Airline Yield Management in a Dynamic Network Environment

by

Zhihang Chi

B.S., Peking University (June 1985)
S.M., Peking University (January 1988)
S.M., Massachusetts Institute of Technology (June 1991)

Submitted to the Operations Research Center
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1995

© Massachusetts Institute of Technology 1995. All rights reserved.

Author

Operations Research Center
September 30, 1994

Certified by

Dimitris Bertsimas
Associate Professor
Thesis Supervisor

Accepted by

Robert Freund
Acting Director, Operations Research Center

Massachusetts Institute of Technology
ARCHIVES

NOV 29 1994
Airline Yield Management in a Dynamic Network

Environment

by

Zhihang Chi

Submitted to the Operations Research Center
on September 30, 1994, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In this dissertation we study the airline yield management problem, with emphasis on its dynamic and network aspects. We propose models and algorithms for the static and dynamic single-leg cases, and the static and dynamic network cases. In each of these cases, we separately address models with and without cancelations and provide strong support for the models and algorithms with extensive computational results.

In the area of single-leg static models, we propose a new optimal exact algorithm (the wait-and-see approach) to compute optimal booking limits. We compare computationally the wait-and-see approach with a current industry practice and find that it consistently outperforms the industry practice, in terms of both total revenue and load factor. We further develop a single-leg static model which directly incorporates cancelations and overbookings. We propose an approximate dynamic programming algorithm, which, in simulation experiments, provides provably near-optimal solutions.

We next propose a hierarchical approach for network static models. At the higher level, we use a concave network flow algorithm to allocate flight capacities among all itineraries in a flight network. At the lower level, we use the wait-and-see approach to determine the booking limits within each individual itinerary, given the capacity allocated by the concave network flow algorithm. We adopt a fast implementation for the concave network flow algorithm and conduct computational experiments that compare the hierarchical approach with the current industry practice, virtual nesting. We find that the hierarchical approach yields comparable results to the virtual nesting approach, but better adapts its behavior to changes in system parameters.

For dynamic models, we develop a single-leg dynamic model using dynamic programming. For cases without cancelations and overbookings, we show that the optimal solution is characterized by the notion of a "threshold time", which leads to a very efficient implementation. For cases with cancelations and overbookings, we con-
jecture that "threshold intervals" exist. Compared with optimal static solutions our proposed algorithm produces significant improvement. Compared with optimal static solutions that are periodically re-optimized, the improvement is not as significant, but still non-negligible.

Finally we develop an optimal algorithm for the dynamic network case and compare our approach to the current industry practice, virtual nesting. Even with re-optimization of the virtual nesting approach, the dynamic network algorithm produces significant benefits, in terms of both total revenue and load factor, a fact that underscores the importance of dynamic network models. We further propose an approximation approach to tackle the high dimensionality of dynamic network models by approximating the "cost-to-go" function with a quadratic form. Our preliminary computational results indicate a close agreement between the approximate and exact solutions.

Thesis Supervisor: Dimitris Bertsimas
Title: Associate Professor
Acknowledgments

My sincerest thanks go to my advisor Professor Dimitris Bertsimas for his relentless efforts in advising and supporting me during the course of this research. I am greatly inspired by his work ethic and high standards. I benefited a great deal from his breadth of knowledge and deep thinking. In addition, the friendship and personal interaction he offered have made my years at MIT much more enjoyable.

I would also like to thank the members of my committee, Professor Amedeo Odoni, for providing expert insight and suggestions as well as heartening encouragement, and Professor Robert Freund, for making valuable comments on the dissertation and for being a friend almost since my very first day at MIT.

I am extremely proud and privileged to be a member of the Operations Research Center (ORC) at MIT. It features superb facilities and the friendliest environment I have ever seen at any school. I feel particularly indebted to its administrative staff, especially Paulette Mosley, Laura Rose and Cheryl Mottley, for their generous help and extraordinary kindness.

Many thanks to my fellow students at the ORC, especially fellow old-timers such as Sungsu Ahn, Armann Ingolfsson, Pieter Klaassen, Kerry Malone, Joe Milner and S. Raghavan, for their general friendship and care. I would like to compliment my fellow system administrators, David Markowitz and Rob Shumsky on a wonderful job in maintaining the Sun workstations, on which my research was heavily dependent upon.

I am most grateful to my parents for the good health they gave me and for their high expectations and unconditional love that have kept me carrying on.

My gifted younger brother Leo Z. Chi has helped proofreading the dissertation and making valuable comments. He deserves special mention.

Last but not least, I owe my deepest gratitude to my wife Li Chen, for always being there, especially when skies were gray.
## Contents

1 Introduction .................................................. 12
   1.1 Characteristics of Airline Yield Management ................. 13
   1.2 Objectives and Contributions of this Dissertation .......... 15
   1.3 Literature Review ........................................ 17
   1.4 Structure of this Dissertation .............................. 19

2 A New Single-Leg Static Model ................................. 21
   2.1 Model and Notation ....................................... 22
   2.2 A Static Model without Cancelations and Overbookings ....... 22
       2.2.1 The Algorithm .................................... 23
       2.2.2 Running Time ...................................... 24
       2.2.3 Properties of the Revenue Function $R(x)$ ............. 24
       2.2.4 Wait-and-See Strategy versus an Optimal A Priori Strategy 34
   2.3 A Single-Leg Static Model with Cancelations and Overbookings .. 41
       2.3.1 Notation .......................................... 41
       2.3.2 Boundary Condition – Overbooking Penalty ............... 42
       2.3.3 Dynamic Programming Recursion ....................... 43
       2.3.4 Approximation of Overbooking Penalty .................. 43
       2.3.5 Approximated Expected Revenue Function $\tilde{R}(\cdot)$ .... 44
       2.3.6 Concave Approximation of $G(S)$ ...................... 47
3 Single-Leg Dynamic Models 52
3.1 Introduction ........................................ 52
3.2 A Single-Leg Dynamic Model ....................... 53
3.3 Threshold Times ...................................... 54
   3.3.1 Properties of the Revenue Function and the Threshold Times 55
   3.3.2 Algorithm for Computing Threshold Times .................. 59
3.4 A Dynamic Model with Cancelations and Overbookings .......... 60
3.5 An Alternative Dynamic Model with Cancelations and Overbookings 66
   3.5.1 An Approximation ................................ 66
   3.5.2 Algorithm for Computing Threshold Times ................. 70
   3.5.3 Computational Complexity .......................... 71

4 Static Network Models 72
4.1 Introduction ........................................ 72
4.2 A Network Model for Stochastic Demands .................... 77
   4.2.1 Revenue Function $R_{ij}(\cdot)$ ....................... 80
   4.2.2 A Hierarchical Algorithm .......................... 80
   4.2.3 Computational Issues .............................. 83

5 Dynamic Network Models 84
5.1 Introduction ........................................ 84
5.2 A Hierarchical Dynamic Model ........................ 84
5.3 A Global Dynamic Network Model for Two Connecting Flights .... 86
   5.3.1 Notation .......................................... 86
   5.3.2 Recursion and Boundary Conditions ................... 87
5.4 A Global Dynamic Model for the Hub-and-Spoke Network .......... 89
   5.4.1 Notation for the General Problem ................... 89
   5.4.2 Generic Recursion ............................... 90
5.4.3 A Global Dynamic Model without Cancelations and Overbookings ........................................ 92
5.5 An Approximation of Dynamic Models ................................. 94
5.5.1 Solving for $\alpha_t, \beta_t$ and $A_t$ ............................... 94

6 Simulation and Computational Results ................................. 96
  6.1 Wait-and-see versus EMSR ........................................ 98
  6.2 Static Models with Cancelations ................................ 103
  6.3 Single-Leg Dynamic Models versus Static Models ............... 105
  6.4 Divide-and-Nest versus Virtual Nesting .......................... 114
  6.5 Dynamic Network Model versus Virtual Nesting ................. 120
  6.6 Quadratic Approximation of Dynamic Models .................... 131

7 Conclusions .................................................................. 136
  7.1 Contributions of this Dissertation ................................. 136
  7.2 Future Research Directions ....................................... 138

A Proofs of Theorems and Propositions .................................. 140
  A.1 Proof of Theorem 2.5 ............................................. 140
  A.2 Proof of Theorem 2.15 ............................................ 144
  A.3 Proof of Theorem 2.16 ............................................ 150
  A.4 Proof of Proposition 3.3 ......................................... 155
  A.5 Proof of Proposition 3.4 ......................................... 158
# List of Figures

1-1 Illustration of the static assumption: $T$ is the entire booking horizon.
There are $n$ fare classes, the higher the index the lower the class.

1-2 Illustration of nested booking limits: $B_i$ is the booking limit for class
$i, i = 1, 2, \ldots, n$, where $n$ is the total number of classes. The higher
the index, the lower the class. $C$ is the flight capacity.

1-3 Illustration of network effects: 3 itineraries share the common Boston-
Chicago leg. The 3 itineraries are Bangor to Chicago via Boston,
Boston to Chicago, and Boston to Los Angeles via Chicago.

