulation of the booking period is run to test the different allocations produced by our traditional approximation models. The simulation has 5000 iterations and Table 4.6 displays the summary statistics of the results.

The simulation results suggest that the EMR seat allocation to the booking buckets,x_s, performs the best under the partitioned booking limit control policy. It generates the highest average simulated revenue, and even though the variation of this model is slightly higher than the DLP or the RLP, the spread of the 90% confidence interval is small. Furthermore, it is the only model that generates a Simulated Revenue close to its Expected Revenue. This is primarily because the model's objective function takes into account the stochasticity of demand, which is not the case for the DLP or the RLP models.

4.4.3 Robust Controls

In this section, we present results for the robust models described in Chapter 3 (section 3.2.1) using partitioned booking limits and compare them with results from Perakis and Roels (2010) and de Boer et al. (2002). As the genetic algorithm has a random element, we record 10 runs of the algorithm and find the best solution.

Table 4.7 presents the partitioned booking limits of 10 runs of the maximin Revenue model. The seat allocation to each bucket (s), the expected revenue $(\boldsymbol{f'x})$, maximum regret (ρ) and maximin Revenue (ϕ^*) for each instance are presented. We choose the booking limits for the maximin Revenue model that return the highest maximin Revenue objective value.

To calculate the minimax Regret (ρ^*) partitioned booking limits we also run 10 instances of the model. This criterion tries to minimise the maximum regret and is on average less conservative than the maximin Revenue criterion. This yields significantly higher expected revenue because less seats are protected for the lower Fare Classes 2 and 3 while more seats are protected for the higher Fare Class 1.

The ellipsoidal uncertainty sets provide different booking limits as seen in Table 4.9. Since demand for the products is characterised differently we see that, less seats are protected for Fare Class 1 compared to the allocation of the polyhedral minimax regret.

Fare Class	s	Runs 1	2	3	4	5	6	7	8	9	10
3	1	44	**40**	38	42	36	43	38	34	43	43
	2	11	**17**	30	23	19	28	20	29	30	30
	3	22	**25**	15	25	23	17	22	19	17	19
	4	24	**25**	25	14	17	21	23	20	26	19
	5	23	**21**	21	29	22	17	17	25	12	22
	6	39	**42**	42	35	42	45	47	39	43	43
2	7	39	**34**	36	34	34	32	35	33	34	34
	8	20	**19**	15	17	19	20	19	18	17	16
	9	21	**21**	21	19	20	21	20	20	21	17
	10	19	**17**	15	16	19	16	18	15	18	17
	11	17	**12**	16	13	16	17	16	13	16	16
	12	33	**35**	39	33	30	31	34	33	42	37
1	13	23	**24**	18	17	22	19	24	28	18	20
	14	10	**10**	10	10	11	10	11	9	10	11
	15	10	**10**	10	12	12	10	10	10	10	10
	16	9	**9**	9	9	9	9	9	9	9	10
	17	14	**13**	13	13	13	14	13	13	14	13
	18	19	**20**	21	20	21	26	20	21	25	22
f/x (£)		70,870	70,160	68,945	68,730	70,365	70,325	70,405	69,870	70,690	69,945
ρ (£)		26,025	26,425	27,860	28,495	26,320	26,400	26,420	26,715	26,545	26,640
ϕ^* (£)		67,280	**67,830**	67,340	67,170	67,110	67,395	67,590	66,885	67,245	67,515

TABLE 4.7: 10 runs of the Maximin Revenue partitioned booking control policy

Fare Class	s	Runs 1	2	3	4	5	6	7	8	9	10
3	1	45	21	**42**	44	23	41	44	40	33	45
	2	1	26	**4**	5	12	0	0	4	4	0
	3	2	4	**11**	3	15	0	0	8	16	0
	4	27	9	**27**	26	9	33	34	20	6	29
	5	2	13	**9**	14	3	0	0	7	12	1
	6	41	32	**39**	41	39	42	50	42	32	47
2	7	34	43	**40**	35	37	39	40	34	38	40
	8	16	5	**5**	12	11	18	20	18	18	17
	9	22	22	**21**	21	23	22	22	19	15	21
	10	20	17	**17**	19	21	19	21	21	23	21
	11	17	16	**16**	10	15	21	14	13	18	35
	12	42	38	**31**	34	33	41	40	35	33	35
1	13	37	37	**35**	34	35	36	32	35	35	35
	14	23	22	**23**	22	23	22	23	23	22	23
	15	20	20	**19**	20	19	19	18	19	19	19
	16	27	25	**26**	26	27	25	27	26	24	28
	17	22	21	**21**	21	21	21	21	22	22	21
	18	32	34	**33**	36	32	33	35	34	33	35
f/x (£)		88,555	86,800	86,705	86,755	86,360	87,635	87,925	87,090	85770	88,580
ϕ (£)		58,270	57,840	59,145	59,030	58,040	57,690	57,60	59,755	58,640	57,910
ρ^* (£)		12,545	13,160	**12,275**	12,770	13,405	12,910	13,660	12,640	13,405	12,630

TABLE 4.8: Minimax Regret Partitioned booking control policy for 10 simulations

As was the case with the partitioned booking limits derived by traditional models and presented in Table 4.5, we again simulate the booking period 5000 times and apply Algorithm 10 to assess the effectiveness of the booking limits derived from the robust models. Table 4.10 presents the summary statistics of the results.

The table shows that the minimax Regret performs significantly better than the maximin Revenue with difference in average revenue of 12%. This an expected result since

Fare Class	s	Runs									
		1	2	3	4	5	6	7	8	9	10
	1	37	35	34	40	37	35	**38**	37	40	42
	2	0	0	0	0	0	0	**0**	0	0	0
3	3	0	0	0	0	0	0	**0**	0	0	0
	4	20	23	25	29	29	22	**24**	28	30	27
	5	8	3	0	0	0	6	**0**	0	0	6
	6	38	43	45	45	46	42	**42**	46	47	40
	7	39	40	40	40	40	40	**38**	37	41	38
	8	25	24	24	22	22	24	**23**	24	25	21
2	9	23	23	24	22	23	23	**23**	24	21	22
	10	19	20	20	20	20	20	**17**	19	20	19
	11	18	19	19	20	18	17	**19**	20	19	18
	12	40	39	39	40	40	39	**38**	38	40	40
	13	34	35	35	34	35	35	**35**	35	34	35
	14	22	23	23	22	23	23	**24**	21	19	23
1	15	20	20	20	20	20	20	**19**	22	19	19
	16	24	24	24	24	24	24	**25**	23	27	24
	17	21	21	21	21	21	21	**26**	18	19	21
	18	32	32	32	32	32	32	**33**	32	32	32
$f'x$ (£)		87,495	87,410	87,355	87,870	87,495	87,090	87,420	88,055	87,760	87,735
ϕ (£)		65,745	65,700	65,640	65,940	65,765	65,600	64,525	65,845	66,160	65,570
ρ^* (£)		8,895	8,940	8,995	8,980	8,895	9,010	8,865	9,195	9,045	9,070

TABLE 4.9: Minimax Regret partitioned booking limits instances using ellipsoidal uncertainty sets

the maximin Revenue model is concerned with the worst-case scenarios and tries to maximise the revenue for such cases. The conservativeness of the maximin Revenue model is reflected in the significantly smaller standard deviation it yields. As was the case with the traditional approximation models the expected revenue is an overestimation of the simulated revenue. Since we are still implementing the partitioned booking control this difference is explained as the lost revenue from empty seats in buckets than cannot be sold to any other customers.

Policy	Objective Value	Expected Revenue	Simulated Revenue				90% Confidence Intervals		5th Percentile	10th Percentile	15th Percentile
			Min	Average	Max	StD					
$\phi^*_{\mathcal{U}_P}$	£64,330	£75,620	£49,285	£66,926	£74,740	£3,638	(£66,841	£67,011)	£60,365	£62,020	£63,070
$\rho^*_{\mathcal{U}_P}$	£11,595	£87,315	£46,810	£70,288	£85,595	£5,882	(£70,151	£70,425)	£60,090	£62,493	£64,143
$\phi^*_{\mathcal{U}_E}$	£70,225	£79,135	£50,550	£68,341	£78,310	£4,077	(£68,246	£68,436)	£61,125	£62,890	£64,075
$\rho^*_{\mathcal{U}_E}$	£8,865	£89,175	£45,945	£70,204	£87,845	£6,256	(£70,059	£70,350)	£59,445	£61,858	£63,663

TABLE 4.10: Comparison of Expected and Simulated Revenues for Robust controls

The final result of this section is table 4.12. It is a collective representation of the summary statistics generated after simulating the booking period for all the seat allocation models we have used under the partitioned booking control policy.

From table 4.12 it is easy to see that partitioned booking limits are a suboptimal booking control policy. All models fail to generate the expected revenue by a considerable margin. At the same time, the nature of the booking policy leads to average booking acceptance decisions that half-fill the resources of the networks. It is worth noting that the EMR is the clear winner in terms of simulated revenue since it considers the probabilistic nature of demand, and the limits generated with this model seem to perform

Fare Class	s	Runs									
		1	2	3	4	5	6	7	8	9	10
1	1	46	46	46	**47**	47	47	46	46	45	46
	2	7	9	9	**8**	11	6	7	9	9	9
	3	4	5	3	**3**	5	2	4	4	3	4
	4	27	25	26	**26**	30	27	27	26	28	26
	5	10	11	10	**12**	7	12	10	10	9	11
	6	43	42	44	**43**	49	43	43	44	46	44
2	7	40	39	39	**40**	38	40	40	39	40	39
	8	25	24	25	**25**	24	25	25	25	25	25
	9	24	23	24	**23**	23	24	24	23	23	23
	10	20	20	20	**20**	20	20	20	20	20	19
	11	20	20	20	**20**	19	20	20	20	20	20
	12	39	39	39	**39**	39	39	39	39	39	38
3	13	25	25	25	**25**	24	26	25	25	26	25
	14	15	15	15	**15**	15	16	15	15	15	15
	15	14	14	14	**14**	13	14	14	14	14	14
	16	16	16	16	**16**	15	16	16	16	16	16
	17	18	18	18	**18**	18	18	18	18	18	18
	18	28	28	28	**28**	27	28	28	28	28	28
$f'x$ (£)		79,230	78,955	79,145	79,135	78,110	79,745	79,230	79,025	79,325	78,925
ρ (£)		16,345	16,320	16,405	16,280	17,135	15,830	16,345	16,365	16,090	16,440
ϕ^* (£)		70,135	70,160	70,100	70,225	70,220	70,045	70,135	70,150	70,205	70,070

TABLE 4.11: Maximin revenue partitioned booking limits using ellipsoidal uncertainty set

Policy	**Expected Revenue**	**Simulation Runs**	**Simulated Revenue**				**90% Confidence Intervals**
			Min	**Average**	**Max**	**StD**	
DLP-Low	£53,585	5000	£45,520	£51,762	£53,585	£1,079	(£51,737, £51,787)
DLP-High	£103,050	5000	£36,930	£64,793	£93,830	£8,993	(£64,584, £65,002)
DLP	£84,915	5000	£47,750	£69,647	£84,150	£5,447	(£69,520, £69,774)
RLP	£82,370	5000	£46,540	£68,779	£81,180	£5,352	(£68,654, £68,903)
EMR	£71,767	5000	£49,960	£70,972	£87,715	£6,125	(£70,830, £71,114)
MiniMax Regret	£86,575	5000	£49,075	£69,918	£85,985	£6,111	(£69,776, £70,060)
Maximin Revenue	£71,250	5000	£54,210	£65,582	£71,140)	£2,780	(£65,518, £65,647)

TABLE 4.12: Comparison of Expected and Simulated Revenues for all controls

the best. The minimax Regret also performs surprisingly well in terms of simulated revenue. Its results are on par with the DLP and RLP models, which showcases that it is not as conservative as the maximin Revenue.

4.5 Nested Booking Limits

When π takes the form of nested booking controls, the booking buckets are no longer equal to the number of ODF products, but as explained in section 2.2.2, products are grouped together according to their OD pair and ranked according to their fare value. Hence, in our network example, there are six OD pairs and six booking buckets.

4.5.1 Deterministic Linear Programming

Consider the seat allocation to each product j under the partitioned booking control policy as calculated by the DLP for the different demand cases (Table 4.3). Then for the nested booking control policy the seat allocation to the booking buckets is calculated by firstly collecting all products that share the same OD pair into the same bucket, thus creating six of them and then ordering the products in each bucket according to their fare value. That is Fare Class1 $\geq$ Fare Class 2 $\geq$ Fare Class 3. Then the nested allocation is calculated as the summation of the partitioned allocations for each product as shown in Table 4.13. Using the same procedure slightly different nested booking limits can be derived by calculating the initial partitioned booking limits using the EMR and RLP models described in Section 2.4.2.

OD pair	Low Demand			Mean Demand			High Demand		
	Class3	Class2	Class1	Class3	Class2	Class1	Class3	Class2	Class1
1	43	77	95	41	81	111	27	72	113
2	22	43	52	0	25	45	0	0	30
3	25	44	53	0	24	44	0	27	57
4	21	37	46	30	50	70	11	34	64
5	19	35	49	1	21	40	0	23	49
6	43	77	98	45	85	115	10	55	94

TABLE 4.13: Nested booking limits (DLP) for Low, Mean and High Demand

4.5.2 Maximin Revenue

o calculate the maximin Revenue nested booking limits, we do not follow the same procedure as described above. In Chapter 3 it is seen that for the robust models minimax Regret and maximin Revenue, we specify the booking control policy constraints in the inner problem (Eq. 3.2a) the booking control policy constraints. Then in the outer problem, the genetic algorithm maximises the minimum revenue given the constraints of the booking control policy we are implementing. For the nested booking limits, the constraint matrix is given in Appendix C.2.

Table 4.14 reports the maximin Revenue nested booking limits. For each instance, the maximum regret (ρ), the maximin Revenue (ϕ^*) and the time to compute the booking limits are shown. For each set of instances (1-3, 4-6, 7-9, 10) of the problem, a different initial population matrix was passed into the GA algorithm by recalculating the nested booking limits for the three demand scenarios using the DLP model. Problem 3.2a was solved to find the minimum revenue, and then the genetic algorithm was employed to change the allocation to the buckets so that the minimum revenue is maximised subject to booking control and capacity constraints. This procedure is repeated until the stopping criteria of the algorithm are met. It is clear from the table that the different initial solutions passed to the genetic algorithm yield slightly different results, but the algorithm converges to the same answer for each initial solution. This is equivalent to the low demand scenario.