3-1 Illustration of the threshold time

3-2 Possible scenarios of the threshold window

4-1 A hub-and-spoke system

4-2 A simple virtual nesting example

4-3 Creation of virtual fare buckets

4-4 Transformation of the hub-and-spoke system into a transportation net-
work

4-5 Different stages of the hierarchical algorithm

5-1 A simple network with 3 itineraries and 2 flight legs

5-2 General hub-and-spoke network

6-1 A simple virtual nesting example
List of Tables

1.1 Previous work and planned work in this research .................. 20
3.1 A counter-example: 2 classes, 30 capacity, $300 penalty ........... 63
3.2 Decisions on Class 1 ............................................. 63
6.1 An Example: 5 fare classes, capacity = 195 ..................... 98
6.2 Results for the example ........................................ 98
6.3 Simulation Statistics .............................................. 99
6.4 Variation 1: 5 fare classes, capacity = 195 ..................... 100
6.5 Results for Variation 1 ........................................... 100
6.6 Simulation Statistics: Increased Variability ..................... 101
6.7 Variation 2: 5 fare classes, capacity = 235 ..................... 101
6.8 Results for Variation 2 ........................................... 102
6.9 Simulation Statistics: More Capacity ................................ 102
6.10 Example: 4 classes, capacity = 200, penalty = $600 .............. 104
6.11 Example: 5 classes, capacity = 280, penalty = $1250 .............. 105
6.12 Example: 4 classes, horizon = 1000, capacity = 150 .............. 107
6.13 Results for the example in Table 6.12 ......................... 107
6.14 Simulation Statistics for the example in Table 6.12 .............. 108
6.15 Results for the example in Table 6.12 with the wait-and-see approach applied ........................................... 109
6.16 Simulation statistics for the example in Table 6.12 with the wait-and-see approach applied .................................. 109
6.17 Re-optimization of the example in Table 6.12 .................................. 111
6.17 Re-optimization of the example in Table 6.12 (continued) .............. 112
6.17 Re-optimization of the example in Table 6.12 (continued) .............. 113
6.18 Definition of virtual fare buckets .................................. 116
6.19 Case 1: capacity = 100 .................................. 116
6.20 Virtual nesting booking limits for Case 1 – Revenue = 48092.33 ....... 116
6.21 Results of Case 1 using divide-and-nest – Revenue = 48139.62 ..... 116
6.22 Virtual nesting booking limits for Case 2 – Revenue = 41274.53 ..... 117
6.23 Results of Case 2 using divide-and-nest – Revenue = 41367.17 ..... 117
6.24 Virtual nesting booking limits for Case 3 – Revenue = 54051.04 ..... 117
6.25 Results of Case 3 using divide-and-nest – Revenue = 53838.32 ..... 118
6.26 Case 4: capacity = 120 .................................. 118
6.27 Virtual nesting booking limits for Case 4 – Revenue = 54428.18 ..... 118
6.28 Results of Case 4 using divide-and-nest – Revenue = 54269.47 ..... 119
6.29 Case 5: capacity = 120 .................................. 119
6.30 Virtual nesting booking limits for Case 5 – Revenue = 55029.38 ..... 119
6.31 Results of Case 5 using divide-and-nest – Revenue = 55260.92 ..... 119
6.32 Example 1: horizon = 1500, 9 ODFs, capacity = 120 ............... 121
6.33 Seats requested and accepted by ODF ................................ 122
6.34 Seats requested and accepted by legs ................................ 122
6.35 Revenue, standard deviation and total number of passengers carried . 122
6.36 Definition of virtual fare classes .................................. 123
6.37 Seats requested and accepted by ODF ................................ 124
6.38 Seats requested and accepted by legs ................................ 124
6.39 Revenue, standard deviation and total number of passengers carried . 125
6.40 Summary of re-optimization results .................................. 126
6.40 Summary of re-optimization results (continued) .................. 127
6.41 Seats requested and accepted by ODF, update interval = 23 ........ 128
6.42 Seats requested and accepted by legs, update interval = 23 ........ 128
6.43 Example 2: horizon = 1500, 9 ODFs, capacity = 80 ............. 128
6.44 Example 2: Seats requested and accepted by ODF ............... 129
6.45 Seats requested and accepted by legs ............................. 129
6.46 Revenue, standard deviation and total number of passengers carried . 129
6.47 Definition of Virtual Fare Classes for Example 2 .................. 129
6.48 Example 2: Seats requested and accepted by ODF ............... 130
6.49 Example 2: Seats requested and accepted by legs ............... 130
6.50 Revenue, standard deviation and total number of passengers carried . 130
6.51 Summary of re-optimization results ............................. 132
6.51 Summary of re-optimization results (continued) .................. 134
6.52 Seats requested and accepted by ODF, update interval = 1 ........ 134
6.53 Seats requested and accepted by legs, update interval = 1 ........ 134
6.54 A 3-class example: horizon = 100, capacity = 80 ............... 135
6.55 Comparison of exact and approximated revenues for (0, 0, 0, t): number of data points is 100 ................................. 135
Chapter 1

Introduction

Two decades ago the airline industry was heavily regulated by government. Entry into the industry was tightly controlled and as a result, there were very few airlines. Competition was at a relatively low level as fares were set by the Civil Aeronautics Board (CAB) based on industry average costs. The airline deregulation act in 1978 resulted in a sharp increase in both the air travel supply and the number of operating airlines. In order to boost demand for air travel to absorb the increased supply, airlines began to offer discount fares to attract more passengers. Today we see a very diverse air fare structure, ranging from full fare to discount fares in different degrees. It is commonplace to find passengers on board the same flight sitting next to each other, some paying $300, or $500 and others paying $800.

A critical issue airlines need to address is how to optimally control seat inventory. On one hand, it is important that airlines improve their market share and so there is an incentive to fill as many seats as possible. On the other hand, revenue consideration tends to prevent seats from being sold at discount fares and “protect” them for potential full-fare passengers. The problem of maximizing revenues is called the yield management (YM) problem.

Yield management has proved vital to modern airlines. Controlling the mix of fare classes can translate into revenue increases of $200 to $500 million for carriers with
Chapter 1. Introduction

total revenues of $1 to $5 billion [7]. When assessing the role of yield management at
prize-winning paper that, “American Airlines estimates the quantifiable benefit at
$1.4 billion over the last three years and expects an annual revenue contribution of
over $500 million to continue into the future.”

1.1 Characteristics of Airline Yield Management

In this section we describe the fundamental characteristics of airline yield management
that motivate the issues we study in this dissertation. In the following discussion,
unless otherwise stated, fare classes are ranked solely by fares. The higher the fare,
the higher the class.

1. Stochastic Demands: Demands for seats demonstrate significant random be-
haviors. As a result, the yield management problem is inherently stochastic.

2. Arrival Patterns of Demands: Demands exhibit certain arrival pattern. For
example, advance purchase requirement on discount fares can generate more
discount-fare reservations than full-fare reservations in the early stages of the
booking process. Arrival patterns open up numerous modeling possibilities.
At one extreme, one can make the assumption that bookings follow a strict
sequence – the lower the fare class, the earlier its booking occurs, and a class will
not start booking until all lower classes have finished booking. This is referred to
as the “static assumption”. At another extreme, arrivals are completely mixed.
This is referred to as the “dynamic assumption”.

3. Network Effects: Airlines operate over large-scale flight networks (typically
the hub-and-spoke system). A passenger itinerary typically consists of multiple
physical flight legs. Aboard a flight, competition for seats not only exists among
different fare classes, but also among different itineraries. In this sense network
effects constitute an important dimension of the yield management problem.

4. **Cancelations and Overbookings**: Normally reservations can be cancelled with no penalty. For certain fare classes, even purchased tickets can be cancelled (fully or partially refunded). Just as demands, cancelations also exhibit significant random behaviors. To compensate for cancelations, airlines usually overbook their flights. The question is how to optimally overbook the flights. If a flight is overbooked too conservatively, it is likely to suffer from cancelations. If it is overbooked too aggressively, it is likely to incur a penalty cost for denied boardings.

In order to address some of these characteristics, several models can be proposed. They can be categorized as follows:

1. **Static vs. Dynamic Models**

   Under the static assumption, lower fare classes book up before higher fare classes. This assumption implies that reservations for different classes happen sequentially. See Figure 1-1 for an illustration of the static assumption. Under the static assumption, we need to control the number of seats that a lower class can occupy so that sufficient seats can be saved for higher but late-arriving classes. The optimal policy is to impose "nested" booking limits on the fare classes. Nested booking limits are illustrated in Figure 1-2. A "nested" booking limit on a fare class is the maximum number of seats allowed for this class and all the classes lower than this class. In addition, any unused booking limit will be accrued to the booking limit for the next higher fare class. In essence, a nesting approach will try to protect seats for higher fare classes. A nesting approach implemented in seat inventory control will allow higher classes to take seats that are supposedly reserved for a lower class.
Figure 1-1: Illustration of the static assumption: $T$ is the entire booking horizon. There are $n$ fare classes, the higher the index the lower the class.

In a dynamic setting, reservations of different classes occur dynamically in time according to some stochastic process.

2. Single-leg vs. Network Models

In a single-leg model, all passengers have the same itinerary, i.e., they all travel from a single origin to a single destination. In a network model, passengers aboard a flight can have different itineraries, i.e., they have different origins and/or different destinations, which may or may not be served by the flight they are aboard. Consider a flight from Boston to Chicago. In a single-leg scenario, all passengers aboard the flight originate from Boston and end in Chicago. In a network scenario, aboard the same flight in addition to passengers who originate from Boston and end in Chicago there are passengers who did not originate from Boston and/or will not end in Chicago, such as those who originate in Bangor, Maine and/or end in Los Angeles. Figure 1-3 illustrates the network scenario.

### 1.2 Objectives and Contributions of this Dissertation

This dissertation aims to conduct a comprehensive study of the airline yield management problem, with an emphasis on its dynamic and network aspects. In particular,
Figure 1-2: Illustration of nested booking limits: $B_i$ is the booking limit for class $i$, $i = 1, 2, \ldots, n$, where $n$ is the total number of classes. The higher the index, the lower the class. $C$ is the flight capacity.

Figure 1-3: Illustration of network effects: 3 itineraries share the common Boston-Chicago leg. The 3 itineraries are Bangor to Chicago via Boston, Boston to Chicago, and Boston to Los Angeles via Chicago.

the primary objectives are as follows:

1. Develop a conceptually better single-leg static model, that can capture cancellations and overbookings in a direct way.

2. Address the dynamic nature of air travel demands; construct and analyze dynamic models for dynamic demands in a single-leg environment.

3. Address network effect on airline yield management in a static setting; develop static models that fully capture the network effect.

4. Address network effects on airline yield management in a dynamic setting; develop models that handle demands in a dynamic environment.
5. Compare and contrast the above models with current industry practices.

1.3 Literature Review

Littlewood [9] introduced the very first single-leg static model for the case where there are two fare classes with no cancelations or overbookings. The model determines the booking limit on the lower fare class. Additional work was done along the same line by Bhatia and Parekh [4] and Richter [11]. In the two-class setting, demands for the two classes are assumed to be independent of each other and the lower class always books up before the higher class. Under these assumptions, the optimal policy is to set a limit on the number of reservations that can be made for the lower class, or alternatively, a protection level for the higher class against the lower class, which is the number of seats reserved for the higher class. The optimal limit is determined using the following rule: keep accepting requests for a lower fare seat until the expectation of selling a seat to a higher class customer exceeds the lower fare.