Fare	OD	Instances									
Class	Pair	1	2	3	4	5	6	7	8	9	10
3	1	43	43	43	44	44	44	43	43	43	44
	2	18	18	18	20	20	20	21	21	21	22
	3	25	25	25	25	25	25	24	24	24	25
	4	22	22	22	20	20	20	40	40	40	25
	5	17	17	17	21	21	21	40	40	40	18
	6	44	44	44	42	42	42	43	43	43	44
2	1	77	77	77	79	79	79	77	77	77	78
	2	38	38	38	41	41	41	42	42	42	42
	3	45	45	45	44	44	44	44	44	44	44
	4	33	33	33	36	36	36	36	36	36	40
	5	33	33	33	37	37	37	36	36	36	34
	6	78	78	78	76	76	76	78	78	78	78
1	1	95	95	95	96	96	96	95	95	95	95
	2	50	50	50	51	51	51	52	52	52	51
	3	55	55	55	53	53	53	53	53	53	54
	4	48	48	48	46	46	46	46	46	46	48
	5	47	47	47	50	50	50	49	49	49	47
	6	98	98	98	97	97	97	98	98	98	99
ρ (£)		27,225	27,225	27,225	27,825	27,825	27,825	27,525	27,525	27,525	28,905
ϕ^* (£)		70,175	70,175	70,175	69,070	69,070	69,070	69,165	69,165	69,165	68,465
Time (sec)		110	97	121	107	137	107	115	93	93	114

TABLE 4.14: Maximin Revenue Nested booking control policy for 10 simulations

4.5.3 Minimax Regret

For the minimax Regret nested booking control policy the same procedure as described in the previous section is followed. The same control policy constraint matrix is used as for the maximin Revenue to solve the inner problem (Eq. 3.2a). Then the outer problem is solved by the GA algorithm with initial solution being the DLP nested booking limits. Table 4.15 presents the results of the algorithm for 10 different instances. The maximum regret is minimised by Run 4 nested booking limits thus these are the limits we choose.

4.5.4 Comparison of Nested control policies

Table 4.16 is a comparison of the Nested booking limits generated by the DLP, maximin Revenue, (ϕ^*), and minimax Regret (ρ^*) models. It is clear from the table that the minimax Regret model produces nested booking limits close to the DLP while the maximin Revenue produces far more conservative limits for all products. Furthermore, the DLP and minimax Regret models severely limit the amount of seats allocated to Fare Class 3 customers with three such products being almost closed. The extra seats are rather

Fare Class	OD Pair	Instances 1	2	3	4	5	6	7	8	9	10
3	1	39	40	40	**38**	38	38	37	42	37	41
	2	0	0	0	**0**	1	0	0	0	0	0
	3	0	0	0	**0**	0	0	1	0	0	0
	4	27	24	30	**27**	34	27	27	28	27	33
	5	1	4	1	**4**	0	2	3	0	1	2
	6	46	44	47	**47**	49	50	45	45	45	57
2	1	80	77	80	**78**	78	78	78	80	84	79
	2	22	24	22	**21**	22	24	20	20	22	20
	3	23	22	22	**22**	22	22	23	22	22	22
	4	49	47	50	**46**	49	51	48	52	47	48
	5	18	21	18	**22**	18	19	20	17	21	19
	6	83	80	87	**82**	87	88	84	84	83	85
1	1	114	113	114	**114**	114	114	114	115	114	115
	2	44	45	44	**44**	44	45	43	44	45	44
	3	42	42	42	**42**	42	41	43	41	41	41
	4	74	71	74	**71**	75	74	73	76	72	74
	5	40	42	40	**43**	39	40	41	39	42	41
	6	118	116	118	**115**	119	119	116	120	117	118
ρ^* (£)		11,730	11,720	11,685	**11,305**	11,770	11,985	11,525	12,095	11,910	11,855
ϕ (£)		58,290	59,030	58,365	58,490	57,770	57,960	58,245	58,045	57,380	58,290
Time (sec)		282	342	88	1580	837	145	940	103	182	191

TABLE 4.15: Minimax Regret Nested booking control policy for 10 simulations

OD pair	Nested - DLP			Nested - ϕ^*			Nested - ρ^*		
	Class3	Class2	Class1	Class3	Class2	Class1	Class3	Class2	Class1
1	41	81	111	43	78	95	39	79	114
2	0	25	45	20	41	51	0	22	44
3	0	24	44	25	44	54	0	22	42
4	30	50	70	27	36	47	28	49	73
5	1	21	41	25	35	49	2	19	41
6	45	85	115	43	77	98	48	84	118

TABLE 4.16: Nested booking limits from the DLP, Minimax Regret and Maximin Revenue controls

allocated to Fare Class 1 products. On the other hand maximin Revenue behaves in the converse way by allocating a significant number of seats to lower fare class customers and having less seats strictly available to Fare Class 1 customers.

To test the limits produced by the different models we run 5000 simulations of the booking period and applied Algorithm 11 to implement the nested booking acceptance policy each time using the limits calculated by a different model. Table 4.17 displays the summary statistics for all nesting booking control policies.

The surprising result here is that the DLP is the model that yields the best limits in terms of average simulated revenue. This is in agreement with de Boer et al. (2002). We further find that the minimax Regret performs on par with the DLP model and outperforms all others. The EMR and RLP models also perform well with average simulated revenues close to those of the DLP and minimax Regret. Finally, the maximin Revenue

Policy	Objective Value	Expected Revenue	Simulated Revenue				90% Confidence Intervals		5th Percentile	10th Percentile	15th Percentile
			Min	Average	Max	StD					
ρ^*_{UP}	£11,595	£87,315	£39,315	£67,015	£93,045	£7,774	£66,834	£67,196	£53,920	£56,955	£59,178
ϕ^*_{UP}	£64,330	£75,620	£44,145	£65,225	£89,720	£6,482	£65,074	£65,376	£55,215	£57,245	£58,523
ρ^*_{UE}	£8,865	£89,175	£39,015	£67,417	£98,180	£7,972	£67,232	£67,603	£54,030	£57,053	£59,185
ϕ^*_{UE}	£70,225	£79,135	£42,765	£66,850	£88,495	£6,589	£66,696	£67,003	£56,513	£58,573	£59,868
Low	£53,585	£53,585	£40,005	£53,728	£68,160	£3,858	£53,638	£53,818	£47,698	£49,023	£49,818
DLP	£84,915	£84,915	£39,750	£68,440	£99,115	£7,817	£68,258	£68,622	£55,600	£58,293	£60,290
High	£103,050	£103,050	£25,870	£59,257	£100,380	£10,973	£59,002	£59,513	£41,758	£45,415	£47,600
EMR	£71,767	£88,675	£38,335	£65,464	£92,035	£7,593	£65,287	£65,640	£53,175	£56,003	£57,783
RLP	£82,370	£82,370	£37,445	£66,394	£90,015	£7,144	£66,228	£66,561	£54,605	£57,360	£59,078

TABLE 4.17: Comparison of Expected and Simulated Revenues for all Nested controls

performs worse than most of the other models but this is expected due to its conservative nature. Nonetheless, it outperforms significantly the low-demand scenario DLP limits and provides a guarantee of minimum revenue.

4.5.5 Comparison with Perakis and Roels (2010) simulated results

Table 4.18 compares our results for the nested booking limits against those by Perakis and Roels (2010). Our results are highlighted for comparison. The booking limits generated by our policies outperform those of Perakis and Roels (2010) in terms of minimum revenue by a considerable margin. Greater minimum revenues are produced, a desirable outcome since we are trying to maximise this quantity, for every model. On the other hand, the Maximum Regret of the booking limits generated from our policies is only slightly higher for the DLP and minimax Regret models while it outperforms the booking limits generated for the maximin Revenue. Furthermore, the 90% CIs on simulated revenue show that all of our policies yield higher revenue on average. The improved performance in revenues is most likely created from the use of a GA algorithm to minimise/maximise problem 3.2a instead of a local gradient algorithm as used by Perakis and Roels (2010).

Policy	Maximum Regret		Minimum Revenue		90% Confidence Intervals	
DLP	£14,040	**£14,275**	£54,820	**£58,440**	$£73,248 \pm 343$	**$£75,666 \pm 164$**
ρ^*	£10,595	**£11,525**	£56,890	**£58,245**	$£73,734 \pm 338$	**$£74,699 \pm 171$**
ϕ^*	£28,150	**£27,225**	£52,085	**£70,175**	$£66,744 \pm 134$	**$£70,614 \pm 161$**

TABLE 4.18: Comparison with Perakis and Roels (2010) Simulated Revenues

4.6 Bid Prices

The last booking control policy that we implement is the fixed bid-price control. To decide whether to accept or reject an incoming booking request, the bid prices of the comprising legs for the requested itinerary are summed to calculate a price threshold. If the fare value for the requested itinerary is higher than the threshold then the request is accepted, otherwise it is rejected.

Consider our example. There are three legs in the network, therefore the bid price of each leg can be calculated using the DLP or the RLP model. Table 4.19 presents the bid and threshold prices. The DLP bid prices are calculated by the dual prices of the capacity constraints. The RLP bid prices are calculated as the average of the dual prices of the capacity constraints from 200 randomised DLPs.

Leg	DLP		RLP	
	Bid Price	Threshold Prices	Bid Price	Threshold Prices
A-B	£75	£75	£60	£60
B-C	£80	£80	£91	£91
C-D	£80	£80	£72	£72
A-C	-	£155	-	£151
B-D	-	£160	-	£163
A-D	-	£235	-	£223

TABLE 4.19: Bid Prices for the different itineraries

Consider that a booking request arrives for the itinerary A-C which comprises of travelling on legs A-B and B-C of the network. Let π be the DLP bid prices. Then, the OD pair A-C has a threshold price of £75 + £80 = £155. Therefore if the booking request is for Fare Classes 1 or 2 it is accepted since the fare values of these classes are greater than the threshold price, £400 > £170 > £155. On the other hand, if the booking request is for Fare Class 3, it is rejected since its value is lower than the threshold price, £130 < £155.

From Table 4.19 we can see that the RLP bid prices are slightly lower for legs A-B and C-D but higher for the bottleneck leg B-C. Furthermore, the lower bid price for A-B leg means that even Fare Class 3 customers have their booking requests accepted on the leg.

To compare the two bid prices controls we simulate the booking period and implement Algorithm 12 to accept or deny booking requests. Table 4.20 summarises the results after 5000 simulation runs.

Policy	Simulation	Simulation	Simulated Revenue				90% Confidence	
	Runs	Time (sec)	Min	Average	Max	StD	Intervals	
DLP	5000	13.91	£43490	£73183	£102575	£8400	£72988	£73379
RLP	5000	14.44	£51455	£76720	£94390	£6016	£76580	£76860

TABLE 4.20: Comparison of Expected and Simulated Revenues for all Nested controls

The RLP outperfoms the DLP bid prices. The effect of the taking stochasticity of demand into account when calculating the bid prices is evident from the considerable increase in average simulated revenue and the higher CI. Furthermore, the RLP bid prices have lower standard deviation for the simulated revenue suggesting that they are a more stable policy.

4.7 Conclusion

This chapter considers three booking control policies: partitioned booking limits, nested booking limits, and fixed bid prices. We also implement seven seat-allocation models: the traditional DLP, EMR, RLP and the robust minimax Regret and maximin Revenue models using polyhedral and ellipsoidal uncertainty sets. To evaluate their performance, we simulate the booking horizon using the non-homogeneous Poisson demand model and implement the three booking request acceptance algorithms described in Section 4.2.

Our numerical experiments are a direct extension of de Boer et al. (2002) and Perakis and Roels (2010) work. We expand de Boer et al. (2002) comparison of DLP and EMR models to include the RLP, minimax Regret and maximin Revenue models. Furthermore, in contrast to de Boer et al. (2002) we provide a study of the partitioned booking limits control (Section 4.4) in addition to the nested booking limits and bid prices which they also examine. For the robust models, we introduce the genetic algorithm for the outer problem in comparison to the local gradient algorithm Perakis and Roels (2010) use which improves the simulated revenue and results in higher minimum revenue for all booking policies. Although it produces narrowly higher maximum regret for the DLP and minimax Regret models, it successfully lowers the maximum regret for the maximin Revenue model. The comparative results are summarised in Table 4.18.

4.7.1 Future Work

Further work could expand our experiments to include more booking control policies like the Displacement Adjusted Virtual Nesting (DAVN). Non-static control policies that allow updating the booking limits during the booking horizon and evaluating all of our models under increased variance demand parametrisation and smaller fare spread are immediate extensions of this work. A further enhancement is the creation of better uncertainty sets that will adequately assess the risk posed by demand uncertainty but will be lenient enough to allow considerable revenue to be generated. As many uncertainty sets have already been proposed, mapping the robustness vs revenue trade-off would also be of interest.

Chapter 5

Cruise Line Application

In this chapter, we introduce a novel application of the methodology described in Chapter 3; a cruise network problem. Cruise revenue management is a complicated problem with a larger number of products with more characteristics and additional constraints compared to the airline formulation. Secondly, while there is an active but albeit small field of research on cruise revenue management, to the extent of our knowledge, the cruise network problem has not been studied thus far, and the robust methodology has not been employed in single or multiple legs setting.

The chapter is organised into five sections. First, in the introduction we state the contributions made in this chapter. Then an overview of the cruise industry is given, and the reasons that motivate the study of the cruise application as a compelling but *different* revenue management problem to the airline application are examined. We then state the mathematical problem to solve and the formulations we employ in section 5.2. In section 5.2.3 we describe the procedure we followed to derive simulation and optimisation parameters from data, crucial to producing the numerical results described in sections 5.3 and 5.4 respectively. Finally, in the last section of the chapter, we conclude and discuss future work on the cruise application.

5.1 Introduction

5.1.1 Contributions

The contributions we make are twofold. Firstly, we consider a network of cruise itineraries and not just a single voyage with a single point of embarkation and debarkation. This is different to previous formulations such as Sturm and Fischer (2018) who only consider a single leg multiple product setting or Li (2014) that consider as multiple resources the various combinations of product characteristics. Our approach also also incorporates

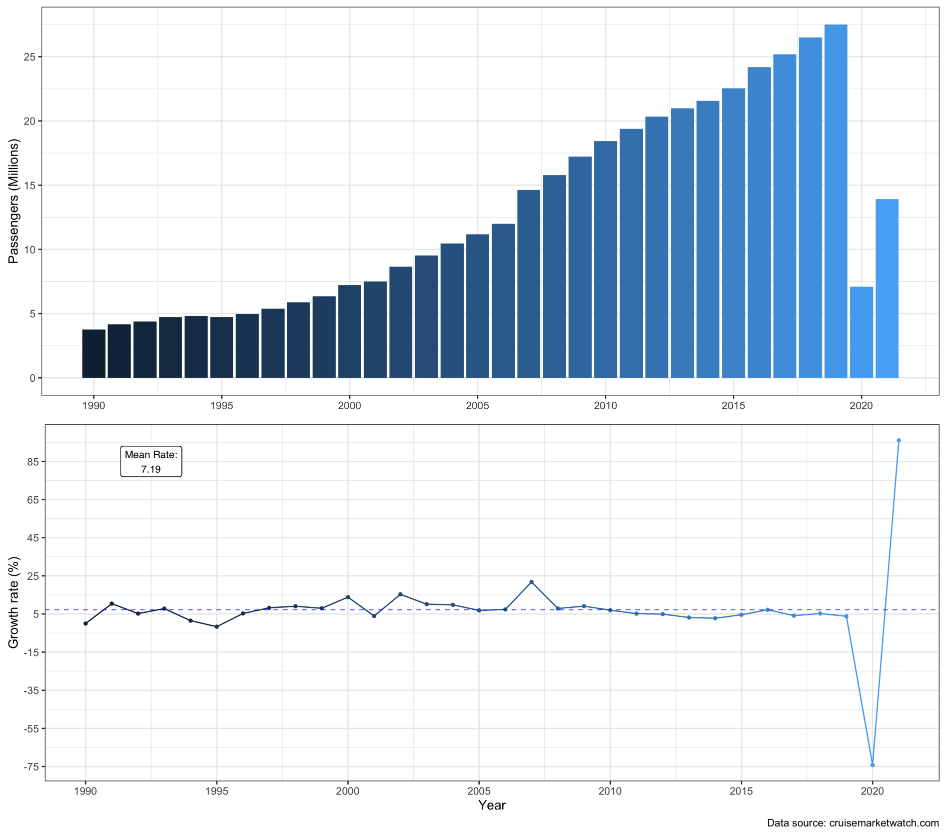


FIGURE 5.1: The rise of the cruise industry

the different product characteristics. This increases the size of the solution space and the complexity of the problem.