Based on Littlewood’s model, Simpson [12] and Belobaba [2] developed a heuristic for multiple fare classes, often referred to as the expected marginal seat revenue (EMSR) model. To calculate the booking limit on a particular fare class, the protection levels of all the higher classes against the class are computed separately, using the 2-class model. The booking limit is capacity minus the sum of the protection levels of all the higher classes. The EMSR model is widely used in the airline industry.

McGill [10] extended Littlewood’s model to multi-fare classes, established sufficient optimality conditions, and proposed for a special case an algorithm that computes optimal booking limits as follows. Consider a case where there are \( n \) fare classes. Let \( X_i \) be the random variable that represents the demand for class \( i, i = 1, \ldots, n \). The input data include:

1. Fares of all \( n \) classes, \( f_1 > f_2 > \cdots > f_n \).

2. Flight capacity \( C \).
3. Probability distributions for $X_i$, $i = 1, \ldots, n$.

The decision variables are $B_i$, the booking limit on class $i$, $i = 1, \ldots, n$. For the special case where the solutions to the following simultaneous equations, $B_i^*$

$$\Pr[X_1 > C - B_2] = \frac{f_2}{f_1}$$
$$\Pr[X_1 > C - B_2 \cap X_1 + X_2 > C - B_3] = \frac{f_3}{f_1}$$

\ldots \ldots
$$\Pr[X_1 > C - B_2 \cap \ldots \cap X_1 + \ldots + X_{n-1} > C - B_n] = \frac{f_n}{f_1}$$

satisfy $0 \leq B_i^* \leq C$, $B_i^*$’s are the optimal booking limits, $i = 1, \ldots, n$, with $B_i^* = C$.

While the above equations constitute a set of sufficient optimality conditions, the underlying algorithm for calculating booking limits might be involved due to the need for computing convolutions and joint probability distributions. For this reason, we propose in Chapter 2 a recursive $O(nC \log C)$ algorithm to calculate optimal booking limits for all classes.

For network models, Buhr [5] considered a two-leg, one-class network (thus 3 itineraries). The decision is how to optimally assign capacity to the 3 itineraries in a non-nesting way. This was extended by Wang [15] to the multi-class case. Glover et al. [8] proposed a deterministic mathematical programming model for general networks and multi-fare classes. Curry [6] introduced a probabilistic mathematical programming model for general networks and multi-fare classes that partitions network capacities and computes the booking limits for fare classes nested based on the legs.

A popular network approach adopted by the airline industry is “virtual nesting”. In virtual nesting, virtual “fare buckets”, which can be seen as aggregated fare classes, are created and assigned weighted values. Classes of all itineraries aboard a flight leg are placed into different fare buckets according to the value of their itinerary. Booking limits are then determined for the virtual fare buckets. Seat inventory control is carried out at the fare bucket level rather than at the individual class level. As
an example, consider a passenger who travels certain fare class from Boston to Los Angeles connecting at Chicago. The fare class, based on the fare of the itinerary, i.e., from Boston to Los Angeles, is mapped to a virtual fare bucket on the Boston-Chicago leg and the Chicago-Los Angeles leg, respectively. Virtual booking limits are checked respectively for the virtual bucket on the Boston-Chicago leg and the Chicago-Los Angeles leg. The passenger is accommodated only if neither of the booking limits are violated. While it captures the network effect to some extent, the "virtual nesting" approach is primarily a leg-based approach.

A recent development in network models is the idea of the bid price [17]. In a bid price setting, demands are seen as bids for seats aboard a flight. There is a bid price associated with each flight. If a passenger is willing to pay at least the bid price, he or she will be accommodated on the flight. Otherwise, he or she is denied seating. From a modeling point of view, bid price models take any of the probabilistic mathematical programming formulations. The bid prices are simply the optimal dual solutions.

1.4 Structure of this Dissertation

In Chapter 2, we develop a new single-leg static model and establish relevant properties. It is shown that the model will lead to optimal solutions. The model is then extended to directly incorporate cancellations and overbookings. An approximation scheme is proposed that addresses the computational difficulties caused by cancellations and overbookings.

In Chapter 3, we study dynamic demands in a single-leg setting. We develop a dynamic model in which fare requests are modeled as stochastic arrivals and the decision to accept or reject a fare request is made based on the capacity remaining at the time of the request, and more importantly on the time of the request itself. We introduce and establish an idea called "threshold time" when no cancellations are allowed. It greatly simplifies the problem and reduces the amount of computation
needed. Finally, the dynamic model is extended to the case with cancelations and overbookings.

In Chapter 4, we develop static network models. We begin by reviewing in some detail the virtual nesting and bid price approaches. We propose a hierarchical approach for the multi-class, static network case. Finally, we transform the network model into a convex flow problem and mention a pseudo-polynomial time algorithm that can be used to solve the problem.

In Chapter 5, we extend the dynamic model developed in Chapter 3 to the network case. We discuss two different approaches. The first one is a global dynamic approach. The second one combines network flow models and the single-leg fully dynamic models.

In Chapter 6, we report computational results on the models we develop in previous chapters. We place particular emphasis on comparing our approaches to current industry practices and use simulation experiments for the comparison.

The following table gives a partial summary of previous work and a structural illustration of this dissertation.

<table>
<thead>
<tr>
<th>Static</th>
<th>Single-Leg</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Two-Class Model (Littlewood [9])</td>
<td>Virtual Nesting</td>
</tr>
<tr>
<td></td>
<td>Exact Solutions (McGill [10])</td>
<td>Bid Price [17]</td>
</tr>
<tr>
<td></td>
<td>EMSR Model (Belobaba [3])</td>
<td>Network Partition [6]</td>
</tr>
<tr>
<td></td>
<td>Chapter 2</td>
<td>Chapter 4</td>
</tr>
</tbody>
</table>

| Dynamic | Chapter 3 | Chapter 5 |
Chapter 2

A New Single-Leg Static Model

In this chapter, we introduce a new approach, based on dynamic programming, for the static multi-class single-leg problem. The new approach is first developed without cancelations and overbookings. The approach is then generalized to the case with cancelations and overbookings.

Our approach is methodologically an improvement over existing models because it utilizes more fully the static assumption. Existing static models determine booking limits on all classes simultaneously. But because the static assumption implies that the fare classes book up one after another, it is not necessary to determine all the booking limits simultaneously. Instead at any point in time we only need to consider the class that is currently open (all higher classes will not begin booking until the current class closes). After the class is closed, we can use the information accumulated during the booking process to determine the booking limit on the next class.

Because our approach will wait till the last minute to determine the booking limits and collect and use previous information, we call it the “wait-and-see” approach.
2.1 Model and Notation

The following notation is followed in this chapter. Additional notation will be defined as needed. We define:

- $C$: flight capacity;
- $f_i$: fare of class $i$, $i = 1, \ldots, n$, and $f_n \leq f_{n-1} \leq \ldots \leq f_1$ where $i = 1, \ldots, n$;
- $p_i(x)$: distribution of demand for class $i$, $i = 1, \ldots, n$;
- $R_i(x)$: optimal expected revenue from classes $1, 2, \ldots, i$ when exactly $x$ seats are available to them;
- $S_i$: booking limit for class $i$, where $i = 1, \ldots, n$. These are the decision variables.

2.2 A Static Model without Cancelations and Overbookings

In this section, we consider the case where there are no cancelations and overbookings. The more complicated case where cancelations and overbookings are allowed will be discussed in the next section.

Given the flight capacity, fares and distribution of demands, we derive the booking limit, i.e., the maximum number of seats allowed for the lowest open class, using dynamic programming.

Given that $k$ classes are open, we want to find $S_k$, the booking limit on class $k$ such that the expected total revenue from the $k$ classes is maximized. At the very beginning of the booking process we want to compute $S_1$, the booking limit on the lowest class.

If class 1 is the only open class, we have
Chapter 2. A New Single-Leg Static Model

for any $S \leq C$,

$$R_1(S) = \max_{0 \leq S_1 \leq S} \left\{ \sum_{x=0}^{S_1} x \cdot f_1 \cdot p_1(x) + \sum_{x=S_1+1}^{\infty} S_1 \cdot f_1 \cdot p_1(x) \right\} \quad (2.1)$$

Note that we essentially obtain a mapping that maps $S (\leq C)$ to $S_1$. In this particular case the optimal $S_1$ is simply $S$.

Next consider two classes, class 1 and class 2. Suppose a total of $S$ seats are available to classes 1 and 2. We have

$$R_2(S) = \max_{0 \leq S_2 \leq S} \left\{ \sum_{x=0}^{S_2} (f_2 x + R_1(S - x))p_2(x) + \sum_{x=S_2+1}^{\infty} [f_2 S_2 + R_1(S - S_2)]p_2(x) \right\} \quad (2.2)$$

For any $S$, (2.2) will generate an $S_2$, the optimal booking limit on class 2.

In general, we consider classes $1, 2, \ldots, i$ that are open with $S$ seats available to them. We have for $i = 2, \ldots, n$,

$$R_i(S) = \max_{0 \leq S_i \leq S} \left\{ \sum_{x=0}^{S_i} [f_i x + R_{i-1}(S - x)]p_i(x) + \sum_{x=S_i+1}^{\infty} [f_i S_i + R_{i-1}(S - S_i)]p_i(x) \right\} \quad (2.3)$$

Note that a mapping from $S (\leq C)$ to $S_i$ is obtained.

We can construct a $n \times C$ table. The rows are the number of classes open. The column headings are the number of seats available. Entry $(i, S)$ of the table is the booking limit on class $i$, the lowest open class given $S$ seats are available to classes $1, \ldots, i$.

2.2.1 The Algorithm

Let $S(i, k)$ be the booking limit on class $i$ when $k$ seats are available to class $1, 2, \ldots, i$.

Algorithm 2.1 (Direct Wait-and-See Algorithm)

BEGIN

Initialize: Compute $R_1(S)$ from (2.1) for $S = 1, \ldots, C$.  