Secondly, we introduce robust optimisation to characterise uncertainty. This approach is particularly useful for legs in the network that have limited historical information available. This is a very common case in the cruise industry because year on year itineraries change, demand for certain voyages can be very limited and sporadic and the vast complexity of the product can mean that little data is available to infer market demand from sales. Furthermore, the COVID-19 pandemic has introduced great uncertainty in the sector and historical data is no longer such a good guide to future behaviour. Robust optimisation is designed to provide a solution that is robust to these uncertainties in demand.

5.1.2 Motivation

Up until the COVID pandemic in 2020, cruising was among the fastest-growing sectors in the leisure-travel industry with an average rate of passenger increase of over 6% year on year, which is more than double the industry's average (Ji and Mazzarella, 2007; Giese, 2020). The sector has been enjoying an increasing number of passengers and revenues for the past 20 years with its global economic impact estimated at 150

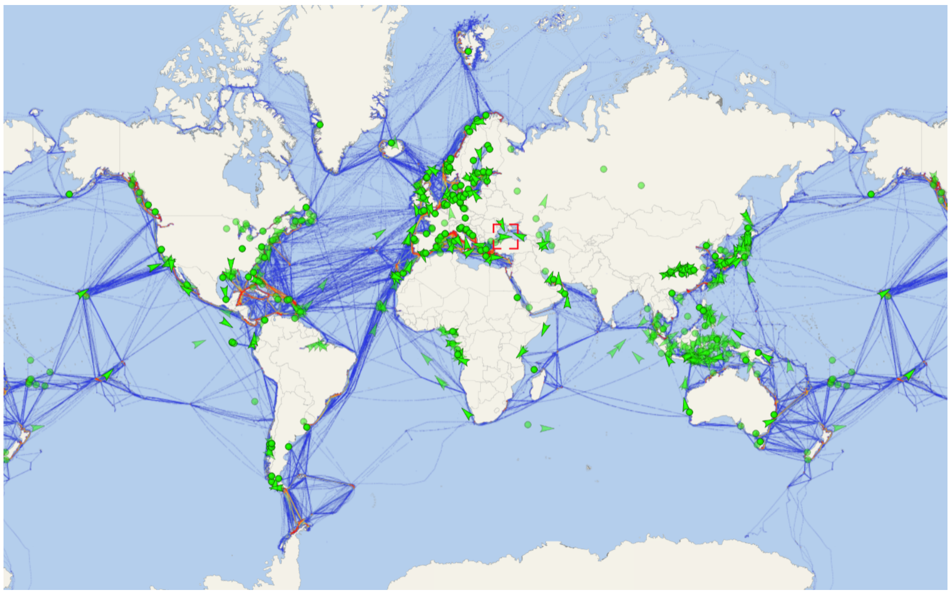


FIGURE 5.2: Cruise networks as tracked by the MarineVesselTravel website

billion US dollars in 2018 by the Cruise Line International Association (CLIA, 2019). The passenger growth, graphed in Figure 5.1, clearly shows the upward trend in the popularity of cruising. A further important characteristic of the sector is the very high occupancy rates of 95%, which distinguish it from similar sectors such as hotels which only have a capacity rate of 59% (Toh et al., 2005).

Despite the industry's continual rise since 1990 both in popularity and revenues, the coronavirus pandemic, which has caused a severe impact on the tourism industry, devastated the cruise sector in particular. The first cluster of COVID-19 cases upon a cruise ship was found on Diamond Princess in February 2020, recording 712 infections and 14 deaths. The outbreak was highly publicised and monitored worldwide, spreading concern and fear about travel and transmission of the virus upon cruise ships. This scepticism, in turn, led to an increased number of people cancelling or postponing their voyages with cruise ships, and in the best interest of public health, cruise lines paused operations. The continuous uncertainty regarding the end of the pandemic led to extending the cruising hiatus for an extended period. At the same time, upon recommencement of operations, new laws and regulations have been imposed, limiting the capacity and deployment of cruise ships while the wider effects of the pandemic on world trade have also disrupted the supply of food, beverages and merchandise to the ships. The elongated pause and new restrictions have led to much higher uncertainty in demand estimates and, consequently, a need for robust pricing and allocation policies.

The cruise sector satisfies many of the underlying assumptions of revenue management practice. All of the characteristics discussed in section 1.1 are evident in the cruising sector.

- Cruise ships have a fixed number of cabins of each type, and a set number of lifeboat seats (fixed capacity)
- Once a specific voyage has sailed, customers can no longer travel on it (Perishable inventory)
- Customers can be segmented via age, itinerary, booking type etc. (Market/Customer segmentation)
- Itinerary tickets are sold in advance, in many cases up to two years before the embark date (Advanced sales/reservations)
- Demand for products varies during the booking period and can be estimated based on historical data (Time-variable demand)
- Marginal costs are low.

The cruise line industry has received little attention in academic literature. Li (2014) suggests that this is because cruising has a smaller scale of operations than airlines. Ayvaz-Cavdaroglu et al. (2019) and Talluri and Van Ryzin (2006) cite the widespread belief that cruise ships are essentially floating hotels; hence the techniques and applications developed for hotels could easily be applied to cruise ships. This view is no longer dominant, and Biehn (2006) explores the differences between the cruising and hospitality sectors.

Most importantly, cruising is unique in the aspect that it *combines* travel, transportation, hospitality, entertainment and shore excursions. A customer can choose from several combinations of cabin types, fares, itineraries and departure days. Cabin types can differ by size, amenities or location on the ship. For example, suites, balconies and inside cabins are all examples of distinct cabin types. Several different fare options are available such as premium, standard and saver where each class offers different perks. Of course the higher the fare the more amenities are included. Figure 5.2 displays the plethora of cruise routes that yield an even greater number of available itineraries to select. These itineraries are often repeated and thus offered on multiple departure days. The complexity introduced by these different characteristics of a cruise product makes the problem computationally and logistically expensive.

Readers might be more familiar with the notion of cruising, where ships travel to several ports, but embarkment and disembarkment happen in a single port. During the journey, the cruise ship calls at various ports, which passengers can visit on a schedule and then return to the ship to continue their voyage back to the original port where they commenced their cruise, thus completing a loop. Figure 5.3 is an example of such standard cruise. The thick lines represent the cruise ship's route, starting and finishing in Southampton with stops first at Norwegian Fjords and then at Iceland. While there are three nodes and three arcs in the figure representing the cruise ship's ports of

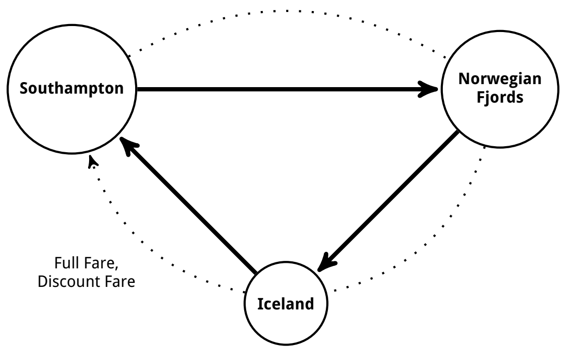


FIGURE 5.3: Representation of standard cruise itinerary

call and route, this does not constitute a cruise network. Because passengers can only embark or disembark in Southampton, even though the ship calls at two more ports, only a single itinerary is available to fill up the cruise ship's capacity, represented in the diagram via the loosely dotted line. Researchers and practitioners model such cruises as a single leg with multiple products.

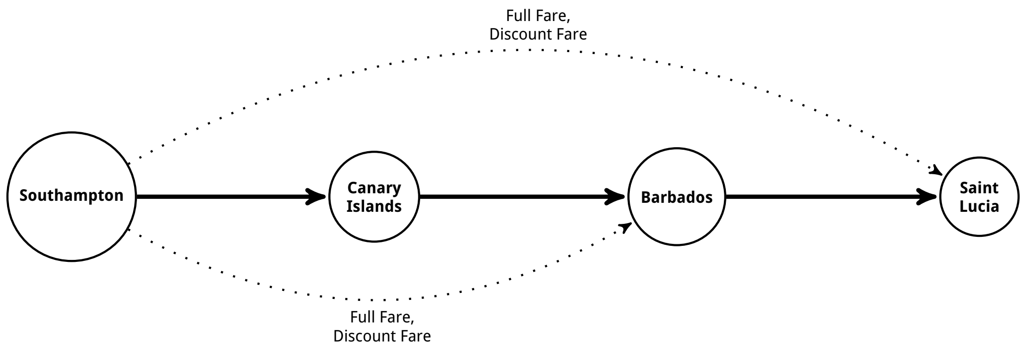


FIGURE 5.4: Representation of Type A cruise network

However, it is more common to consider a network of cruise itineraries. Cruise companies often allow passengers to embark at one port and disembark at another while offering a flight option back to the passenger's original point of embarkment. Such cruises use a network of resources. Figure 5.4 demonstrates such a network. As in the previous figure, thick lines represent the route that the ship follows while the dotted lines represent the itineraries on sale. The cruise commences in the UK and heads to the Caribbean. Customers can either travel until they reach Barbados or continue until they reach Saint Lucia. These are two distinct itineraries operated by the same ship on the same route. Such networks arise in cruises that travel to distant destinations and we refer to them as Type A. Another important example of a type A cruise network is the World cruise, where a ship travels around the globe. While the whole world route is on sale, there is usually not enough demand to fill the whole ship, so supplementary itineraries of shorter duration are offered to fill up the ship.

There exists a second type of cruise network which we refer to as Type B. Consider the case where a cruise route is strictly between only two ports, such as Southampton and New York. A cruise company deploys a cruise ship on this route, and operates it continuously; that is, the ship goes back and forth to these two ports only for a specified period. The company then offers four different origin - destination itineraries; Southampton to Southampton, Southampton to New York, New York to Southampton and New York to New York. Thus, a network arises as seen in figure 5.5. This is a three leg network where 5 itineraries are on offer, Southampton to New York is offered twice where the second journey is at a different date to the first.

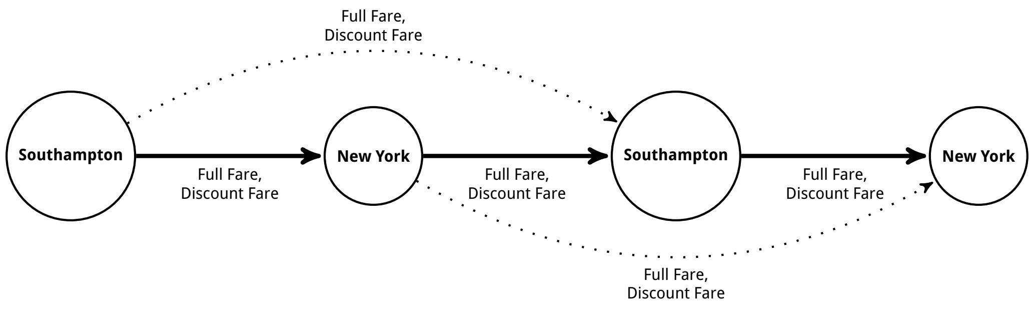


FIGURE 5.5: Representation of Type B cruise network

In any of the above cruises, revenue generated by a passenger is the sum of the cabin fare and passengers' on-board expenditure. On-board expenditure is the cash flow generated for a cruise operator from a passenger's spending on bars, restaurants, shops, spas, casinos and activities such as onshore excursions. On-board expenditure is integral to the total revenue generated by cruise lines. Cruise Market Watch (2018), a market watchdog, states that 'the average per passenger per day cruise expense is projected to be \$214.25, with \$152.12 per person per day ticket price and \$62.13 per person per day on-board spending'. This figure translates to 29% of the total revenue that a cruise line generates. At the same time Giese (2020) reports a 38% of cruise revenue attributed to on-board spend. While the numbers from different sources might not agree, it is clear that a significant proportion of the revenue results from on-board spend and as such, attention must be paid to the way it is forecast and modelled in the decision making process of allocating network resources to various demand requests.

5.2 Problem Statement

Based on the points discussed above, the revenue management problem that cruise companies are considering is the following: make pricing and cabin assignment decisions for the cabins according to price tiers in order to maximize the total revenue over the ship's whole route, which is calculated as the sum of the cabin sales and passengers' on-board spend. In other words, given the market demand and its elasticity, the company has to decide how much to charge for each cabin-type and fare-class-itinerary combination while also deciding how many cabins should be assigned to that cabin-type and fare-class-itinerary combination given the constraints on the number of cabins of each type that are available and a second constraint on the total number of people on board. This second constraint arises from the limit on the lifeboat capacity. Note that demand is forecast based on historical data according to customer requests for cabin amenities.

Notational convention In what follows scalars are denoted by lower-case letters, for example α, while vectors are always assumed to be column vectors and are typeset in bold lower-case letters, $\boldsymbol{x}$. The transpose of a vector is given by $\boldsymbol{x}^T$ while subscripts denote vector components, x_j. Matrices are represented by capital letters, $\boldsymbol{A}$ and similarly to vectors a_{ij} represents its component in the i^{th} row and j^{th} column. $\boldsymbol{a}_i^T$ is the i^{th} row of the matrix while $\boldsymbol{a}_j$ the j^{th} column.

Consider a cruise network that is comprised of multiple legs. A leg is a route in a cruise network where it commences at a port of embarkation and ends at a port of debarkation. Let the set L represent the set of all such legs such that $L = \{l \mid l = 1, ..., |L|\}$. On this network, a cruise company sells several itineraries. These itineraries can span a single or multiple legs and comprise set I which is defined as $I = \{i \mid i = 1, ..., |I|\}$. On each itinerary, different cabins types are available. Cabins of the same type are similar in location, size and amenities but are distinct from other types. Cabin types are defined as the set $K = \{k \mid k = 1, ..., |K|\}$. In addition, each cabin type can have different occupancy. The industry standard is double occupancy, but occupancy can be higher with bunk beds. In larger cabin types like suites, more than two beds might already be installed in the permanent configuration of the room. At the same time, single occupancy cabins also exist, while companies will often sell higher occupancy cabins to solo travellers to fill up the ship if there is not enough demand for the higher occupancy. Let the set $O = \{o \mid o = 1, ..., |O|\}$ denote the different occupancy available. Furthermore, different price tiers are available. Typical examples of price tiers are Full, Saver and Discount fares. Let price tiers comprise the set $P = \{p \mid p = 1, ..., |P|\}$.

Then a booking request for a product consists of an itinerary, which utilises a certain number of network legs, a cabin type, an occupancy and a price tier. Hence the set of products is given as $J = \{j \mid j = 1, ..., |J|\}$ where the cardinality of the set is given by $|J| = |I| \times |K| \times |O| \times |P|$.

Consider a finite time horizon $(0, T]$ during which sales can be made. Over this period, customers arrive according to a stochastic process and request to buy products $j \in J$. The revenue from selling product j is given by the fare r_j, where r_j is the sum of the fare and the on-board expenditure of the passenger mix as estimated from historical data. We assume no cancellations occur during the booking period, an unrealistic modelling assumption as it is a common phenomenon in the cruise industry, but simplifies our treatment of the problem considerably.

5.2.1 Capacity control formulation

Given the above notation and assuming that revenue for the different price tiers are fixed, the cabin and lifeboat capacity control problem to solve is the following.

$$\begin{aligned} \underset{x}{\text{maximise}} \quad & \boldsymbol{f}'\boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{Ax} \leq \boldsymbol{c} \\ & \boldsymbol{Bx} \leq \boldsymbol{e} \\ & \boldsymbol{x} \leq \boldsymbol{d} \\ & \boldsymbol{x} \geq 0 \end{aligned} \tag{5.1}$$

Let $\boldsymbol{x}$ be the column vector representing the decision vector such that $\boldsymbol{x} \in \boldsymbol{R}^{|J|}$. It represents the realised sales. The amount $\boldsymbol{f}'\boldsymbol{x}$ is the total revenue obtained from selling $\boldsymbol{x}$ different products across a cruise network. The capacity vector $\boldsymbol{c} \in \boldsymbol{R}^{|K|\times|L|}$ is the cabin type capacity vector. Its elements are the total number of each cabin type available at each leg of the network. The incidence matrix $\boldsymbol{A} \in \boldsymbol{R}^{|K|*|L|\times|J|}$ where $\boldsymbol{A}_{\{kl\}j}$ is equal to 1 when a product j comprises of cabin type k and uses leg l of the network and 0 otherwise. Matrix $\boldsymbol{B} \in \boldsymbol{R}^{|L|\times|J|}$ is a different construct to matrix $\boldsymbol{A}$ because it represents the number of lifeboat seats a product requires in contrast to the number of cabins a product requires. Thus while it also represents a capacity constraint, it is a distinct constraint concerning lifeboats with different structure. It is defined as $\boldsymbol{B}_{lj} = o_j$ where o_j is the occupancy of product j that utilises leg l of the network and 0 otherwise. Finally, the vector $\boldsymbol{e} \in \boldsymbol{R}^{|L|}$ represents the lifeboat seat capacity per leg of the network. The final two constraints ensure that the realised product sales do not exceed their respective demand and are non negative.

Since cabin types are set and cannot be altered between legs of the network, as opposed to aeroplane seats that can be altered, it is not feasible to employ the nested booking limits policy. Instead, we use the partitioned booking limits policy (see section 2.2).

5.2.2 Robust Formulation

Using the notation introduced in section 5.2, the maximum regret ρ for a cruise network is given by the optimal value of the below MIP.

$$
\begin{aligned}
\underset{\boldsymbol{z},\boldsymbol{x},\boldsymbol{d},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\delta}}{\text{maximise}} \quad & \boldsymbol{f}'\boldsymbol{z} - \boldsymbol{f}'\boldsymbol{x} && \text{(5.2a)}\\
\text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{c} && \text{(5.2b)}\\
& A\boldsymbol{z} \leq \boldsymbol{c} && \text{(5.2c)}\\
& B\boldsymbol{x} \leq \boldsymbol{e} && \text{(5.2d)}\\
& B\boldsymbol{z} \leq \boldsymbol{e} && \text{(5.2e)}\\
& 0 \leq \boldsymbol{x} \leq \boldsymbol{d} && \text{(5.2f)}\\
& 0 \leq \boldsymbol{z} \leq \boldsymbol{d} && \text{(5.2g)}\\
& \boldsymbol{d} \in \mathcal{U} && \text{(5.2h)}\\
& \sum_{s \in S} x_s \leq b_s \quad S \in \mathcal{S} && \text{(5.2i)}\\
& \boldsymbol{x} + M(1-\alpha) \geq \boldsymbol{d} && \text{(5.2j)}\\
& \sum_{s \in S} x_s \geq \beta b_s \qquad\qquad S \in \mathcal{S} && \text{(5.2k)}\\
& \boldsymbol{a}_i^T \boldsymbol{x} \geq \boldsymbol{c}_i \gamma_i && \text{(5.2l)}\\
& \boldsymbol{b}_i^T \boldsymbol{x} \geq \boldsymbol{e}_i \delta_i && \text{(5.2m)}\\
& \alpha_j + \sum_{S: j \in S} \beta_s + \sum_{l=1:a_l}^{|L|} \gamma_l + \sum_{l=1:a_l}^{|L|} \delta_l \geq 1 && \text{(5.2n)}\\
& \boldsymbol{\alpha} \in \{0,1\}^{|J|} && \text{(5.2o)}\\
& \boldsymbol{\beta} \in \{0,1\}^{|S|} && \text{(5.2p)}\\
& \boldsymbol{\gamma} \in \{0,1\}^{|L|} && \text{(5.2q)}\\
& \boldsymbol{\delta} \in \{0,1\}^{|L|} && \text{(5.2r)}
\end{aligned}
$$

where $M \geq \{\max_j d_j : \boldsymbol{d} \in \mathcal{U}\}$. Similarly, the minimum revenue ϕ is equal to the negative of the optimal value of 5.2 when $\boldsymbol{z} = 0$.

Constraints 5.2b and 5.2c ensure that the decision variables $\boldsymbol{z}$ and $\boldsymbol{x}$ do not violate the cabin type capacity. Similarly, constraints 5.2d and 5.2e ensure that the lifeboat capacity is not exceeded. Constraints 5.2f - 5.2h concern the demand. Constraints 5.2f and 5.2g enforce that the chosen sales will not exceed the demand. Constraint 5.2h stipulates that an uncertainty set characterises $\boldsymbol{d}$. As in the previous chapter, we examine polyhedral uncertainty sets. Constraint 5.2i is to facilitate that the booking limits b_s are not exceeded. Finally, the last set of constraints, which involves the binary variables α, β, γ, and δ ensures that the realised sales $\boldsymbol{x}$ equal the demand $\boldsymbol{d}$ unless the demand is censored. In constraint 5.2j $\alpha = 1$ only if $\boldsymbol{x} = \boldsymbol{d}$. Similarly, constraint 5.2k states that $\beta = 1$

Itinerary	**Cabin Type**	**Fare Class**	**Ocucpancy**			
			1	**2**	**3**	**4**
A - B	**Suite**	**Full**	£7,416	£7,499	£7,821	£8,140
		Saver	£6,437	£6,437	£6,759	£7,078
	Mini Suite	**Full**	£6,184	£6,276	£6,596	£6,917
		Saver	£5,632	£5,632	£5,952	£6,273
	Balcony	**Full**	£3,234	£4,183	£4,503	£4,952
		Saver	£2,880	£3,700	£4,020	£4,437
	Inside	**Full**	£2,001	£2,508	£2,925	£3,246
		Saver	£1,748	£2,186	£2,571	£2,892
B- C	**Suite**	**Full**	£7,644	£7,725	£8,044	£8,366
		Saver	£6,727	£6,727	£7,045	£7,367
	Mini Suite	**Full**	£6,512	£6,598	£6,918	£7,561
		Saver	£5,954	£5,954	£6,274	£6,917
	Balcony	**Full**	£3,315	£4,247	£4,600	£5,049
		Saver	£2,961	£3,764	£4,117	£4,534
	Inside	**Full**	£2,001	£2,508	£2,925	£3,246
		Saver	£1,748	£2,186	£2,603	£2,924
A - C	**Suite**	**Full**	£15,016	£14,531	£14,755	£16,294
		Saver	£ 12,353	£ 12,296	£ 13,591	£ 15,059
	Mini Suite	**Full**	£ 12,805	£ 12,048	£ 12,228	£ 13,597
		Saver	£ 11,510	£ 10,741	£ 12,785	£ 14,110
	Balcony	**Full**	£7,201	£9,051	£8,798	£ 10,672
		Saver	£5,381	£7,681	£8,483	£8,172
	Inside	**Full**	£3,848	£5,511	£5,892	£6,346
		Saver	£3,242	£4,064	£4,835	£5,947

TABLE 5.1: Revenue generated per product on offer for a two-leg cruise network

only if the booking limit for product j in bucket s has been reached while γ and δ equal 1 if on some leg of the network the cabin or lifeboat capacity respectively is reached (constraints 5.2l and 5.2m). Finally, constraint 5.2n joins these scenarios and guarantees that at least one occurs for every product.

We choose uncertainty sets to be either polyhedral or ellipsoidal. We define $\mathcal{U}_\mathcal{P}$ to be a polyhedral uncertainty set such that $U_P(\boldsymbol{d}) = [\boldsymbol{l}, \boldsymbol{u}]$ where $\boldsymbol{l}$ and $\boldsymbol{u}$ are the lower and upper bounds respectively and are computed as $\boldsymbol{l} = [l_j \mid l_j = \mu_j - 3\sigma_j \ \forall j \in J]$ and $\boldsymbol{u} = \{u_j \mid u_j = \mu_j + 3\sigma_j \ \forall j \in J\}$ where $\boldsymbol{\mu}, \boldsymbol{\sigma}$ are the mean and standard deviation estimates for each product j. Similarly, we define $\mathcal{U}_\mathcal{E}$ to be an ellipsoidal uncertainty set such that $U_E(\boldsymbol{d}) = (\boldsymbol{d} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{d} - \boldsymbol{\mu}) \leq \lambda$ where λ is a scalar.

5.2.3 Derivation of optimisation parameter values

For an instance of the formulations 5.1 and 5.2 to be solved and produce numerical results, several parameter values have to be estimated from data regarding cruise ships and bookings.

The parameters to be estimated are the following:

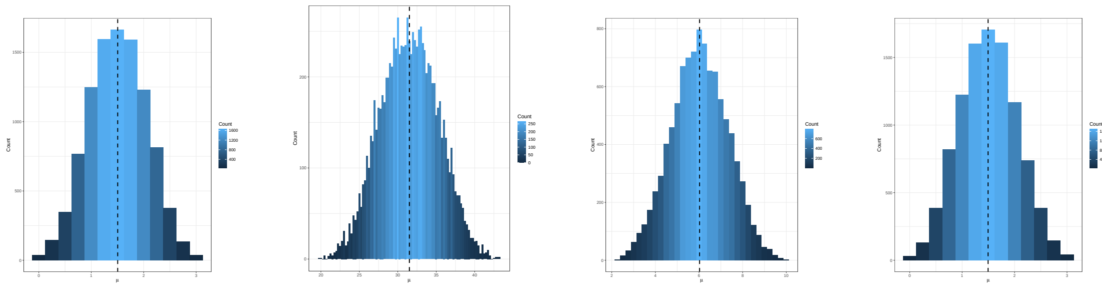
1. The vector $\boldsymbol{c}$ represents the total number of cabins per cabin type per leg of the network.
2. The vector $\boldsymbol{e}$ represents the total number of lifeboat seats on the ship excluding staff for a leg of the network.
3. The revenue associated with each product f_j. A product is the result of an itinerary, cabin type, occupancy and fare class combination. Thus the income related to the product is the product combination fare plus the expected onboard revenue to be generated from the occupants.
4. The lower and upper bounds on the demand for each product, $l_j \leq d_j \leq u_j$.
5. Infer the distribution parameters for simulating the booking process over the booking horizon.

Forecasting demand in the cruise industry can be very challenging as there are many factors to be considered, such as seasonality; time-of-day, day-of-week and week-of-year variability; demand dependencies between booking classes; sensitivity to pricing; demand volatility; schedule changes; truncation of historical demand data; reservation system limitations; and of course the impact from external shocks like the coronavirus pandemic.

Several techniques can be used to model demand ranging from simple exponential smoothing (ES) techniques to more advanced methods such as neural networks, principal component analysis and adaptive models. A common technique is using Poisson models to simulate the booking arrival process. Another is a Gamma distribution with Poisson random errors, which gives a negative binomial distribution for total demand.

Here we have two objectives; to estimate the aggregate demand per product and infer the booking curves during the booking horizon. To achieve these goals, we rely on sample booking data from the cruise industry. While there are itineraries with a considerable history of booking requests, there are also new or not so popular itineraries where limited data is available. Cabin type capacities, $\boldsymbol{c}$, and lifeboat seats, $\boldsymbol{e}$, per ship are publicly available information from owners' and shipyards' websites. All product booking parameters are estimated from historical booking data provided by a leading company in the sector. Historical booking data reveal the aggregate demand and fares paid per product combination. The expected onboard spending per passenger is assumed to be proportional to the product combination fare and thus can be inferred from an analysis of historical booking and onboard sales data.

Since the available historical data sample was small, we estimate the mean, $\hat{\boldsymbol{\mu}}$, and standard deviation, $\hat{\boldsymbol{\sigma}}$, of the products in offer, and then use the statistical technique of bootstrap sampling to assess the variability associated with these estimates. We followed the procedure below:



(A) Occ:1 (B) Occ:2 (C) Occ:3 (D) Occ:4

FIGURE 5.6: Histograms of the estimates of $\hat{\boldsymbol{\mu}}$ obtained from 10000 bootstrap samples for all four different occupancies of product AB - Suite - Full fare

1. From the original dataset observations, determine the demand values observed for each product combination.

2. Choose a large number B to produce B different bootstrap samples $Z_1^*, ..., Z_B^*$.

3. Obtain a distinct bootstrap dataset Z_i^* $i \in B$ by repeatedly resampling observations for each product from the original dataset with replacement. The size of the bootstrap sample must be equal to the size of the original data set.

4. A new estimate for the mean, μ_i^*, and standard deviation, σ_i^{*2}, is calculated for each bootstrap sample Z_i^* by

$$\mu_i^* = \frac{1}{|Z_i^*|} \sum_{r=1}^{|Z_i^*|} Z_{ir}^* \qquad \forall i \in B$$

$$\sigma_i^{*2} = \frac{1}{|Z_i^*| - 1} \sum_{r=1}^{|Z_i^*|} (Z_{ir}^* - \mu_i^*)^2 \qquad \forall i \in B$$

5. Repeat this B times, then we have $\mu_1^*, \mu_2^*, ..., \mu_B^*$ and $\sigma_1^{*2}, \sigma_2^{*2}, ..., \sigma_B^{*2}$. These are empirical bootstrap distributions of the sample mean and variance. Figure 5.6 is an example for the product combination of a double occupancy suite cabin at the full fare tier for an itinerary that utilises all the legs of the example network.

6. We compute the standard error and create confidence intervals for these estimators using the formulae,

$$\hat{\boldsymbol{\mu}} = Mean_B(\boldsymbol{\mu}^*) = \frac{1}{B} \sum_{i=1}^{B} \mu_i^*$$

$$SE_B(\hat{\boldsymbol{\mu}}) = \sqrt{\frac{1}{B-1} \sum_{i=1}^{B} \left(\mu_i^* - \frac{1}{B} \sum_{i'=1}^{B} \mu_{i'}^* \right)^2}$$

and similarly for $\hat{\boldsymbol{\sigma}}$.