...
**Iteration:** For $i$ from 2 to $n$, do

for $k = 1, \ldots, C$, do

1. Solve for $S(i, k)$:

\[
S(i, k) = \arg \max_{0 \leq s \leq k} \left\{ \sum_{x=0}^{g} [f_i x + R_{i-1}(k - x)]p_i(x) + \sum_{x=s+1}^{\infty} [f_i s + R_{i-1}(k - s)]p_i(x) \right\}
\]

2. Compute value $R_i(k)$:

\[
R_i(k) = \sum_{x=0}^{S(i,k)} [f_i x + R_{i-1}(k - x)]p_i(x) + \sum_{x=S(i,k)+1}^{\infty} [f_i S(i,k) + R_{i-1}(k - S(i,k))]p_i(x)
\]

**Output:** $S(i, k), i = 1, 2, \ldots, n; k = 1, \ldots, C$.

**END**

### 2.2.2 Running Time

To compute $R_i(S)$, the running time is simply $O(S)$. Therefore, in a brute force implementation, the total running time to compute all $R_i(S)$ for $S = 1, \ldots, C$ for a fixed $i$ is $O(C^2)$. If we have $n$ classes, the total running time is $O(n \cdot C^2)$. In a later section, we will exploit the structure of the problem further to propose an improved implementation.

### 2.2.3 Properties of the Revenue Function $R(x)$

We proceed to prove that $R_i(x), i = 1, \ldots, n$ are concave in $x$, i.e., $R_i(x + 1) + R_i(x - 1) \leq 2 \cdot R_i(x)$. Concavity will be used extensively in network models (see Chapter 4). Also as we will see later, we can significantly speed up the computation of booking limits by using the concavity property.

A discrete function $f(x)$ is concave if $\forall x$

\[
f(x + 1) - f(x) \leq f(x) - f(x - 1).
\]
It is quasi-concave if
\[ f(x) \geq \min[f(x - 1), f(x + 1)]. \]

We consider the following generic problem:
\[
g(S, t) = \sum_{x=0}^{t}[f \cdot x + E(S - x)]p(x) + \sum_{x=t+1}^{\infty}[f \cdot t + E(S - t)]p(x),
\]
where \( S \) and \( t \) are integers, \( E(\cdot) \) is a non-decreasing and concave function.

We define
\[
G(S) = g(S, t^*),
\]
where
\[
t^* = \begin{cases} 
\max \{1 \leq t \leq S : f \geq E(S - (t - 1)) - E(S - t)\} & \text{if } f \geq E(S) - E(S - 1) \\
0 & \text{if } f < E(S) - E(S - 1)
\end{cases}
\]
(2.4)

**Proposition 2.1** \( t^* \) maximizes \( g(S, t) \).

**Proof:** We consider two cases as enumerated in the definition of \( t^* \).

**Case 1:** \( f < E(S) - E(S - 1) \).

By definition, \( t^* = 0 \). For all \( S > t \geq 0 \),
\[
g(S, t + 1) = \sum_{x=0}^{t+1}[f \cdot x + E(S - x)]p(x) + \sum_{x=t+2}^{\infty}[f \cdot t + E(S - t - 1)]p(x)
\]
\[
g(S, t) = \sum_{x=0}^{t}[f \cdot x + E(S - x)]p(x) + \sum_{x=t}^{\infty}[f \cdot t + E(S - t)]p(x)
\]

We have,
\[
g(S, t + 1) - g(S, t) = \sum_{x=t}[f + E(S - t - 1) - E(S - t)]p(x)
\]
Because $E(\cdot)$ is concave, we have

$$(f <) \ E(S) - E(S - 1) \leq E(S - t) - E(S - t - 1)$$

hence,

$$f + E(S - t - 1) - E(S - t) < 0$$

which implies

$$g(S, t + 1) - g(S, t) \leq 0$$

Therefore $t^* = 0$ is an optimal solution.

**Case 2:** $f \geq E(S) - E(S - 1)$.

If $t^* < S$, this implies

$$f + E(S - t^*) - E(S - t^* + 1) \geq 0$$

By concavity of $E(\cdot)$, for any $1 \leq t \leq t^*$,

$$f + E(S - t) - E(S - t + 1) \geq 0$$

Thus,

$$g(S, t) - g(S, t - 1) = \sum_{x=t}^{\infty} [f + E(S - t) - E(S - t + 1)p(x) \geq 0$$

Now $t^* < S$ also implies, $f + E(S - t^* - 1) - E(S - t^*) < 0$. We then have,

$$g(S, t^* + 1) - g(S, t^*) = \sum_{x=t}^{\infty} [f + E(S - t^* - 1) - E(S - t^*)p(x) \leq 0$$

Therefore, $t^*$ attains the maximum value for $g(S, t)$.

Finally, if $t^* = S$, then

$$f + E(0) - E(1) \geq 0$$
By concavity of $E(\cdot)$, for any $1 \leq t \leq S$, we have

$$f + E(S - t) - E(S - t + 1) \geq 0$$

Thus,

$$g(S, t) - g(S, t - 1) = \sum_{x=t}^{\infty} [f + E(S - t) - E(S - t + 1)]p(x) \geq 0$$

So $g(S, t)$ is monotonically non-decreasing. Therefore an optimal solution is $t^* = S$.

Proposition 2.1 implies that Eq. (2.4) defines an optimal solution for $g(S, t)$. From now on, unless specified otherwise, we refer to an optimal solution to $g(S, t)$ as one defined by Eq. (2.4).

**Proposition 2.2** $g(S, t)$ is non-decreasing in $S$.

**Proof:** We have

$$g(S + 1, t) = \sum_{x=0}^{t}[f \cdot x + E(S + 1 - x)]p(x) + \sum_{x=t+1}^{\infty} [f \cdot t + E(S + 1 - t)]p(x)$$

$$g(S, t) = \sum_{x=0}^{t}[f \cdot x + E(S - x)]p(x) + \sum_{x=t+1}^{\infty} [f \cdot t + E(S - t)]p(x)$$

Therefore,

$$g(S+1, t) - g(S, t) = \sum_{x=0}^{t}[E(S+1-x) - E(S-x)]p(x) + \sum_{x=t+1}^{\infty} [E(S+1-t) - E(S-t)]p(x)$$

By non-decreasingness of $E(\cdot)$, $g(S + 1, t) - g(S, t) \geq 0$.

**Proposition 2.3** $g(S, t)$ is quasi-concave in $t$.

**Proof:** We have,

$$g(S, t + 1) = \sum_{x=0}^{t+1}[f \cdot x + E(S - x)]p(x) + \sum_{x=t+2}^{\infty} [f \cdot (t + 1) + E(S - t - 1)]p(x)$$
\[ g(S, t) = \sum_{x=0}^{t} [f \cdot x + E(S - x)]p(x) + \sum_{x=t+1}^{\infty} [f \cdot t + E(S - t)]p(x) \]
\[ g(S, t - 1) = \sum_{x=0}^{t-1} [f \cdot x + E(S - x)]p(x) + \sum_{x=t}^{\infty} [f \cdot (t - 1) + E(S - t + 1)]p(x) \]

Therefore,
\[ g(S, t + 1) - g(S, t) = [f + E(S - t - 1) - E(S - t)] \sum_{x=t+1}^{\infty} p(x) \quad (2.5) \]
\[ g(S, t) - g(S, t - 1) = [f + E(S - t) - E(S - t + 1)] \sum_{x=t}^{\infty} p(x) \quad (2.6) \]

By the concavity of \( E(\cdot) \),
\[ E(S - t + 1) - E(S - t) \leq E(S - t) - E(S - t - 1) \]

which implies
\[ f + E(S - t - 1) - E(S - t) \leq f + E(S - t) - E(S - t + 1). \]

From (2.5) and (2.6) we obtain:
- If \( g(S, t + 1) - g(S, t) \geq 0 \), then \( g(S, t) - g(S, t - 1) \geq 0 \),
- If \( g(S, t) - g(S, t - 1) \leq 0 \), then \( g(S, t + 1) - g(S, t) \leq 0 \).

This implies \( g(S, t) \geq \min[g(S, t + 1), g(S, t - 1)]. \)

**Proposition 2.4** Let
\[ G(S) = g(S, t^*), \]
\[ G(S + 1) = g(S + 1, u^*). \]

Then
Chapter 2. A New Single-Leg Static Model

1. \(0 \leq u^* - t^* \leq 1\)

2. \(u^* = t^*\) if and only if \(u^* = t^* = 0\).

**Proof:** We have

\[
g(S, t^*) - g(S, t^* - 1) = [f + E(S - t^*) - E(S - t^* + 1)] \sum_{x=t^*}^{\infty} p(x).
\]

\[
g(S + 1, u^*) - g(S + 1, u^* - 1) = [f + E(S + 1 - u^*) - E(S + 2 - u^*)] \sum_{x=u^*}^{\infty} p(x).
\]

By the optimality of \(t^*\) (for \(g(S, \cdot)\)), we have

\[
f + E(S - t^*) - E(S + 1 - t^*) \geq 0.
\]

By the concavity of \(E(\cdot)\) we obtain

\[
f + E(S + 1 - t^*) - E(S + 2 - t^*) \geq f + E(S - t^*) - E(S + 1 - t^*) \geq 0,
\]

i.e.,

\[
g(S + 1, t^*) \geq g(S + 1, t^* - 1).
\]

Since \(u^*\) is optimal for \(g(S + 1, t)\) and \(g(S + 1, t)\) is concave in \(t\),

\[
u^* \geq t^*.
\]

We proceed to show that \(u^* - t^* \leq 1\). Suppose on the contrary \(u^* - t^* > 1\). By optimality of \(u^*\) (for \(g(S + 1, \cdot)\)),

\[
f + E(S + 1 - u^*) - E(S + 2 - u^*) \geq 0.
\]

Let \(v^* = u^* - 1\), then

\[
f + E(S - v^*) - E(S + 1 - v^*) \geq 0,
\]
i.e.,
\[ g(S, v^*) - g(S, v^* - 1) \geq 0. \]

By concavity of \( g(S, t) \) in \( t \), \( t^* \) could not have been optimal and therefore
\[ v^* \leq t^* \Rightarrow u^* - t^* \leq 1. \]

To prove 2, suppose \( u^* = t^* \), but \( t^* > 0 \). Then \( t^* - 1 \) is well-defined. By definition of \( t^* \), we have:
\[ f \geq E(S - (t^* - 1)) - E(S - t^*) = E(S + 1 - t^*) - E(S - t^*) \quad (2.7) \]

By definition of \( u^* \), we have:
\[ f \geq E(S + 1 - (u^* - 1)) - E(S + 1 - u^*) \]
and
\[ f < E(S + 1 - u^*) - E(S + 1 - (u^* + 1)) \quad (2.8) \]

Since \( u^* = t^* \), Eq. (2.8) becomes
\[ f < E(S + 1 - t^*) - E(S - t^*) \]
which is a contradiction to Eq. (2.7).

**Theorem 2.5** \( G(\cdot) \) is concave.

The proof of Theorem 2.5 is rather lengthy and can be found in Appendix A.1.

**Corollary 2.6** The revenue function \( R_i(\cdot), i = 1, \ldots, n, \) are concave.

**Proof:** It can be shown easily that \( R_1(\cdot) \) is concave. Using recursion (2.3) and applying Theorem 2.5, we have \( R_i(\cdot) \) is concave, for \( i = 2, \ldots, n \).
Having established the concavity for $R_i(\cdot) \ (i = 1, \ldots, n)$, the optimal booking limits $S(i, k) \ (i = 1, \ldots, n$ and $k = 1, \ldots, C)$ in the direct wait-and-see algorithm (Algorithm 2.1) can be equivalently defined by Eq. (2.4).

$$S(i, k) = \begin{cases} \max \{1 \leq t \leq k : f_i \geq R_{i-1}(k - (t - 1)) - R_{i-1}(k - t)\} , & \text{if } f_i \geq R_{i-1}(k) - R_{i-1}(k - 1) \\ 0 & \text{if } f_i < R_{i-1}(k) - R_{i-1}(k - 1) \end{cases} \quad (2.9)$$

As a direct result of Proposition 2.4, we have the following very useful property of $S(i, k)$.

**Theorem 2.7** The following relation holds:

$$S(i, k - 1) = \begin{cases} S(i, k) - 1 & \text{if } S(i, k) > 0 \\ 0 & \text{if } S(i, k) = 0 \end{cases}$$

for $i = 1, \ldots, n$ and $k = 1, \ldots, C$.

**Proof:** By Proposition 2.4, we have

$$S(i, k) - S(i, k - 1) = 0 \text{ or } 1$$

If $S(i, k) > 0$, by Proposition 2.4, we must have

$$S(i, k) - S(i, k - 1) = 1$$

If $S(i, k) = 0$, $S(i, k - 1)$ must be 0, as $S(i, k - 1) \leq S(i, k)$.

Theorem 2.7 implies that for any class $i \ (1 \leq i \leq n)$
1. The booking limits $S(i, k)$ have the following pattern:

$$[S(i, 0), S(i, 1), S(i, 2), \ldots, S(i, C)] = [0, 0, \ldots, 0, 1, 2, \ldots, C - m_i]$$

where $m_i$ is a non-negative integer between 0 and $C$.

2. Once we find $k_{i0}$ such that $S(i, k_{i0}) = 1$, the rest $S(i, k)$'s ($k_{i0} < k \leq C$) are simply $S(i, k_{i0}) + k - k_{i0}$.

3. If $S(i, k) = 0$, then $R_i(k) = R_{i-1}(k)$. If $S(i, k) > 0$, then

$$R_i(k) = \sum_{x=0}^{S(i,k)} [f_i x + R_{i-1}(k - x)] p(x) +$$

$$[f_i \cdot S(i, k) + R_{i-1}(k - S(i, k))] \sum_{x=S(i,k)+1}^{\infty} p_i(x)$$

$$R_i(k - 1) = \sum_{x=0}^{S(i,k)-1} [f_i x + R_{i-1}(k - 1 - x)] p(x) +$$

$$[f_i \cdot (S(i, k) - 1) + R_{i-1}(k - 1 - (S(i, k) - 1))] \sum_{x=S(i,k)}^{\infty} p_i(x)$$

We get

$$R_i(k) - R_i(k - 1) = \sum_{x=0}^{S(i,k)-1} [R_{i-1}(k - x) - R_{i-1}(k - 1 - x)] p_i(x) + f_i \sum_{x=S(i,k)}^{\infty} p_i(x)$$

The direct wait-and-see algorithm (Algorithm 2.1) can be improved as a result of the above discussion.

Algorithm 2.2 (Improved Wait-and-See Algorithm)

BEGIN

Initialize: Set $S(i, 0) = 0$ and compute $R_1(S)$ from (2.1) for $S = 1, \ldots, C$.

Iteration: For $i$ from 2 to $n$, do

for $k = 1, \ldots, C$, do
1. Solve for $S(i, k)$:

   if $S(i, k - 1) = 0$, then
   
   if $f_i < R_{i-1}(k) - R_{i-1}(k - 1)$, then
   
   $S(i, k) = 0$
   
   else
   
   $S(i, k) = \max \{1 \leq t \leq k : f_i \geq R_{i-1}(k - (t - 1)) - R_{i-1}(k - t)\}$
   
   endif

   else

   $S(i, k) = S(i, k - 1) + 1$

   endif

2. Compute value $R_i(k)$ using (2.10), i.e.,

$$R_i(k) - R_i(k - 1) = \sum_{x=0}^{S(i,k)-1} [R_{i-1}(k - x) - R_{i-1}(k - 1 - x)] p_i(x) + f_i \sum_{x=S(i,k)}^{\infty} p_i(x)$$

Output: $S(i, k), i = 1, 2, \ldots, n; k = 1, \ldots, C.$

END

As the function $R_{i-1}(k - (t - 1)) - R_{i-1}(k - t)$ is non-decreasing in $t$, we can compute $S(i, k)$ using binary search with $O(\log k)$ comparisons. Notice that only differences $R_{i-1}(\cdot) - R_{i-1}(\cdot - 1)$ are needed and that the first term on the right-hand side of (2.10) constitutes an approximate convolution. Using fast Fourier transform techniques, we can calculate all $R_{i-1}(k) - R_{i-1}(k - 1), k = 1, \ldots, C$ in $O(C \log C)$ summations and multiplications, leading to an $O(nC \log C)$ running time to compute $R_i(k) - R_i(k - 1)$ for all $i = 1, \ldots, n$ and $k = 1, \ldots, C$. In total, the improved wait-and-see algorithm takes $O(n \log C)$ comparisons and $O(nC \log C)$ summations and multiplications. In our experiments, we have found that our implementation of the improved wait-and-see algorithm is very fast (comparable in speed with the EMSR heuristic algorithm).
2.2.4 Wait-and-See Strategy versus an Optimal A Priori Strategy

In the wait-and-see approach we determine the booking limits one at a time and will not calculate the booking limit for a particular class until right before the class starts booking.

On the contrary, an a priori approach, by definition, determines the booking limits for all fare classes before the booking process starts. Let \( B_i \) be the a priori booking limit on class \( i \), \( i = 1, \ldots, n \). Let \( y_i \) be the realization of class \( i \) demand and \( x_i \) be the accommodated class \( i \) demand, \( i = 1, \ldots, n \). Under the a priori rule, we will have:

\[
\begin{align*}
x_n &= \min(y_n, B_n) \\
x_{n-1} &= \min(y_{n-1}, B_{n-1} - x_n) \\
x_{n-2} &= \min(y_{n-2}, B_{n-2} - x_n - x_{n-1}) \\
&\vdots \\
x_1 &= \min(y_1, B_1 - \sum_{k=2}^{n} x_k)
\end{align*}
\]

It is conceivable that the wait-and-see approach performs better than a priori because the former postpones decisions until the last minute and therefore utilizes more information. Indeed the following theorem provides concrete a proof for the claim.

**Theorem 2.8** The wait-and-see strategy is at least as good as an a priori optimal strategy.

**Proof:** We use induction on the number of classes \( i \). When \( i = 2 \), the wait-and-see approach and the a priori approach are identical because for the wait-and-see approach there is no future decision to made as the only decision is the booking limit on the lower class. We need to show now that if wait-and-see out-performs a priori for the case of \( i \) classes, then the same holds for the case of \( i + 1 \) cases.
We define $R^w_i(S)$ to be the optimal revenue generated by the wait-and-see approach when classes 1 through $i$ are open and $S$ seats are available to them. Define $R^a_i(S)$ to be the optimal revenue generated by the \textit{a priori} approach when classes 1 through $i$ are open and $S$ seats are available to them. Let $B_n \leq B_{n-1} \leq \ldots \leq B_1$ be the booking limits generated by the \textit{a priori} approach. Clearly,

$$R^w_{i+1}(S) = \max_{0 \leq t \leq S} \left\{ \sum_{x=0}^{t} [f_{i+1} x + R^w_i(S - x)] p_{i+1}(x) + [f_{i+1} t + R^w_i(S - t)] \sum_{x=t+1}^{\infty} p_{i+1}(x) \right\}$$

$$\geq \sum_{x=0}^{B_{i+1}} [f_{i+1} x + R^w_i(S - x)] p_{i+1}(x) + [f_{i+1} B_{i+1} + R^w_i(S - B_{i+1})] \sum_{x=B_{i+1}+1}^{\infty} p_{i+1}(x)$$

$$\geq \sum_{x=0}^{B_{i+1}} [f_{i+1} x + R^a_i(S - x)] p_{i+1}(x) + [f_{i+1} B_{i+1} + R^a_i(S - B_{i+1})] \sum_{x=B_{i+1}+1}^{\infty} p_{i+1}(x).$$

We used the induction assumption in the last inequality. Now,

$$R^a_{i+1}(S) = \sum_{x=0}^{B_{i+1}} [f_{i+1} x + \tilde{R}^a_i(S - x)] p_{i+1}(x) + [f_{i+1} B_{i+1} + \tilde{R}^a_i(S - B_{i+1})] \sum_{x=B_{i+1}+1}^{\infty} p_{i+1}(x)$$

where $\tilde{R}^a_i(\cdot)$ is the expected revenue generated by the \textit{a priori} approach from classes 1 through $i$, after class $i + 1$ is closed. Apparently we have:

$$R^a_i(\cdot) \geq \tilde{R}^a_i(\cdot)$$

since by definition $R^a_i(\cdot)$ is the revenue under the optimal \textit{a priori} approach.

All considered, we have:

$$R^w_{i+1}(S) \geq R^a_{i+1}(S).$$

We just showed that the wait-and-see approach provides an upper bound on the \textit{a priori} approach. Next we show that if we properly choose the \textit{a priori} booking limits
the a priori will achieve the same results as the wait-and-see approach.

Optimal A Priori Booking Limits

Recall that $S(i, k)$ is the booking limit on class $i$ when the $k$ seats are available to classes 1 through $i$, where $i = 1, 2, \ldots, n$. We define:

$$B_i = S(i, C), \quad i = 1, 2, \ldots, n \tag{2.11}$$

where $C$ is the flight capacity.