Cabin Type	Fare Class	Occupancy	boot(100)		boot(1000)		boot(10000)		$\Gamma(\alpha)$	$\Gamma(\beta)$	$B(\alpha)$	$B(\beta)$
			$N(\mu)$	$N(\sigma)$	$N(\mu)$	$N(\sigma)$	$N(\mu)$	$N(\sigma)$				
Suite	**Select**	**1**	1.48	1.10	1.45	1.11	1.50	1.11	1.81	0.83	15	2
		2	42.18	8.16	43.07	7.84	43.01	7.81	30.33	1.42		
		3	9.14	2.93	9.48	3.43	9.46	3.45	7.52	1.26		
		4	2.16	1.50	2.02	1.43	1.98	1.41	1.96	1.01		
	Saver	**1**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
		2	4.36	1.02	4.48	1.11	4.51	1.13	16.06	0.28		
		3	1.00	0.80	0.98	0.83	1.01	0.82	1.51	0.67		
		4	0.49	0.50	0.51	0.50	0.50	0.50	1.01	0.50		
Mini Suite	**Select**	**1**	3.14	1.91	2.92	1.99	2.98	2.01	2.19	1.36	10	2
		2	60.06	9.68	61.28	9.65	61.04	9.47	41.58	1.47		
		3	9.51	2.25	9.48	2.29	9.54	2.30	17.25	0.55		
		4	2.98	1.46	3.03	1.38	2.99	1.41	4.47	0.67		
	Saver	**1**	1.10	0.80	1.00	0.83	0.99	0.82	1.48	0.67		
		2	17.09	2.75	16.53	2.87	16.50	2.86	33.27	0.50		
		3	2.92	0.80	3.01	0.84	3.01	0.82	13.56	0.22		
		4	0.47	0.50	0.52	0.50	0.50	0.50	0.98	0.50		
Balcony	**Select**	**1**	71.58	11.72	70.54	11.19	69.88	11.31	38.15	1.83	4	5
		2	874.69	120.29	857.63	111.64	863.56	111.19	60.32	14.32		
		3	43.26	11.67	42.02	12.52	41.91	12.46	11.31	3.71		
		4	13.32	3.40	13.56	3.38	13.50	3.42	15.54	0.87		
	Saver	**1**	16.17	2.47	16.02	2.55	16.01	2.59	38.36	0.42		
		2	199.11	25.95	196.89	26.67	197.13	25.85	58.14	3.39		
		3	9.87	2.85	9.61	2.93	9.50	2.87	10.93	0.87		
		4	3.19	0.80	3.00	0.81	3.00	0.82	13.52	0.22		
Inside	**Select**	**1**	52.42	11.83	53.40	11.97	53.41	11.66	21.00	2.54	2	4
		2	250.37	36.96	246.83	38.08	248.54	38.15	42.43	5.86		
		3	17.67	5.15	17.36	5.26	17.45	5.16	11.43	1.53		
		4	5.71	1.98	6.04	2.01	5.98	1.98	9.16	0.65		
	Saver	**1**	22.53	5.08	22.49	5.20	22.56	5.19	18.90	1.19		
		2	103.56	16.51	106.54	16.77	105.72	16.47	41.18	2.57		
		3	7.45	2.42	7.48	2.26	7.48	2.31	10.50	0.71		
		4	2.57	1.05	2.45	1.12	2.48	1.12	4.96	0.50		

TABLE 5.2: Bootstrap estimated values for distribution parameters $N(\mu), N(\sigma), \Gamma(\alpha), \Gamma(\beta), B(\alpha)$ and $B(\beta)$ for itineraries that utilise two legs of the example network

To simulate the booking period we follow the same procedure as we described in Chapter 4.1. Therefore we need to estimate the parameters of Gamma distibution. We set the parameters of the gamma distribution using the the boostrap estimators of mean and standard deviation, using the formulae,

$$\alpha = \hat{\mu}^2/\hat{\sigma} \tag{5.3}$$

$$\beta = \hat{\sigma}/\hat{\mu} \tag{5.4}$$

while these are not maximum likelihood estimators of the parameters, they do provide a convenient, fast and empirically well performing outcome hence they are our method of choice. Finally, the *beta* distribution parameters were chosen so that the shape of the pdf function fits the arrival pattern curve of customers for different cabin types.

Table 5.2 details the bootstrapped means and standard deviations using different B values, namely 100, 1000 and 10000. The bootstrap estimators' values are very close.

Thus, we expect that choosing to use the values of any B in the optimisation realisations and the simulation study will not significantly affect the results. We opt for $B = 10000$.

5.3 Simulating the booking horizon

We follow the same approach used in section 4.1. As discussed previously, the non-homogeneous Poisson arrival process allows for the flexibility to simulate the different arrival patterns of customers asking for different products.

Figure 5.7 gives the probability density function curves for each cabin type's booking arrivals pattern using the beta and gamma distributions parameters as inferred in section 5.2.3. The pattern for each cabin type is different. Suites and Mini-Suites that are higher valued resources have booking requests arriving late in the booking horizon. There is a zero probability of requests arriving in the first half of the booking horizon. When comparing the two high-value cabin types, it is clear that Suites booking requests arrive even later than those of Mini-Suites as the curve peaks almost at the end of the horizon.

On the other hand, the lower-yielding cabin types, Balconies and Insides, have their booking requests start arriving early and keep arriving almost throughout the whole period of the booking horizon and die out in the right tail of the distribution. The pdf yields a zero probability of a booking request arriving for any of the two cabin types in the last fifth of the booking horizon. Balconies differ from Insides because the bulk of requests for Balconies arrives later than Insides, just before the midpoint of the horizon.

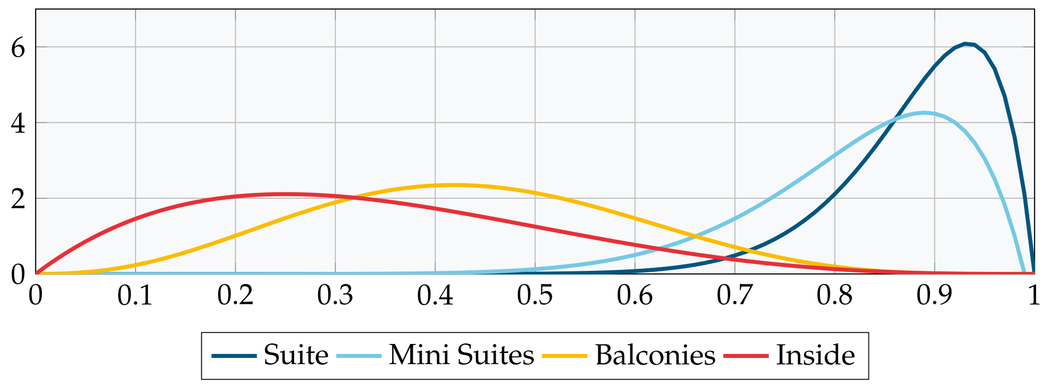


FIGURE 5.7: Inferred beta distribution for the shape of arrivals of booking requests per cabin type

5.4 Numerical Results

As described in section 5.2.3, to solve instances of the cruise optimisation formulations given by equations 5.1 and 5.2, we inferred parameter values for a type A cruise network as seen in Figure 5.8.

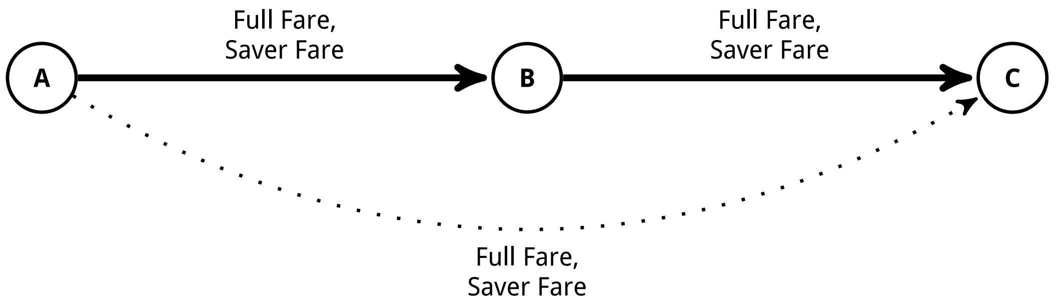


FIGURE 5.8: Type A cruise network example

The cruise network example we choose resembles the structure of several real life cruise networks and is similar to the network from which we obtained historical data to set the optimisation parameters. The network consists of two legs, AB and BC, and offers three itineraries, $I = \{AB, BC, AC\}$. The cruise ship deployed on the network has four different cabin types; Suites, Mini-Suites, Balconies and Insides, $K = \{S, M, B, I\}$, where each cabin type is assumed to be able to hold up to four occupants, $O = \{1, 2, 3, 4\}$. Finally, the cruise company offers two price tiers, a Full fare and a Saver fare, $P = \{F, S\}$. Hence, there exist 96 distinct products on sale on this cruise network.

Because of COVID restrictions a limit to the total number of passengers allowed onboard of the cruise ship was often placed on the cruise companies. To facilitate this restriction, the lifeboat passenger capacity constraint right hand side value can be altered. To that end we investigate two scenarios, the normal case where a 100% passenger capacity is allowed and a second scenario where only a 75% passenger capacity is allowed.

5.4.1 DLP

We first investigate the results of the DLP relaxation to the 5.1 formulation. Table 5.3 displays the derived booking limits for both lifeboat capacity scenarios. As expected, the DLP reserves more seats for higher-yielding products. For the 100% lifeboat scenario, the Saver price tier is closed for Suites and Balconies. Since Suites are the highest revenue yielding cabin type and Balconies are the most popular, it is more profitable not to offer the lower price tier. The Saver price tier is open mainly for the Inside cabin type, indicating that there might not be enough demand for Full fare passengers to fill up Inside cabins. Further to Inside cabins, a few Saver fare seats are also reserved for Mini-Suite cabins. Since Mini-Suites are higher-yielding than balconies, if there is still lifeboat capacity available, it is more profitable to allow Mini-Suite than Balconies Saver fare passengers on board.

Cabin Type	Price Tier	Occupancy	A-B	A-C	B-C
Suite	Full	1	2	1	2
		2	43	5	43
		3	9	0	9
		4	2	2	2
	Saver	1	0	0	0
		2	0	0	0
		3	0	0	0
		4	0	0	0
Mini Suite	Full	1	3	1	3
		2	61	11	61
		3	9	0	10
		4	3	0	3
	Saver	1	1	1	0
		2	0	0	0
		3	0	1	0
		4	0	1	0
Balcony	Full	1	0	0	0
		2	783	408	786
		3	0	0	0
		4	8	7	5
	Saver	1	0	0	0
		2	0	0	0
		3	0	0	0
		4	0	0	0
Inside	Full	1	53	18	53
		2	249	78	249
		3	17	7	17
		4	6	2	5
	Saver	1	0	0	1
		2	17	0	10
		3	0	0	7
		4	0	1	0
Objective Value			€14,845,932.73		

(A) 100% Lifeboat occupancy scenario

Cabin Type	Price Tier	Occupancy	A-B	A-C	B-C
Suite	Full	1	2	1	2
		2	43	18	43
		3	0	0	0
		4	0	0	0
	Saver	1	0	0	0
		2	0	0	0
		3	0	0	0
		4	0	0	0
Mini Suite	Full	1	3	1	3
		2	61	25	61
		3	0	0	0
		4	0	0	0
	Saver	1	1	1	1
		2	0	0	0
		3	0	0	0
		4	0	0	0
Balcony	Full	1	70	35	70
		2	677	408	677
		3	0	0	0
		4	0	0	0
	Saver	1	16	0	16
		2	0	0	0
		3	0	0	0
		4	0	0	0
Inside	Full	1	53	18	53
		2	0	20	0
		3	0	0	0
		4	0	0	0
	Saver	1	23	7	23
		2	0	0	0
		3	0	0	0
		4	0	0	0
Objective Value			€12,820,513.87		

(B) 75% Lifeboat occupancy scenario

TABLE 5.3: Booking limits derived using the DLP under mean demand

In the 75% lifeboat capacity scenario, it is interesting to see that the DLP still reserves more seats for the higher-yielding price tier products, but now the saver fare is also open for the single occupancy Balcony cabin type. The effect of the reduced lifeboat capacity is evident by the zero booking limits allowed for any product where occupancy is equal to 3 or 4 and the significant reduction of booking limits for all products involving Inside cabins. This is an expected result, as inside cabins are the lowest yielding. Of course, the objective value is lower compared to the full capacity scenario as fewer passengers are allowed on board.

In both scenarios, more cabins are assigned on itineraries that utilise single legs of the network instead of products that utilise both legs, while symmetry between the A-B and B-C legs of the network is also evident as the optimisation parameters for these products are very similar.

5.4.2 Robust Measures

For the same scenarios, we also implement the robust formulations described in equation 5.2. Table 5.4 shows the allocation using maximin revenue robust control with Polyhedral uncertainty set.

Cabin Type	Price Tier	Occupancy	A-B	A-C	B-C
Suite	Full	1	0	0	0
		2	37	15	37
		3	7	1	7
		4	1	0	1
	Saver	1	0	0	0
		2	1	2	1
		3	0	0	0
		4	0	0	0
Mini Suite	Full	1	1	0	1
		2	52	19	53
		3	7	1	8
		4	0	0	3
	Saver	1	0	0	1
		2	10	1	3
		3	1	0	2
		4	0	0	0
Balcony	Full	1	8	0	6
		2	790	354	793
		3	18	2	14
		4	11	4	10
	Saver	1	4	0	1
		2	6	1	12
		3	6	1	7
		4	1	0	1
Inside	Full	1	49	14	47
		2	222	67	223
		3	15	5	14
		4	4	0	5
	Saver	1	12	4	16
		2	49	0	48
		3	6	0	5
		4	1	0	0
Objective Value			€14,655,343.12		

(A) 100% available lifeboat occupancy scenario

Cabin Type	Price Tier	Occupancy	A-B	A-C	B-C
Suite	Full	1	0	0	0
		2	37	23	37
		3	2	0	1
		4	0	0	1
	Saver	1	0	0	0
		2	1	1	1
		3	0	0	0
		4	0	0	0
Mini Suite	Full	1	1	0	1
		2	54	29	54
		3	0	0	0
		4	0	0	1
	Saver	1	0	0	0
		2	8	0	7
		3	0	0	0
		4	0	0	0
Balcony	Full	1	62	27	61
		2	742	354	745
		3	0	0	0
		4	0	0	0
	Saver	1	14	2	14
		2	3	2	1
		3	0	0	0
		4	0	0	0
Inside	Full	1	46	14	43
		2	1	15	0
		3	0	0	0
		4	0	0	0
	Saver	1	18	5	19
		2	0	0	0
		3	0	0	0
		4	0	0	0
Objective Value			€12,696,498.34		

(B) 75% available lifeboat occupancy scenario

TABLE 5.4: Booking limits derived using the maximin revenue criterion and polyhedral uncertainty set

For the 100% lifeboat scenario, in contrast to the DLP, the maximin revenue control does not close off the saver price tier. As with the DLP, the maximin revenue assigns the most saver products to the Inside cabin type. The symmetry of the allocation to A-B and B-C itineraries is still visible, especially on the Full price tier allocations.

On the other hand, for the 75% scenario, the maximin revenue control also closes almost all products with occupancy higher than three passengers. Furthermore, the reduction in passengers is achieved by reducing the booking limits of the Full fare inside cabins.

The maximin revenue yields a 1.3% lower objective value than the DLP, but this is expected as the DLP returns an upper bound of the objective function and the minimax revenue is a measure concerning itself with the worst-case scenario.

The second robust measure we look at is the minimax regret. Table 5.5 shows the allocation using minimax regret robust control with Polyhedral uncertainty set.

This measure, for the 100% lifeboat capacity, yields different booking limits to both the DLP and the maximin revenue measure. The stark difference of the minimax regret to the other policies is the open Saver price tier for all products, including the Suite cabin type. In similar fashion to the other methods there exists a symmetry to the allocations of A-B and B-C itineraries with lower allocation to A-C itinerary.