**Theorem 2.9** Suppose the capacity is $C$ and there are $n$ fare classes. Let $B_i$ be as defined in Eq. (2.11). Then $B_i$'s are optimal a priori booking limits.

To prove Theorem 2.9, we need to first prove some propositions.

**Proposition 2.10** For any $C$, $R_i(C + 1) - R_i(C)$ is non-decreasing in $i$ ($R_0(\cdot) \equiv 0$).

**Proof:** Let $t_i^*$ and $u_i^*$ be the optimal solutions to $R_i(C)$ and $R_i(C + 1)$, respectively.

By Proposition 2.4,

$$u_i^* = t_i^* \text{ or } t_i^* + 1.$$

We discuss the two cases separately.

**Case 1:** $u_i^* = t_i^*$.

We have

$$R_i(C + 1) = \sum_{x=0}^{t_i^*} [f_i \cdot x + R_{i-1}(C + 1 - x)]p_i(x) + \sum_{x=t_i^*+1}^{\infty} [f_i \cdot t_i^* + R_{i-1}(C + 1 - t_i^*)]p_i(x)$$

$$R_i(C) = \sum_{x=0}^{t_i^*} [f_i \cdot x + R_{i-1}(C - x)]p_i(x) + \sum_{x=t_i^*+1}^{\infty} [f_i \cdot t_i^* + R_{i-1}(C - t_i^*)]p_i(x)$$

So,

$$R_i(C + 1) - R_i(C) = \sum_{x=0}^{t_i^*} [R_{i-1}(C + 1 - x) - R_{i-1}(C - x)]p_i(x) + \sum_{x=t_i^*+1}^{\infty} [R_{i-1}(C + 1 - t_i^*) - R_{i-1}(C - t_i^*)]p_i(x)$$
Chapter 2. A New Single-Leg Static Model

By concavity of $R_{i-1}(\cdot)$,

$$R_{i-1}(C + 1 - x) - R_{i-1}(C - x) \geq R_{i-1}(C + 1) - R_{i-1}(C)$$

Thus,

$$R_i(C + 1) - R_i(C) \geq \sum_{x=0}^{t_i^*} [R_{i-1}(C + 1) - R_{i-1}(C)]p_i(x) + \sum_{x=t_i^* + 1}^{\infty} [R_{i-1}(C + 1) - R_{i-1}(C)]p_i(x)$$

$$= R_{i-1}(C + 1) - R_{i-1}(C)$$

**Case 2: $u_i^* = t_i^* + 1$.**

We have

$$R_i(C + 1) = \sum_{x=0}^{t_i^* + 1} [f_i \cdot x + R_{i-1}(C + 1 - x)]p_i(x) + \sum_{x=t_i^* + 2}^{\infty} [f_i \cdot (t_i^* + 1) + R_{i-1}(C - t_i^*)]p_i(x)$$

$$R_i(C) = \sum_{x=0}^{t_i^*} [f_i \cdot x + R_{i-1}(C - x)]p_i(x) + \sum_{x=t_i^* + 1}^{\infty} [f_i \cdot t_i^* + R_{i-1}(C - t_i^*)]p_i(x)$$

So we have,

$$R_i(C + 1) - R_i(C) = \sum_{x=0}^{t_i^*} [R_{i-1}(C + 1 - x) - R_{i-1}(C - x)]p_i(x) + \sum_{x=t_i^* + 1}^{\infty} f_i \cdot p_i(x)$$

By optimality of $u_i^* = t_i^* + 1$, we have

$$f_i \geq R_{i-1}(C + 1 - t_i^*) - R_{i-1}(C - t_i^*)$$

Thus,

$$R_i(C + 1) - R_i(C) \geq \sum_{x=0}^{t_i^*} [R_{i-1}(C + 1) - R_{i-1}(C)]p_i(x) + \sum_{x=t_i^* + 1}^{\infty} [R_{i-1}(C + 1) - R_{i-1}(C)]p_i(x)$$

$$= R_{i-1}(C + 1) - R_{i-1}(C)$$

Proposition 2.10 implies that the more classes the more marginal expected revenue an additional seat will generate.
Proposition 2.11 \(B_i\) as defined in Eq. (2.11) are non-increasing in \(i\).

**Proof:** We only need to prove that \(B_i \geq B_{i+1}\).

If \(B_i = 0\), then \(f_i < R_{i-1}(C) - R_{i-1}(C - 1)\). Using Proposition 2.10, we have:

\[
f_{i+1} < f_i < R_{i-1}(C) - R_{i-1}(C - 1) \leq R_i(C) - R_i(C - 1)
\]

Thus \(B_{i+1} = 0\).

If \(B_i = C\), then \(B_{i+1} \leq C = B_i\).

If \(1 \leq B_i < C\), we have

\[
f_i < R_{i-1}(C - B_i) - R_{i-1}(C - B_i - 1)
\]

Using Proposition 2.10, we have

\[
f_{i+1} < f_i < R_{i-1}(C - B_i) - R_{i-1}(C - B_i - 1) \leq R_i(C - B_i) - R_i(C - B_i - 1)
\]

which implies that \(B_{i+1} \leq B_i\).

Proposition 2.12 If classes \(n, n-1, \ldots, i+1\) have been closed and the seats sold to them are \(x_n, x_{n-1}, \ldots, x_{i+1}\), which satisfy

\[
x_n \leq B_n
\]

\[
x_n + x_{n-1} \leq B_{n-1}
\]

\[
\ldots \ldots
\]

\[
x_n + x_{n-1} + \cdots + x_{i+1} \leq B_{i+1}
\]
where $B_1, B_2, \ldots, B_n$ are as defined in Eq. (2.11), then the optimal wait-and-see booking limit for class $i$ is

$$B_i = \sum_{k=i+1}^{n} x_k$$

Proof: We discuss 3 different cases.

Case 1: $B_i = 0$. This is a trivial case.

Case 2: $B_i = C$.

From Eq. (2.4), we must have:

$$f_i \geq R_{i-1}(1) - R_{i-1}(0)$$

If $x_n + x_{n-1} + \cdots + x_{i+1} = B_i (= C)$, this is the case where all the seats have been sold.

If $x_n + x_{n-1} + \cdots + x_{i+1} < B_i$. In this case there are $B_i - \sum_{k=i+1}^{n} x_k$ seats left. Given these many seats we need to determine the optimal wait-and-see booking limit on class $i$. But

$$B_i \geq R_{i-1}((C - \sum_{k=i+1}^{n} x_k) - (C - \sum_{k=i+1}^{n} x_k - 1)) - R_{i-1}((C - \sum_{k=i+1}^{n} x_k) - (C - \sum_{k=i+1}^{n} x_k))$$

By Eq. (2.4), $C - \sum_{k=i+1}^{n} x_k$ is the optimal wait-and-see booking limit on class $i$.

Case 3: $1 \leq B_i < C$.

By definition of $B_i$, we have

$$f_i \geq R_{i-1}(C - (B_i - 1)) - R_{i-1}(C - B_i) \quad (2.12)$$

$$f_i < R_{i-1}(C - B_i) - R_{i-1}(C - (B_i + 1)) \quad (2.13)$$

If $x_n + x_{n-1} + \cdots + x_{i+1} = B_i$, then from (2.13)

$$f_i < R_{i-1}(C - \sum_{k=i+1}^{n} x_k) - R_{i-1}(C - \sum_{k=i+1}^{n} x_k - 1)$$
Chapter 2. A New Single-Leg Static Model

By Eq. (2.4), the optimal wait-and-see booking limit on class $i$ should be zero, which is equal to $B_i - \sum_{k=i+1}^{n} x_k$.

If $x_n + x_{n-1} + \cdots + x_{i+1} < B_i$, from (2.12) and (2.13), we have

$$f_i \geq R_{i-1}((C - \sum_{k=i+1}^{n} x_k) - (B_i - \sum_{k=i+1}^{n} x_k)) - R_{i-1}((C - \sum_{k=i+1}^{n} x_k) - (B_i - \sum_{k=i+1}^{n} x_k))$$

$$f_i < R_{i-1}((C - \sum_{k=i+1}^{n} x_k) - (B_i - \sum_{k=i+1}^{n} x_k)) - R_{i-1}((C - \sum_{k=i+1}^{n} x_k) - (B_i - \sum_{k=i+1}^{n} x_k + 1))$$

By Eq. (2.4), the optimal wait-and-see booking limit for class $i$ is

$$B_i - \sum_{k=i+1}^{n} x_k$$

We have just shown that with $B_1, B_2, \ldots, B_n$ the wait-and-see approach and the a priori approach will result in the same policy for each booking class, thus the same revenue. But we showed previously that for any instance the wait-and-see approach produces a revenue that is an upper bound on the revenue produced by the a priori approach. Therefore $B_1, B_2, \ldots, B_n$ must be a set of optimal a priori booking limits and Theorem 2.9 is proved.

In the case of 2 classes, we know that wait-and-see and a priori are identical. Notice that the booking limit on the lower class has nothing to do with its own distribution. In fact, if we carefully examine Eq. (2.4), we realize that booking limit $B_i$ depends on its fare and the fares and demand distributions of all its higher classes, but not on its own distribution.

Remark: The results in this section are a generalization of McGill's results [10] (see Eq. (1.1)) in that we have actually found an algorithm that speedily computes optimal booking limits for the generic case.
2.3 A Single-Leg Static Model with Cancelations and Overbookings

The assumption in the previous section has been that a request for a seat will eventually materialize. In practice, however, reservations can be canceled any time without penalty. If in such a case airlines strictly stick to the physical flight capacity they will likely suffer a loss of revenue resulting from cancelations. To counter cancelations, airlines normally overbook their flight, i.e., they accept more reservations than the actual flight capacity.

The issue of overbooking is two-folded. On one hand, overbooking should be at a level high enough to buffer potential cancelations and no-shows. A level too low will likely leave seats empty. On the other hand, if overbooking is carried out too aggressively, the airline will likely pay excessive penalty for denied boardings. The goal is to find a overbooking level that maintains the best balance between the risk of lost revenue and that of excessive penalty.

Past overbooking models have followed a framework of scaling the flight capacity. In this section, we develop a model that directly and explicitly handles cancelations and overbookings. A penalty cost is imposed on bookings that are over the flight capacity. The goal is to maximize the total expected revenue, which is the total expected ticket sales minus the total overbooking penalty.

Without cancelations and overbookings, the information needed to characterize the state of the system is the currently booking class and the number of seats still available. With cancelations and overbookings, we will need to keep a record of how many seats are reserved for each class.

2.3.1 Notation

In addition to the notation, such as fares, distributions, etc., defined for the case without cancelations and overbookings, we define:
• $w$: overbooking penalty,

• $r_i$: probability that a class $i$ booking will materialize,

• $X_i$: vector that represents reservations for classes $i$ through $n$, i.e., $X_i = (x_i, x_{i+1}, \ldots, x_n)$, where $x_i$ through $x_n$ are reservations for classes $i$ through $n$, respectively,

• $R_i(C, X_{i+1})$: expected remaining revenue when class $i$ is open and reservations is $X_{i+1} = (x_{i+1}, x_{i+2}, \ldots, x_n)$.

The expected remaining revenue means the revenue to be expected from now until departure.

### 2.3.2 Boundary Condition – Overbooking Penalty

At departure time, all reservations are final. What remains stochastic is how many reservations will materialize. We have:

$$R_0(C, X_1) = w \cdot E[\min(0, C - \sum_{i=1}^{n} \sum_{j=1}^{x_i} Y_{ij})] \quad (2.14)$$

where for $i = 1, \ldots, n$,

$$Y_{ij} = \begin{cases} 
1 & \text{w.p. } r_i, \quad j = 1, \ldots, x_i \\
0 & \text{w.p. } 1-r_i 
\end{cases}$$

are independent Bernoulli random variables.
2.3.3 Dynamic Programming Recursion

Consider the case where all classes but the highest fare class are closed. We know the bookings \(x_2, x_3, \ldots, x_n\). The expected remaining revenue is:

\[
R_1(C, X_2) = \max_{S \geq 0} \left\{ \sum_{x_1 = 0}^{S} [f_1 r_1 x_1 + R_0(C, x_1, x_2, \ldots, x_{n-1}, x_n)] p_1(x_1) \\
+ [f_1 r_1 S + R_0(C, S, x_2, \ldots, x_{n-1}, x_n)] \sum_{x_1 = 0}^{\infty} p_1(x_1) \right\} 
\]

(2.15)

The general recursion is as follows:

\[
R_i(C, X_{i+1}) = \max_{S \geq 0} \left\{ \sum_{x_i = 0}^{S} [f_i r_i x_i + R_{i-1}(C, x_i, x_{i+1}, \ldots, x_n)] p_i(x_i) + \\
[f_i r_i S + R_{i-1}(C, S, x_{i+1}, \ldots, x_n)] \sum_{x_i = S+1}^{\infty} p_i(x_i) \right\} 
\]

(2.16)

2.3.4 Approximation of Overbooking Penalty

We have seen that with cancellations and overbookings, the dimensionality of the model increases drastically. To tackle the dimensionality difficulty, we introduce an approximation of the overbooking penalty. In boundary condition (2.14), we exchange the order of expectation and minimum, i.e., we approximate \(E[\min(0, C - \sum_{i=1}^{n} \sum_{j=1}^{x_i} Y_{ij})] \) with \(\min(0, E[C - \sum_{i=1}^{n} \sum_{j=1}^{x_i} Y_{ij}] \). As a result, the approximated penalty cost is:

\[
\hat{R}_0(C, x_1, x_2, \ldots, x_n) = w \cdot \min(0, C - \sum_{i=1}^{n} r_i x_i). 
\]

(2.17)

Note that \(\hat{R}_0\) is now only a function of the expected surplus capacity, i.e., the capacity minus the sum of the expected materialized reservations from each individual class. This implies that for the boundary condition, we do not need to know the number of reservations for each individual class any more; instead we only need the total number
of materialized reservations. This holds in general: the approximated expected revenue is only a function of the expected surplus capacity. With this approximation the recursion (2.16) can be simplified as follows. Let \( \hat{R}(x) \) be the approximated expected revenue, where \( x \) is the expected surplus capacity. Note that \( x \) is not necessarily an integer and can be negative. When \( x \) is negative it represents overbookings. The recursion for the approximated revenue is:

\[
\hat{R}_i(x) = \max_{S \geq 0} \left\{ \sum_{x_i=0}^{S} [f_i r_i x_i + \hat{R}_{i-1}(x - r_i x_i)] p_i(x_i) + \left[ f_i r_i S + \hat{R}_{i-1}(x - r_i S) \right] \sum_{x_i=S+1}^{\infty} p_i(x_i) \right\}, \tag{2.18}
\]

with the boundary condition being:

\[
\hat{R}_0(x) = w \cdot \min(0, x)
\]

### 2.3.5 Approximated Expected Revenue Function \( \hat{R}(\cdot) \)

In order to define an algorithm for computing booking limits \( S \), we need to first establish some properties for the approximated expected revenue function \( \hat{R}(\cdot) \).

Consider the generic problem

\[
g(S, t) = \sum_{x=0}^{t} [f r x + E(S - r x)] p(x) + \sum_{x=t+1}^{\infty} [f r t + E(S - r t)] p(x)
\]

where \( S \) is real, \( 0 < r \leq 1 \), \( t \) is integer, and \( E(\cdot) \) is non-decreasing and concave. Define

\[
G(S) = g(S, t^*)
\]

where \( t^* = \arg\max_{t \geq 0} g(S, t) \). Without loss of generality assume that \( t^* \) exists (if the unit penalty cost is sufficiently high) and is unique (we can take the smallest such \( t^* \)). We start the proof of the concavity of the revenue function by considering the
following two propositions.

**Proposition 2.13** $g(S, t)$ is non-decreasing in $S$.

**Proof.** We have

$$
g(S + 1, t) = \sum_{x=0}^{t} [r.fx + E(S + 1 - rx)]p(x) + \sum_{x=t+1}^{\infty} [r.ft + E(S + 1 - rt)]p(x).
$$

$$
g(S, t) = \sum_{x=0}^{t} [r.fx + E(S - rx)]p(x) + \sum_{x=t+1}^{\infty} [r.ft + E(S - rt)]p(x).
$$

Therefore,

$$
g(S+1, t) - g(S, t) = \sum_{x=0}^{t} [E(S+1-rx) - E(S-rx)]p(x) + \sum_{x=t+1}^{\infty} [E(S+1-rt) - E(S-rt)]p(x).
$$

By non-decreasingness of $E(\cdot)$, $g(S + 1, t) - g(S, t) \geq 0$.

**Proposition 2.14** Let

$$
G(S) = g(S, t^*)
$$

$$
G(S + 1) = g(S + 1, u^*)
$$

Then $0 \leq u^* - t^* \leq \left\lceil \frac{1}{r} \right\rceil$.

**Proof.** We have

$$
g(S, t^*) - g(S, t^* - 1) = [rf + E(S - rt^*) - E(S - rt^* + r)] \sum_{x=t^*}^{\infty} p(x).
$$

$$
g(S + 1, u^*) - g(S + 1, u^* - 1) = [rf + E(S + 1 - ru^*) - E(S + 1 - u^* + r)] \sum_{x=u^*}^{\infty} p(x).
$$

By optimality of $t^*$ (for $g(S, \cdot)$), we have

$$
rf + E(S - rt^*) - E(S + r - rt^*) \geq 0.
$$
Concavity of $E(\cdot)$ gives

$$rf + E(S + 1 - rt^*) - E(S + 1 + r - rt^*) \geq rf + E(S - rt^*) - E(S + r - rt^*) \geq 0.$$ 

This and the optimality of $u^*$ (for $g(S + 1, \cdot)$) yield

$$u^* \geq t^*.$$

We proceed to show that $u^* - t^* \leq \left\lfloor \frac{1}{r} \right\rfloor$. Suppose on the contrary $u^* - t^* > \left\lfloor \frac{1}{r} \right\rfloor$. By optimality of $u^*$ (for $g(S + 1, \cdot)$),

$$rf + E(S + 1 - ru^*) - E(S + 1 + r - ru^*) \geq 0.$$

Let $v^* = u^* - \left\lfloor \frac{1}{r} \right\rfloor$, we have

$$rf + E(S + 1 - rv^*) - E(S + 1 + r - rv^*) > 0$$

or

$$rf + E(S - rt^* + 1 - r \left\lfloor \frac{1}{r} \right\rfloor) - E(S + r - rt^* + 1 - r \left\lfloor \frac{1}{r} \right\rfloor) > 0,$$

so

$$rf + E(S - rt^*) - E(S + r - rt^*) > 0$$

This implies that $t^*$ is not optimal. Hence we must have

$$u^* - t^* \leq \left\lfloor \frac{1}{r} \right\rfloor.$$

This proposition implies that we will take no more than $\left\lfloor \frac{1}{r} \right\rfloor$ reservations to fill an additional seat.

The revenue function is not concave at all points. We can prove, however, that
the revenue function is concave at points that are \( r \) distance apart.

**Theorem 2.15**

\[
G(S + r) + G(S - r) \leq 2G(S)
\]

*where \( r \) is the materialization (show-up) rate.*

The proof is very similar to that of Theorem 2.5. It is quite lengthy and can be found in Appendix A.2.

### 2.3.6 Concave Approximation of \( G(S) \)

The concavity of \( G(\cdot) \) is guaranteed only at points \( r \) distance apart. However, we will need concavity at all points. What we can do is to obtain a piecewise linear approximation of \( G(\cdot) \). We can show that the piecewise approximation is concave at all points.

**Break Points**

We define:

\[
S_0 = \sup_{-\infty < S < \infty} \{ S : E(S + r) - E(S) \geq rf \}
\]

Essentially, \( S_0 \) is, roughly speaking, the expected surplus capacity for which we can break even by selling any seats at all, i.e., if the surplus capacity is greater than \( S_0 \) we will sell a seat; otherwise it is not worth the risk of overbooking penalty. Notice that the break point does not depend on the capacity. In general \( S_0 + nr \ (n \geq 0) \) are the expected overbookings for which we can sell at most \( n \) seats (easy proof of this can be found in Proposition A.4). We call \( S_0 + nr \) the "break points". We show in Appendix A.3 (Proposition A.4) that \( S_0 + nr \)'s have the property that \( G(\cdot) \) is concave on \( [S_0 + nr, S_0 + (n + 1)r] \).
The Approximation

Once we have the break points the approximation is:

\[
\tilde{G}(S) = \begin{cases} 
G(S_0 + nr) + \frac{S - S_0 - nr}{r}[G(S_0 + (n + 1)r) - G(S_0 + nr)] & \text{if } S_0 + nr < S \leq S_0 + (n + 1)r, \ n \geq 0 \\
G(S) & \text{if } S \leq S_0 
\end{cases}
\]

(2.19)

Essentially we are approximating \( G(S) \) on \([S_0 + nr, S_0 + (n+1)r]\) using the line segment connecting \( S_0 + nr \) and \( S_0 + (n+1)r \). We show in Appendix A.3 (see Proposition A.4) that \( \tilde{G}(\cdot) \) is concave at all points.