Cabin Type	Price Tier	Occupancy	A-B	A-C	B-C
Suite	Full	1	3	2	3
		2	25	11	27
		3	9	1	9
		4	3	1	4
	Saver	1	2	0	0
		2	3	1	3
		3	3	0	1
		4	0	0	1
Mini Suite	Full	1	4	2	4
		2	38	18	38
		3	11	1	11
		4	2	1	4
	Saver	1	0	0	0
		2	11	2	7
		3	2	0	4
		4	0	0	0
Balcony	Full	1	2	0	4
		2	697	466	694
		3	8	0	5
		4	12	1	10
	Saver	1	6	2	9
		2	6	0	10
		3	1	0	4
		4	5	0	1
Inside	Full	1	62	16	54
		2	210	71	210
		3	15	3	21
		4	8	0	7
	Saver	1	16	0	13
		2	35	0	41
		3	7	0	9
		4	5	0	3
	Objective Value		€6,061,575.64		

(A) 100% Lifeboat Occupancy

Cabin Type	Price Tier	Occupancy	A-B	A-C	B-C
Suite	Full	1	0	1	2
		2	34	22	30
		3	1	1	2
		4	1	0	1
	Saver	1	0	0	2
		2	0	0	2
		3	0	0	0
		4	0	0	0
Mini Suite	Full	1	3	1	3
		2	45	23	44
		3	5	0	3
		4	1	0	1
	Saver	1	1	0	0
		2	7	4	8
		3	1	0	1
		4	0	0	0
Balcony	Full	1	79	41	82
		2	639	358	569
		3	5	6	20
		4	4	2	6
	Saver	1	19	8	21
		2	15	7	18
		3	0	3	10
		4	2	0	1
Inside	Full	1	53	19	57
		2	6	20	15
		3	0	0	6
		4	2	0	1
	Saver	1	19	8	26
		2	0	1	9
		3	1	0	0
		4	1	0	0
	Objective Value		€6,283,604.69		

(B) 75% Lifeboat Capacity

TABLE 5.5: Booking limits derived using the minimax regret criterion and polyhedral uncertainty set

For the 75% scenario, the minimax regret control in contrast to the previous measures does not close all products with occupancy higher than three passengers. It does however, in the same fashion as before, achieve the required reduction in passengers by reducing the booking limits of the Full fare inside cabins from 448 total inside cabins in each network leg to 130 and 162 Inside cabins per A-B and B-C legs respectively.

The minimax regret for the full lifeboat capacity, yields a 3.5% lower objective value than the 75% scenario.

5.4.3 Simulation Comparison

Table 5.6 details the simulation results for the Cruise network example. In the comparison we have 5 booking policies, run on two scenarios, 75% and 100% capacity.

Policy	Objective Value	Expected Revenue	Simulated Revenue				90% Confidence Intervals		5^{th} Percentile	10^{th} Percentile	15^{th} Percentile
			Min	Average	Max	StD					
DLP-Low	£ 11,622,461	£ 14,587,345	£ 12,569,491	£ 14,215,799	£ 14,567,552	£ 260,516	£ 14,209,738	£ 14,221,860	£ 13,659,519	£ 13,878,684	£ 14,006,855
DLP	£ 14,916,945	£ 14,845,933	£ 11,883,201	£ 13,936,790	£ 14,700,050	£ 472,551	£ 13,925,796	£ 13,947,785	£ 13,013,725	£ 13,268,028	£ 13,439,822
DLP-High	£ 14,845,933	£ 14,916,945	£ 11,418,363	£ 13,568,225	£ 14,722,545	£ 583,899	£ 13,554,640	£ 13,581,810	£ 12,512,598	£ 12,763,699	£ 12,917,801
$\boldsymbol{\phi}^*$	£ 14,655,343	£ 14,698,044	£ 12,422,538	£ 14,259,941	£ 14,684,376	£ 335,976	£ 14,252,124	£ 14,267,758	£ 13,558,251	£ 13,801,601	£ 13,933,605
$\boldsymbol{\rho}^*$	£ 6,061,576	£ 14,758,951	£ 11,702,577	£ 13,706,422	£ 14,555,914	£ 566,030	£ 13,693,253	£ 13,719,591	£ 12,689,125	£ 12,915,162	£ 13,071,297

(A) 100% Lifeboat Occupancy

Policy	Objective Value	Expected Revenue	Simulated Revenue				90% Confidence Intervals		5^{th} Percentile	10^{th} Percentile	15^{th} Percentile
			Min	Average	Max	StD					
DLP-Low	£ 9,570,468	£ 11,302,476	£ 9,518,826	£ 11,007,957	£ 11,297,847	£ 205,282	£ 11,003,181	£ 11,012,733	£ 10,581,471	£ 10,744,085	£ 10,822,386
DLP	£ 12,898,480	£ 12,820,514	£ 10,368,327	£ 12,211,501	£ 12,767,503	£ 413,431	£ 12,201,882	£ 12,221,120	£ 11,377,327	£ 11,596,554	£ 11,747,945
DLP-High	£ 12,820,514	£ 12,898,480	£ 9,826,151	£ 11,827,398	£ 12,769,970	£ 555,942	£ 11,814,463	£ 11,840,332	£ 10,817,837	£ 11,035,845	£ 11,196,740
$\boldsymbol{\phi}^*$	£ 12,696,498	£ 12,696,498	£ 10,667,266	£ 12,409,143	£ 12,696,498	£ 279,352	£ 12,402,644	£ 12,415,642	£ 11,816,937	£ 12,026,170	£ 12,150,264
$\boldsymbol{\rho}^*$	£ 6,283,605	£ 12,460,978	£ 10,448,838	£ 12,018,735	£ 12,384,052	£ 255,045	£ 12,012,801	£ 12,024,669	£ 11,467,616	£ 11,674,716	£ 11,823,425

(B) 75% Lifeboat Occupancy

TABLE 5.6: Simulated Revenues for Cruise network example

The full passenger capacity scenario reveals a surprising result; the policies that are extremely risk-averse yield the best results. The maximin revenue control, $\boldsymbol{\phi}^*$, yields the highest average simulated revenue and the worst-case DLP has the highest simulated revenue in the lower percentiles of the revenue distribution. The DLP and minimax regret, $\boldsymbol{\rho}^*$, control policies perform similarly but the DLP outperforms the robust control. Since the main difference in the structure of these controls was that the $\boldsymbol{\rho}^*$ is leaving products in the Saver class open that the othe policies close and seems to deteriorate the average performance.

The limited passenger capacity scenario (75%) showcases the power of robust controls. The maximin revenue control, $\boldsymbol{\phi}^*$, yields the highest average simulated revenue and the highest simulated revenue in the low percentiles of the revenue distribution. The DLP closely follows the maximin revenue. The fact that the two control policies yield similar simulated results is not very surprising since they share the same structure, i.e. they close the same products and only differ in the number of cabins they assign to open products. The minimax regret, $\boldsymbol{\rho}^*$, also performs well and yields revenue, close to albeit lower than the DLP. It still however outperforms the booking limits yielded by the worst-case DLP scenario and the over optimistic best-case DLP.

5.5 Conclusion

The simulation results revealed some unexpected results; First, as expected the scenario that restricts capacity to 75% forces all policies to limit the passengers on board by closing the lowest yielding products, Inside cabins on the Saver fare class. Furthermore, as also expected, it yields lower simulated revenue for all policies. The surprising result is the overwhelming success of the maximin revenue control policy which outperforms all others both in average simulated revenue but also has the highest revenue on the lower percentiles of the revenue distribution.

This is also true in the full capacity scenario. The maximin revenue assigns the most cabins to the Balcony cabin type for both Full and Saver classes. These products enjoy the highest demand and thus seem to be more important than the higher-value yielding Suites and Mini-Suites that are way lower in numbers. On the other hand the minimax regret suffers from the fact that it has allowed the Saver class for all Cabin types to remain open. If the Saver class for Suites and Mini Suites remained closed and those capacities were allocated to other products, predominantly Balconies, then the structure pf the booking limits would be similar to the one by the DLP and the maximin revenue, but it would probably lead to higher simulated revenue.

5.5.1 Future Work

One major simplification assumption made in our methodology is that prices remain constant and we are only considering the cabin allocation problem. If pricing is volatile then one could either treat the whole problem as a price-based problem and adopt treatments like the ones by Ayvaz-Cavdaroglu et al. (2019) and Maddah et al. (2010). Of course this approach would then lack the dimension of resource allocation.

A different approach would be to attempt to optimise cabin allocation and pricing at the same time by including a pricing term in the objective function as done by Beck et al. (2021).

A third approach, that would be a direct expansion to the methodology presented here, would be to remain in a cabin allocation formulation and consider splitting fare classes to pricing buckets. That is increase the number of elements in the fare class set, F, to include elements that denote different price ranges within a fare class denomination. For example, one could decide that the the prices of fare class f are strictly in the range $[a, b]$. Then the interval $[a, b]$ can be split up in p number of portions. Then the set of price tiers would expand in size by a factor p. This would increase the number of products available therefore expanding the size of the problem.

Further to the modifications discussed above, another important modelling question arising is whether should price tiers be allowed to overlap. If they are allowed to overlap, how does one model or impose constraints on such overlaps? Furthermore, in such case it is very likely that there are going to be migrations of customers between cabin or fare segments, a situation that business practitioners refer to as *cannibalisation*. What are the rules and assumptions that dictate such migrations? Are there enough data to model such movements? How can a vendor detect them and how can he manage them to maximise revenue?

Chapter 6

Conclusions and Future Work

6.1 Research outcomes

In this research project, we expand current methodologies used to tackle the problem of quantity-based network revenue management to carry out robust capacity control in both an airline and cruise network settings. To achieve this goal, we first gained a comprehensive knowledge of the problems by familiarising ourselves with traditional and state-of-the-art model formulations and identifying gaps in the current literature. In the robust capacity control problem, we add to the formulation by introducing ellipsoidal uncertainty sets, applying a faster and more efficient optimisation routine in the form of a tailored genetic algorithm, and conducting an expanded numerical comparison of models and policies.

We further discuss the application of capacity control in a cruise application, an industry where revenue management is well suited to be applied but has received little academic attention. We introduce a straightforward formulation for a multiple leg cruise network problem that, in addition to the limited resources of lifeboat seats, cabin types and occupancy of the ship, also considers multiple ports of embarkation and debarkation, a level of complexity that, to the best of our knowledge has not been dealt with before in the cruise revenue management literature. We further expand the novelty of the formulation by introducing robust controls to the formulation and employing polyhedral uncertainty sets to characterise demand. We discuss the alterations that needed to be made to the genetic algorithm to facilitate the added complexity of the cabin *and* lifeboat capacity constraints, and we conduct a simulation study of the revenue.

The Literature Review (Chapter 2) distinguishes between traditional risk-neutral methodologies and risk-averse and robust modelling. In the Classical Revenue Management approach, the problem is tackled by assuming that demand follows a known statistical distribution and that the decision-maker is risk-neutral and the methods find the strategy that optimises the expected revenue. The problem can be formulated as a dynamic program but is easier solved by approximating it, either by dissecting the network into many single-resource RM problems that are considerably easier to handle or by static mathematical programming models. We focus on the latter case and describe the traditional DLP, EMR and RLP models that are among the most researched and implemented in the industry. Booking Control policies that dictate the acceptance or rejection of booking requests are also introduced.

We discuss more recent approaches that aim to relax the risk-neutrality assumption of the traditional models. Research into methods that use robust optimisation approaches are almost exclusively focused on the single-leg example. To the best of our knowledge, the only work to have applied a robust approach to a capacity control network up to this time is by Perakis and Roels (2010). We also investigate a third approach, which is to model customers' behaviour and their choice of products over a set of products available to each type of customer. Customer choice models developed so far for the network setting are based on the DLP model to create the set of offered products to customers while there is no published work on robust customer choice network revenue management. Hence we discuss Robust Optimisation methodology and conclude the chapter by identifying two areas of *future research*: a) *robust/risk-averse network revenue management* and b) *robust customer-choice*.

In Chapter 3 we introduce a comprehensive and universal mathematical notation for all the booking control policies, seat-allocation models, booking acceptance algorithms and demand simulation models that we collected from the different works in literature. We expand the methodology by introducing ellipsoidal uncertainty sets in the inner problem and suggesting a genetic algorithm to optimise the outer problem.

In Chapter 4 we present our numerical results where we built upon Perakis and Roels (2010) work and reproduced their results for the two robust seat allocation models they introduce. We show that our genetic algorithm performs better than their local gradient algorithm by producing 3% higher simulated revenues on average. Furthermore, in a similar fashion to de Boer et al. (2002) which describes a comparison of 2 traditional seat-allocation models, we built a comparison of 5 seat allocation models, of which two are the robust controls we introduced in Chapter 3. We implement three different booking control policies and simulate the booking horizon to evaluate the performance of each model under these policies.

In Chapter 5 we present our novel contributions to the cruise network capacity control. In addition to the multiple resources considered by other formulations such as lifeboat seats, cabin types and occupancy, we allow the network considered to span multiple itineraries. We introduce robust controls for the network formulation characterised by polyhedral uncertainty sets and perform a simulation on the booking horizon to assess the performance of the formulations in terms of revenue. The simulation results revealed that the maximin revenue measure performed the best as it assigned higher volume of Balcony cabins to Full Fare customers and closed lower yielding products such as Saver fare on Inside cabins.

6.2 Limitations and Future Work

We have already implemented several models and booking control strategies. However, as mentioned in section 4.7, our experiments can be improved by implementing new booking controls, specifically DAVN. A further enhancement to the investigation is to implement dynamic controls. That is to optimise the network at fixed points during the booking horizon instead of using the static controls that we are currently implementing, which only optimise the network once at the beginning of the booking horizon. Finally, a different avenue to explore would be to expand our experiments to include different parametrisation of the demand model used to simulate the arrival pattern of booking requests and investigate the effect of a lower fare spread or increased demand variance, as seen in Tables A.2 and A.3. These extensions could provide better approximations of real-life systems and allow a more comprehensive comparison of the methods under investigation.

An immediate step to improve this body of research is to reformulate the robust controls seen in Chapter 5 by introducing ellipsoidal uncertainty sets to problem 5.2. The motivation behind this is the link of ellipsoidal uncertainty sets to risk-measures such as VaR as argued by Natarajan et al. (2009). We are particularly interested in investigating the new robust formulation's computational tractability, computation speed, and theoretical properties.

A more significant extension to this work would be the investigation of robust controls with customer behaviour models. Current academic and industry literature suggests that companies and consumers focus on the *personalisation* of the product offer. This is clearly an area where customer-choice network revenue management could impact significantly. Specifically, we are interested in modelling the arrival of customers, the probability of a customer choosing a product and the inner assortment problem of which products should be offered to the customer. These aspects of the problem are usually modelled by specifying a probabilistic demand model, estimating probabilities from booking data observed in the past and the assortment problem is usually solved as a DLP (Davis et al., 2014). These approaches are prone to data uncertainty, which motivates the use of a robust approach.

Appendix A

Airline Distributions Parametrisation

The following tables present the parametrisation of demand distributions used in our numerical experiments in Chapter 4.

OD Pair	Class 3, $\alpha = 5, \beta = 6$				Class 3, $\alpha = 2, \beta = 5$				Class 1, $\alpha = 2, \beta = 13$			
	p	γ	μ	σ	p	γ	μ	σ	p	γ	μ	σ
1	80	1.6	50	9.01	80	2	40	7.75	3	0.1	30	18.17
2	80	2	40	7.75	50	2	25	6.12	2	0.1	20	14.83
3	60	2	30	6.71	72	3	24	5.66	2	0.1	20	14.83
4	60	2	30	6.71	40	2	20	5.48	2	0.1	20	14.83
5	60	2	30	6.71	60	3	20	5.12	6	0.3	20	9.31
6	80	1.6	50	9.01	80	2	40	7.75	6	0.2	30	13.42

TABLE A.1: Initial Demand Parametrisation for $B(\alpha, \beta)$, $\Gamma(p, y)$, and $N(\mu, \sigma)$ distributions

OD Pair	Class 3, $\alpha = 5, \beta = 6$				Class 3, $\alpha = 2, \beta = 5$				Class 1, $\alpha = 2, \beta = 13$			
	p	γ	μ	σ	p	γ	μ	σ	p	γ	μ	σ
1	20	0.4	50	13.23	20	0.5	40	10.95	3	0.1	30	18.17
2	20	0.5	40	10.95	5	0.2	25	12.25	2	0.1	20	14.83
3	15	0.5	30	9.49	18	0.75	24	7.48	2	0.1	20	14.83
4	15	0.5	30	9.49	10	0.5	20	7.75	2	0.1	20	14.83
5	15	0.5	30	9.49	15	0.75	20	6.83	6	0.3	20	9.31
6	20	0.4	50	13.23	20	0.5	40	10.95	6	0.2	30	13.42

TABLE A.2: Increased Variance of demand Parametrisation of $B(\alpha, \beta)$, $\Gamma(p, y)$, and $N(\mu, \sigma)$ distributions for Fare Classes 2 and 3

OD number	Origin-Destination	Fare class 3	Fare class 2	Fare class 1
1	A-B	£75 (1)	£125 (7)	£175 (13)
2	A-C	£130 (2)	£170 (8)	£220 (14)
3	A-D	£200 (3)	£320 (9)	£440 (15)
4	B-C	£100 (4)	£150 (10)	£210 (16)
5	B-D	£160 (5)	£200 (11)	£250 (17)
6	C-D	£80 (6)	£110 (12)	£160 (18)

TABLE A.3: Smaller fare value spread

Appendix B

Mathematical Background

B.1 Introduction to Robust Optimisation

Consider the following uncertain linear optimization problem,

$$\max\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\} \tag{B.1}$$

where $\mathbf{c} \in \mathbb{R}^n$ are the coefficients of the decision variables $\mathbf{x} \in \mathbb{R}^n$. The column vector $\mathbf{b} \in \mathbb{R}^m$ contains the right hand side values of the constraints and $\mathbf{A} \in \mathbb{R}^{m\times n}$ is the constraint matrix.

It is assumed that data uncertainty only affects the elements in matrix $\mathbf{A}$. This is because uncertainty in the objective function can be treated by reformulating the original uncertain linear problem to an uncertain linear problem with certain objective. Introduce a new variable t and rewrite the uncertain problem as,

$$\max\{\mathbf{t} \mid \mathbf{t} - \mathbf{c}^T\mathbf{x} \leq 0,\ \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$$

The data uncertainty is characterised as follows. Consider a particular row i of the matrix $\mathbf{A}$ and let J_i represent the set of coefficients in row i that are subject to uncertainty. Each entry a_{ij}, $j \in J_i$ is modelled as a symmetric and bounded random variable $\tilde{a}_{ij}$, $j \in J_i$ (as introduced by Ben-Tal and Nemirovski (2000)) takes values in the interval $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$. Associated with the uncertain data $\tilde{a}_{ij}$, we define the random variable $\xi_{ij} = (\tilde{a}_{ij} - a_{ij})/\hat{a}_{ij}$, which obeys an unknown but symmetric distribution, and takes values in $[-1, 1]$.

B.2 Robust Counterpart Optimisation

We consider the following uncertain linear optimisation problem,

$$\begin{aligned} \text{maximise} \quad & c^T x \\ \text{subject to} \quad & \sum_{j=1}^{n} \tilde{a}_{ij} x_j \leq \tilde{b}_i \qquad i = 1, ..., m \end{aligned} \tag{B.2}$$

where $\tilde{a}_{ij}$ and $\tilde{b}_i$ represent the true value of the parameters which are subject to uncertainty. Assuming that the uncertainty in each constraint is independent, consider the ith constraint of problem B.2 where both the LHS and RHS parameters are subject to uncertainty. We define the uncertainty as follows,

$$\tilde{a}_{ij} = a_{ij} + \xi_{ij} \hat{a}_{ij} \quad \forall j \in J_i \tag{B.3a}$$

$$\tilde{b}_i = b_i + \xi_{i0} \hat{b}_i \tag{B.3b}$$

where a_{ij} and bi represent the nominal value of the parameters while $\hat{a}_{ij}$ and $\hat{b}_i$ represent the constant perturbations. J_i is the index subset that contains the variable indices that have uncertain coefficients. $\xi_{i0}, \xi ij \forall i, \forall j \in J_i$ are random variables which are subject to uncertainty. Hence, the ith constraint becomes,

$$\sum_{j \notin J_i} a_{ij} x_j + \sum_{j \in J_i} \tilde{a}_{ij} x_j \leq \tilde{b}_i \tag{B.4}$$

which can be formulated by substituting in equations (B.3a) and (B.3b),

$$\sum_{j} a_{ij} x_j + \left[-\xi_{i0} \hat{b}_i + \sum_{j \in J_i} \xi_{ij} \hat{a}_{ij} x_j \right] \leq b_i \tag{B.5}$$

Under the robust optimisation methodology we introduce an uncertainty set $\mathcal{U}$ and aim to find solutions that remain feasible no matter the value ξ variables take. Replacing the original uncertain constraints with the robust counterpart constraints (B.5), the Robust Counterpart of the original problem is obtained,

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & \sum_{j} a_{ij} x_j + \left[\max_{\xi \in \mathcal{U}} \left\{ -\xi_{i0} \hat{b}_i + \sum_{j \in J_i} \xi_{ij} \hat{a}_{ij} x_j \right\} \right] \leq b_i \qquad \forall i \end{aligned} \tag{B.6}$$

B.3 Uncertainty Sets

As described by Li et al. (2011).

Definition B.1 (Box Uncertainty Set)**.** The box uncertainty set is described using the ∞-norm of the uncertain data vector as follows:

$$\mathcal{U}_\infty = \{\xi \mid \|\xi\|_\infty \leq \Psi\} = \{\xi \mid |\xi| \leq \Psi, \ \forall \, j \ \in \ J_i\} \tag{B.7}$$

where Ψ is the adjustable parameter controlling the size of the uncertainty set.

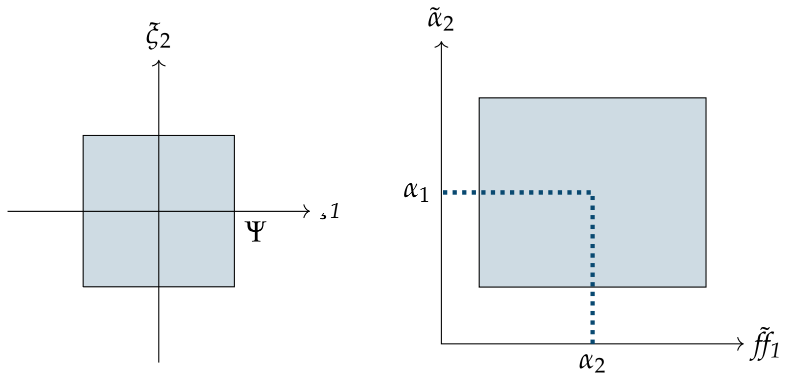


FIGURE B.1: Illustations of box uncertainty set

Definition B.2 (Ellipsoidal Uncertainty Set)**.** The ellipsoidal uncertainty is described using the 2-norm of the uncertain data vector as shown in Figure B.2,

$$\mathcal{U}_2 = \{\xi \mid \|\xi\|_2 \leq \Omega\} = \left\{\xi \mid \sqrt{\sum_{j \, \in \, J_i} \xi_j^2} \leq \Omega\right\} \tag{B.8}$$

where Ω is the adjustable parameter controlling the size of the uncertainty set. Note that it is known from geometry that for the bounded uncertainty $\xi_j \ \in \ [-1, 1]$, when $\Omega \ \geq \ (|J_i|)^{\frac{1}{2}}$ (where $|J_i|$ is the cardinality of the set J_i), the entire uncertain space is covered by the ellipsoid uncertainty set.

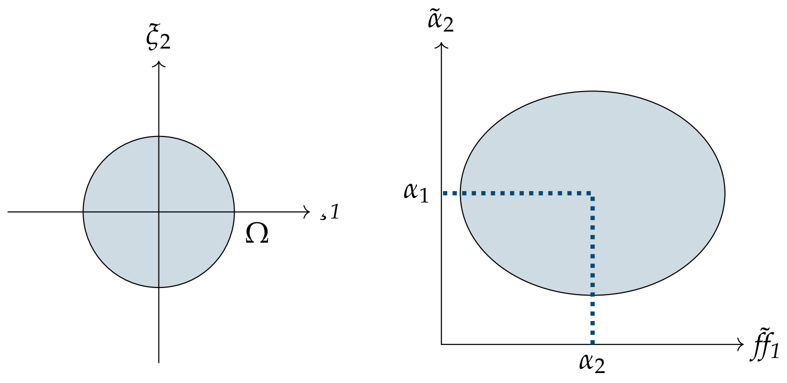


FIGURE B.2: Illustations of ellipsoidal uncertainty set

Definition B.3 (Polyhedral Uncertainty Set)**.** The polyhedral uncertainty is described using the 1-norm of the uncertain data vector as shown in Figure B.3,

$$\mathcal{U}_1 = \{\xi \mid \|\xi\|_1 \leq \Gamma\} = \left\{\xi \mid \sum_{j \in J_i} |\xi_j| \leq \Gamma\right\} \tag{B.9}$$

where Γ is the adjustable parameter controlling the size of the uncertainty set. Note that for the bounded uncertainty $\xi_j \in [-1, 1]$, when $\Gamma \geq |J_i|$, the overall uncertain space is covered by the polyhedral uncertainty set.

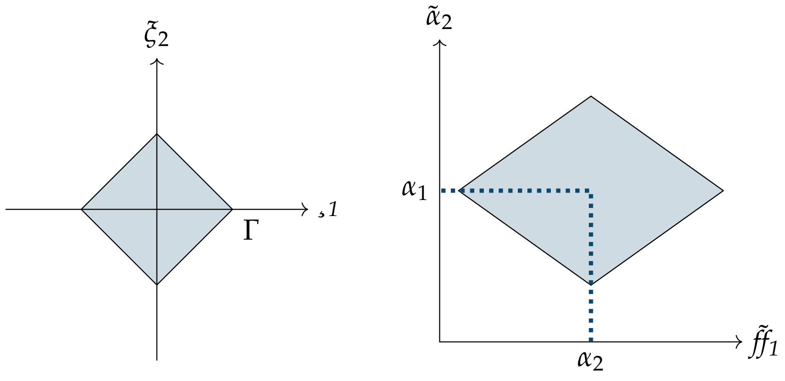


FIGURE B.3: Illustations of polyhedral uncertainty set

Definition B.4 ("Box+ellipsoidal" Uncertainty Set)**.** This type of uncertainty set is the intersection between an ellipsoid and a box defined as follows,

$$\mathcal{U}_{2\cap\infty} = \{\xi \mid \sum_{j \in J_i} \xi_j^2 \leq \Omega^2,\ |\xi| \leq \Psi,\ \forall j \in J_i\} \tag{B.10}$$

In order to avoid the situation where the intersection of the box and ellipsoidal uncertainty set reduces to any one of its components, the parameters must satisfy:

$$\Psi \leq \Omega \leq \Psi\sqrt{|J_i|} \tag{B.11}$$

Definition B.5 ("Box+polyhedral" Uncertainty Set)**.** This type of uncertainty set is the intersection between a polyhedral and an interval set defined with either the 1-norm or ∞-norm as follows,

$$\mathcal{U}_{1\cap\infty} = \{\xi \mid \sum_{j \in J_i} |\xi_j| \leq \Gamma,\ |\xi| \leq \Psi,\ \forall j \in J_i\} \tag{B.12}$$

In order to avoid the situation where the intersection of the interval and polyhedral uncertainty set reduces to any one of its components, the parameters must satisfy:

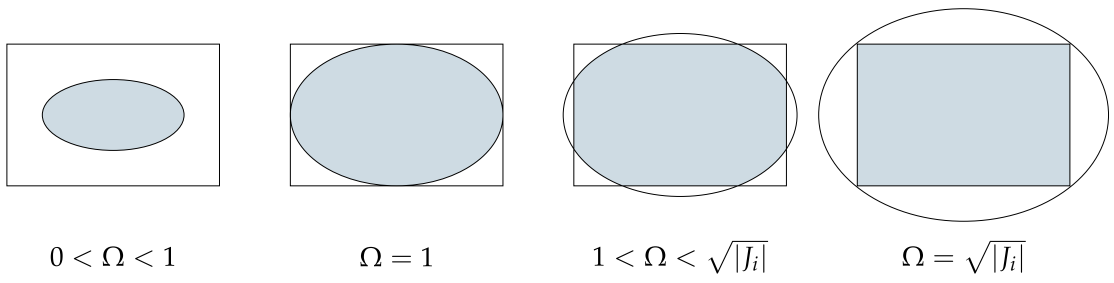
$$\Psi \leq \Omega \leq \Psi|J_i| \tag{B.13}$$

Definition B.6 ("Box+ellipsoidal+polyhedral" Uncertainty Set)**.** This type of uncertainty set is the intersection between the ellipsoid, polyhedral and box sets defined as follows,

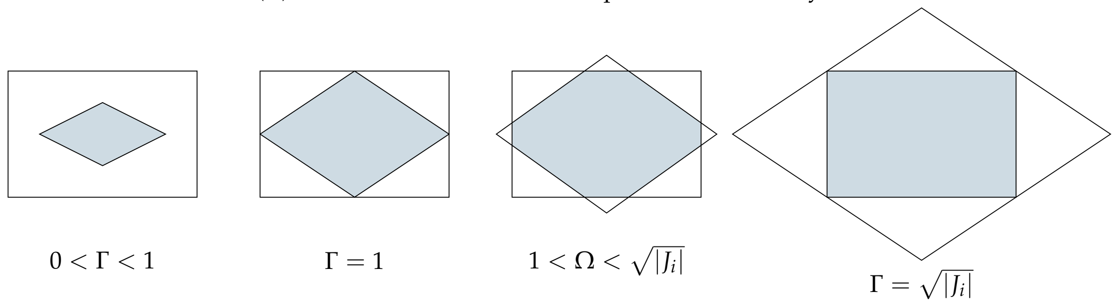
$$\mathcal{U}_{1\cap 2\cap \infty} = \{\xi \mid \sum_{j\,\in\,J_i} |\xi_j| \leq \Gamma,\ \sum_{j\,\in\,J_i} \xi_j^2 \leq \Omega^2,\ |\xi| \leq \Psi,\ \forall j \in J_i\} \tag{B.14}$$

In order to avoid the situation where the intersection of the set reduces to any one of its components, the parameters must satisfy:

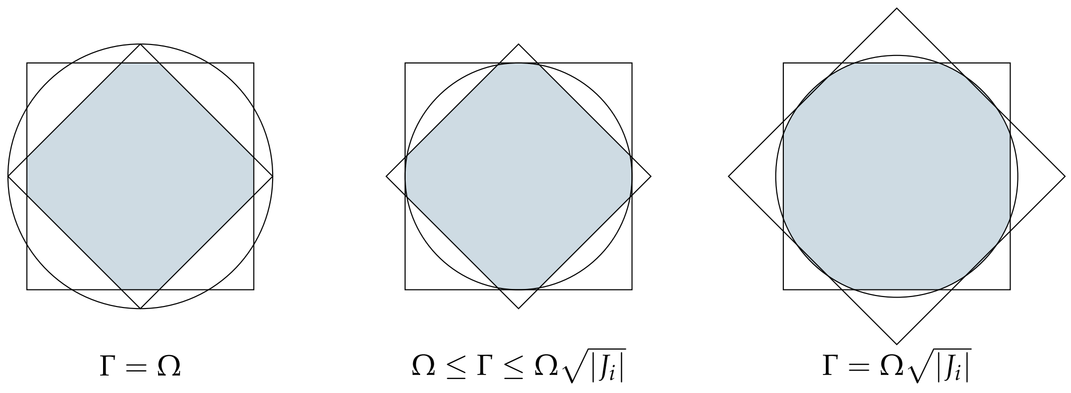
$$\begin{gathered} \Psi \leq \Omega \leq \Psi\sqrt{|J_i|} \\ \Omega \leq \Gamma \leq \Omega\sqrt{|J_i|} \end{gathered} \tag{B.15}$$



(A) Illustration of "Interval + Ellipsoidal" uncertainty set



(B) Illustration of "Interval + Polyhedral" uncertainty set


$\Gamma = \Omega$
$\Omega \leq \Gamma \leq \Omega\sqrt{|J_i|}$
$\Gamma = \Omega\sqrt{|J_i|}$

(C) Illustration of "Interval + Polyhedral + Box" uncertainty set

Appendix C

Booking control Constraint Matrices

C.1 Partitioned booking limits

The capacity constraint matrix for our example is an $m \times n$ matrix where $a_{kj} = 1$ if a product j uses resource k. For the example in Figure 4.3, $m = 3$ and $n = 18$,

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & & \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & & \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & & \end{pmatrix} \tag{C.1}$$

The demand and booking limits constraints have the same $n \times n$ diagonal structure where n is the number of products in offer,

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

C.2 Nested booking limits

There are 6 booking buckets, that use the 3 resources of the network. Each bucket contains three products ordered by Fare Value such that FareClass1 $\geq$ FareClass2 $\geq$ FareClass3 ≥ 0 To represent this relationship between the products in the booking buckets, let there be n columns in a matrix where each column corresponds to a product. Let the first 6 columns be Fare Class 3 products and the last six columns be Fare Class 1 products. Then, the hierarchical order of products is represented by the matrix,

$$\left(\begin{array}{c|c|c}
\begin{matrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{matrix} &
\begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{matrix} &
\begin{matrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 \end{matrix} \\
\hline
\begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{matrix} &
\begin{matrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 \end{matrix} &
\begin{matrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{matrix} \\
\hline
\begin{matrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 \end{matrix} &
\begin{matrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{matrix} &
\begin{matrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{matrix}
\end{array}\right)$$

The capacity matrix of Fare Class 1 products is given by,

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{C.2}$$

References

Daniel Adelman. Dynamic bid prices in revenue management. *Operations Research*, 55 (4):647–661, 2007.

Elodie Adida and Georgia Perakis. Dynamic pricing and inventory control: robust vs. stochastic uncertainty models—a computational study. *Annals of Operations Research*, 181(1):125–157, 2010.

Nur Ayvaz-Cavdaroglu, Dinesh K Gauri, and Scott Webster. Empirical evidence of revenue management in the cruise line industry. *Journal of Travel Research*, 58(1):104–120, 2019.

Michael O Ball and Maurice Queyranne. Toward robust revenue management: Competitive analysis of online booking. *Operations Research*, 57(4):950–963, 2009.

Justin Beck, John Harvey, Kristina Kaylen, Corrado Sala, Melinda Urban, Peter Vermeulen, Norman Wilken, Wei Xie, Dan Iliescu, and Pratik Mital. Carnival optimizes revenue and inventory across heterogenous cruise line brands. *INFORMS Journal on Applied Analytics*, 51(1):26–41, 2021.

Peter Belobaba. Air travel demand and airline seat inventory management. Technical report, Cambridge, MA: Flight Transportation Laboratory, Massachusetts Institute of Technology,[1987], 1987a.

Peter P Belobaba. Survey paper — airline yield management an overview of seat inventory control. *Transportation Science*, 21(2):63–73, 1987b.

Peter P Belobaba. Or practice — application of a probabilistic decision model to airline seat inventory control. *Operations Research*, 37(2):183–197, 1989.

Aharon Ben-Tal and Arkadi Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of uncertain linear programs. *Operations Research letters*, 25(1):1–13, 1999.

Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000.

Aharon Ben-Tal, Arkadi Nemirovski, and Cees Roos. Robust solutions of uncertain quadratic and conic-quadratic problems. *SIAM Journal on Optimization*, 13(2):535–560, 2002.

Dimitris Bertsimas and David B Brown. Constructing uncertainty sets for robust linear optimization. *Operations Research*, 57(6):1483–1495, 2009.

Dimitris Bertsimas and Sanne De Boer. Simulation-based booking limits for airline revenue management. *Operations Research*, 53(1):90–106, 2005.

Dimitris Bertsimas and Ioana Popescu. Revenue management in a dynamic network environment. *Transportation Science*, 37(3):257–277, 2003.

Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1-3):49–71, 2003.

Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52 (1):35–53, 2004.

Dimitris Bertsimas and Aurélie Thiele. A robust optimization approach to inventory theory. *Operations Research*, 54(1):150–168, 2006.

Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and Applications of Robust Optimization. *SIAM Review*, 53(3):464–501, 2011. ISSN 0036-1445. . URL `http://books.google.com/books?hl=en{&}lr={&}id=DttjR7IpjUEC{&}oi=fnd{&}pg=PR9{&}dq=Robust+Optimization{&}ots=W463fBWlR-{&}sig=N1DRhYF4NIE1XxZRwGsmZXy-0gU`.

Neil Biehn. A cruise ship is not a floating hotel. *Journal of Revenue and Pricing Management*, 5(2):135–142, 2006.

S Ilker Birbil, JBG Frenk, Joaquim AS Gromicho, and Shuzhong Zhang. The role of robust optimization in single-leg airline revenue management. *Management Science*, 55(1):148–163, 2009.

Gabriel Bitran and René Caldentey. An overview of pricing models for revenue management. *Manufacturing & Service Operations Management*, 5(3):203–229, 2003.

Shelby L Brumelle and Jeffrey I McGill. Airline seat allocation with multiple nested fare classes. *Operations Research*, 41(1):127–137, 1993.

Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

Wen-Chyuan Chiang, Jason CH Chen, and Xiaojing Xu. An overview of research on revenue management: current issues and future research. *International journal of revenue management*, 1(1):97–128, 2006.

CLIA. The global economic contribution of cruise tourism 2018. Technical report, Cruise Lines International Association, 2019.

William L Cooper. Asymptotic behavior of an allocation policy for revenue management. *Operations Research*, 50(4):720–727, 2002.

Robert G Cross. *Revenue management: Hard-core tactics for market domination*. Crown Business, 1997.

Cruise Market Watch. Financial breakdown of typical cruiser, 2018. URL `https://cruisemarketwatch.com/financial-breakdown-of-typical-cruiser/`.

Renwick E Curry. Optimal airline seat allocation with fare classes nested by origins and destinations. *Transportation Science*, 24(3):193–204, 1990.

James M Davis, Guillermo Gallego, and Huseyin Topaloglu. Assortment optimization under variants of the nested logit model. *Operations Research*, 62(2):250–273, 2014.

Sanne V de Boer, Richard Freling, and Nanda Piersma. Mathematical programming for network revenue management revisited. *European Journal of Operational Research*, 137 (1):72–92, 2002.

Laurent El Ghaoui and Hervé Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

Laurent El Ghaoui, Francois Oustry, and Hervé Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9(1):33–52, 1998.

Virginie Gabrel, Cécile Murat, and Aurélie Thiele. Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3):471–483, 2014.

G. Gallego, G. Iyengar, R. Phillips, and A. Dubey. Managing flexible products on a network, 2004.

Guillermo Gallego and Garrett Van Ryzin. A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research*, 45(1):24–41, 1997.

Monique Giese. Covid-19 impacts on global cruise industry. Blog post, July 2020. URL `https://home.kpmg/xx/en/blogs/home/posts/2020/07/covid-19-impacts-on-global-cruise-industry.html`.

Fred Glover, Randy Glover, Joe Lorenzo, and Claude McMillan. The passenger-mix problem in the scheduled airlines. *Interfaces*, 12(3):73–80, 1982.

Jochen Gonsch. A survey on risk-averse and robust revenue management. *European Journal of Operational Research*, 263(2):337 – 348, 2017. ISSN 0377-2217. . URL `http://www.sciencedirect.com/science/article/pii/S0377221717304800`.

John J Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, 16(1):122–128, 1986.

John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. 1992.

Lu Ji and Joseph Mazzarella. Application of modified nested and dynamic class allocation models for cruise line revenue management. *Journal of Revenue and Pricing Management*, 6(1):19–32, 2007.

Sheryl E Kimes. Yield management: a tool for capacity-considered service firms. *Journal of operations management*, 8(4):348–363, 1989.

Robert Klein, Sebastian Koch, Claudius Steinhardt, and Arne K Strauss. A review of revenue management: Recent generalizations and advances in industry applications. *European Journal of Operational Research*, 2019.

Oliver Kramer. Genetic algorithms. In *Genetic algorithm essentials*, pages 11–19. Springer, 2017.

Sumit Kunnumkal and Kalyan Talluri. A note on relaxations of the choice network revenue management dynamic program. *Operations Research*, 64(1):158–166, 2016.

Yingjie Lan, Huina Gao, Michael O Ball, and Itir Karaesmen. Revenue management with limited demand information. *Management Science*, 54(9):1594–1609, 2008.

Tak C Lee and Marvin Hersh. A model for dynamic airline seat inventory control with multiple seat bookings. *Transportation Science*, 27(3):252–265, 1993.

Bingzhou Li. A cruise line dynamic overbooking model with multiple cabin types from the view of real options. *Cornell Hospitality Quarterly*, 55(2):197–209, 2014.

Zukui Li, Ran Ding, and Christodoulos A Floudas. A comparative theoretical and computational study on robust counterpart optimization: I. robust linear optimization and robust mixed integer linear optimization. *Industrial & engineering chemistry research*, 50(18):10567–10603, 2011.

Ken Littlewood. Special issue papers: Forecasting and control of passenger bookings. *Journal of Revenue and Pricing Management*, 4(2):111–123, 2005.

Qian Liu and Garrett Van Ryzin. On the choice-based linear programming model for network revenue management. *Manufacturing & Service Operations Management*, 10 (2):288–310, 2008.

Bacel Maddah, Lama Moussawi-Haidar, Muhammad El-Taha, and Hussein Rida. Dynamic cruise ship revenue management. *European Journal of Operational Research*, 207 (1):445–455, 2010.

Jeffrey I McGill and Garrett J Van Ryzin. Revenue management: Research overview and prospects. *Transportation Science*, 33(2):233–256, 1999.

Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

Karthik Natarajan, Dessislava Pachamanova, and Melvyn Sim. Constructing risk measures from uncertainty sets. *Operations research*, 57(5):1129–1141, 2009.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

Georgia Perakis and Guillaume Roels. Robust controls for network revenue management. *Manufacturing & Service Operations Management*, 12(1):56–76, 2010.

Georgia Perakis and Anshul Sood. Competitive multi-period pricing for perishable products: A robust optimization approach. *Mathematical Programming*, 107(1-2):295–335, 2006.

Lawrence W Robinson. Optimal and approximate control policies for airline booking with sequential nonmonotonic fare classes. *Operations Research*, 43(2):252–263, 1995.

Marvin Rothstein. An airline overbooking model. *Transportation Science*, 5(2):180–192, 1971.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Paat Rusmevichientong and Huseyin Topaloglu. Robust assortment optimization in revenue management under the multinomial logit choice model. *Operations Research*, 60(4):865–882, 2012.

Leonard J Savage. The theory of statistical decision. *Journal of the American Statistical association*, 46(253):55–67, 1951.

Robert Warren Simpson. *Using network flow techniques to find shadow prices for market demands and seat inventory control*. MIT, Department of Aeronautics and Astronautics, Flight Transportation Laboratory, 1989.

SN Sivanandam and SN Deepa. Genetic algorithms. In *Introduction to genetic algorithms*, pages 15–37. Springer, 2008.

Allen L Soyster. Technical note — convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

Arne K Strauss, Robert Klein, and Claudius Steinhardt. A review of choice-based revenue management: Theory and methods. *European Journal of Operational Research*, 271 (2):375–387, 2018.

Daniel Sturm and Kathrin Fischer. Cruise line revenue management: Overview and research opportunities. In *Operations Research Proceedings 2016*, pages 441–447. Springer, 2018.

Kalyan T Talluri and Garrett J Van Ryzin. An analysis of bid-price controls for network revenue management. *Management Science*, 44(11-part-1):1577–1593, 1998.

Kalyan T Talluri and Garrett J Van Ryzin. A randomized linear programming method for computing network bid prices. *Transportation Science*, 33(2):207–216, 1999.

Kalyan T Talluri and Garrett J Van Ryzin. Revenue management under a general discrete choice model of consumer behavior. *Management Science*, 50(1):15–33, 2004.

Kalyan T Talluri and Garrett J Van Ryzin. An introduction to revenue management. *Tutorials in operations research*, pages 142–195, 2005.

Kalyan T Talluri and Garrett J Van Ryzin. *The theory and practice of revenue management*, volume 68. Springer Science & Business Media, 2006.

Aurélie Thiele. *A robust optimization approach to supply chains and revenue management*. PhD thesis, Massachusetts Institute of Technology, 2004.

Rex S Toh, Mary J Rivers, and Teresa W Ling. Room occupancies: cruise lines out-do the hotels. *International Journal of Hospitality Management*, 24(1):121–135, 2005.

Garrett Van Ryzin and Gustavo Vulcano. Computing virtual nesting controls for network revenue management under customer choice behavior. *Manufacturing & Service Operations Management*, 10(3):448–467, 2008.

John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton university press, 1944.

Abraham Wald. Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, pages 265–280, 1945.

Abraham Wald. *Statistical decision functions*. Wiley, New York, 1950.

Lawrence R Weatherford, Samuel E Bodily, and Phillip E Pfeifer. Modeling the customer arrival process and comparing decision rules in perishable asset revenue management situations. *Transportation Science*, 27(3):239–251, 1993.

Elizabeth Louise Williamson. Comparison of optimization techniques for origin-destination seat inventory control. Technical report, [Cambridge, Mass.: Massachusetts Institute of Technology], Flight Transportation Laboratory,[1988], 1988.

Elizabeth Louise Williamson. *Airline network seat inventory control: Methodologies and revenue impacts*. PhD thesis, Massachusetts Institute of Technology, 1992.

RD Wollmer. A hub-spoke seat management model. *Unpublished Internal Report, Mc Donnell Douglas Corporation, Long Beach, CA*, 1986.

Richard D Wollmer. An airline seat management model for a single leg route when lower fare classes book first. *Operations Research*, 40(1):26–37, 1992.

Dan Zhang and Daniel Adelman. An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science*, 43 (3):381–394, 2009. ISSN 00411655, 15265447. URL http://www.jstor.org/stable/25769459.