Approximation Applied to Revenue Function

The approximation approach is as follows. We approximate \( R_0(\cdot) \) with \( \hat{R}_0(\cdot) \) as defined in Eq. (2.17). We then compute the break points for \( \hat{R}_1(\cdot) \) and approximate \( \hat{R}_1(\cdot) \) using Eq. (2.19) and get \( \hat{R}_1(\cdot) \). Then use \( \hat{R}_1(\cdot) \) in Eq. (2.18) to define \( \hat{R}_2(\cdot) \) and compute the break points for \( \hat{R}_2(\cdot) \). Use the break points to obtain \( \hat{R}_2(\cdot) \). We continue until we reach class \( n \). The detailed algorithm follows.

The Algorithm

We assume that all show-up rates are rational numbers and have at most two decimal digits. By making the assumption, all calculations below are finite over points 0.01 from each other.

In the following algorithm \( S \) denotes the expected surplus capacity and \( b_i \) denotes the break point for class \( i \). We also assume, without loss of generality, that the penalty \( w \) is greater than the highest fare.

Algorithm 2.3 (Break Points Algorithm)

BEGIN

Initialization:
1. The break point for class 1:

\[ b_1 = \arg \max_{0 \leq x \leq C} \{w \cdot \min(0, x + r_1) - w \cdot \min(0, x)\} \]

where the step size of \( x \) is 0.01.

2. If \( S \leq 0, S_1 = 0 \); else

\[ S_1(S) = \left\lfloor \frac{S}{r_1} - 1 \right\rfloor \]

3. For all \( S = b_1 + nr_1 \) for all \( n \geq 0 \) such that

\[ S \leq \left\lfloor \frac{C}{r_1} \right\rfloor \]

compute:

\[ \hat{R}_1(S) = \sum_{x_1 = 0}^{S_1} [f_1 r_1 x_1 + w \cdot \min(0, S - r_1 x_1)] p_1(x_1) + [f_1 r_1 S_1 + w \cdot \min(0, S - r_1 S_1)] \sum_{x_1 = S_1 + 1}^\infty p_1(x_1) \]

where \( S_1 \) is as determined in 2.

4. Use Eq. (2.19) to define, for \( S \leq \left\lfloor \frac{C}{r_1} \right\rfloor \), \( \hat{R}_1(S) \):

\[ \hat{R}_1(S) = \begin{cases} \hat{R}_1(b_1 + nr_1) + \frac{S - b_1 - nr_1}{r_1} [\hat{R}_1(b_1 + (n + 1)r_1) - \hat{R}_1(b_1 + nr_1)] & \text{if } b_1 + nr_1 < S \leq b_1 + (n + 1)r_1, \ n \geq 0 \\ \hat{R}_1(S) & \text{if } S \leq b_1 \end{cases} \]

**Iteration:**

for \( i = 2 \) to \( n \) do:

1. Compute the break point, i.e., solving for \( b_i \)

\[ b_i = \arg \max_{0 \leq x \leq C} \{\hat{R}_{i-1}(x + r_i) - \hat{R}_{i-1}(x) \geq r_i f_i\} \]

where the step size of \( x \) is 0.01.
2. If $b_i \geq S$, booking limit $S_i = 0$; else find $S_i$ such that

$$b_i + S_i r_i < S \leq b_i + (S_i + 1) r_i$$

i.e.,

$$S_i = \left\lfloor \frac{S - b_i}{r_i} - 1 \right\rfloor$$

3. For all $S = b_i + nr_i$ for all $n \geq 0$ such that

$$S \leq \left\lfloor \frac{C}{r_i} \right\rfloor$$

compute:

$$\tilde{R}_i(S) = \sum_{x_i=0}^{S_i} \left[ f_i r_i x_i + w \cdot \min(0, S - r_i x_i) \right] p_i(x_i) + \left[ f_i r_i S_i + w \cdot \min(0, S - r_i S_i) \right] \sum_{x_i=S_i+1}^{\infty} p_i(x_i)$$

where $S_i$ is as determined in 2.

4. Use Eq. (2.19) to define, for $S \leq \left\lfloor \frac{C}{r_i} \right\rfloor$, $\tilde{R}_i(S)$:

$$\tilde{R}_i(S) = \begin{cases} 
\tilde{R}_i(b_i + nr_i) + \frac{S - b_i - nr_i}{r_i} \left[ \tilde{R}_i(b_i + (n + 1)r_i) - \tilde{R}_i(b_i + nr_i) \right] 
& \text{if } b_i + nr_i < S \leq b_i + (n + 1)r_i, \ n \geq 0 \\
\tilde{R}_i(S) 
& \text{if } S \leq b_i 
\end{cases}$$

END

Note that in Steps 1 we are able to restrict the lower bound of $x$ to 0 because of the assumption that the penalty is greater than the highest fare.

Computational Complexity

A breakdown of the running time for the above algorithm is as follows:

- Computing break points: $O(nC)$
Chapter 2. A New Single-Leg Static Model

- Computing booking limits: $O(nC)$

- Computing revenue function $\tilde{R}_i(\cdot)$: since the booking limit $S_i \leq \frac{C}{r_i}$, the time spent on computing the revenue function $\tilde{R}_i(\cdot)$ will be no more than

$$\frac{C}{r_i} \cdot \frac{C}{r_i}$$

The time spent on computing all $\tilde{R}_i(\cdot), i = 1, \ldots, n$ is $\sum_{i=1}^{n} \frac{C^2}{r_i^2}$.

The total running time is $O\left(\frac{nC^2}{\min_i\{r_i^2\}}\right)$.

Bound on Error of Approximation

As we have seen previously, the approximated $R_0$ (§2.3.4) is an over-estimation. The approximations in §2.3.6 is an under-estimation. We need to have a bound on the errors of the approximations.

**Theorem 2.16**

$$|\tilde{G}(S) - G(S)| \leq rf.$$  

The proof can be found in Appendix A.3. In general if we have $n$ classes, the errors of approximation is bounded by $\sum_{i=1}^{n} r_i f_i$. 
Chapter 3

Single-Leg Dynamic Models

3.1 Introduction

Static models rely on the static assumption, i.e., lower classes book up first. In reality this assumption is not valid in many cases. For example, an executive may very well choose to purchase his full-fare ticket one month in advance for a scheduled meeting in order to avoid potential last-minute difficulties in securing a seat. As another example, last-minute promotion sales will generate additional demands for discount fares.

The static assumption is in fact a very strong assumption about the order of arrivals of fare requests. It implies that the arrivals of fare requests happen in mutually exclusive time windows. As a result, the exact timing of arrival of each individual fare request can be ignored. However, when fare requests for different fare classes arrive mixed, the order in which fare requests arrive becomes important. In this case we need to consider the actual process in which arrivals occur.

We model fare requests as stochastic processes. We discretize the booking horizon into small intervals. We model arrivals as a geometric process, i.e., within each small interval there can be at most one request from one of the fare classes. Each time a request occurs, we need to decide whether to accept or reject the request depending
on the number of seats available and the timing of the request. We introduce the
idea of the “threshold time” for each fare class for the case where no cancellations
are allowed. We show that the threshold time does exist. The introduction of the
threshold time greatly simplifies both the computation and the seat inventory control.

For the case with cancellations, we formulate a dynamic programming model simi-
lar to the case without cancellations. We demonstrate that in this case the threshold
time is no longer well-defined. We conjecture that a threshold interval can be defined
instead.

3.2 A Single-Leg Dynamic Model

In this section we propose a dynamic model which decides whether to accept or reject
a request at time $t$ depending on the number of seats available at time $t$.

Let us consider a booking horizon $[0, T]$, with $T$ being the beginning of the booking
process and 0 the departure time. We partition the entire booking horizon into small
discrete intervals. In each interval at most one request from one of the fare classes can
occur. This request originates from a class $i$ customer ($i = 1, \ldots, K$) with probability
$p_i$. A class is defined by its fare $f_i$, where $f_K < f_{K-1} < \cdots < f_1$. The objective is
to dynamically accept/reject each incoming request in order to maximize expected
revenue.

Let $R(n, t)$ be the expected revenue at time $t$ from the departure time (time 0)
with $n$ seats available. For boundary conditions we have:

$$ R(n, 0) = 0, \forall n > 0. $$
$$ R(0, t) = 0, \forall t. $$

Let $r(n, i, t)$ be the (optimal) expected revenue at time $t$ with $n$ seats available and
given that a request of class $i$ occurs at time $t$. It follows that:

$$ r(n, i, t) = \max\{f_i + R(n - 1, t - 1), R(n, t - 1)\}. $$

$$ R(n, t) = \sum_{i=1}^{K} p_i \cdot r(n, i, t) + (1 - \sum_{i=1}^{K} p_i) \cdot R(n, t - 1). $$

The dynamic programming recursion is:

For $n = 1, \ldots, C$ (flight capacity),

$$ R(0, 0) = 0. $$

For $t = 1, \ldots, T$,

$$ R(n, t) = \sum_{i=1}^{K} p_i \max\{f_i + R(n - 1, t - 1), R(n, t - 1)\} + (1 - \sum_{i=1}^{K} p_i) R(n, t - 1). $$

Clearly, it takes $O(CT)$ steps to perform the recursion.

### 3.3 Threshold Times

For the dynamic model we can define threshold times as follows. The time $t_n^i$ is called the threshold time for class $i$ when $n$ seats are available, if it satisfies:

$$ f_i + R(n - 1, t_n^i - 1) \geq R(n, t_n^i - 1). \quad (3.1) $$

$$ f_i + R(n - 1, t_n^i) < R(n, t_n^i). \quad (3.2) $$

Intuitively, the threshold time is a point in time on the booking horizon, before which requests are rejected and after which requests are accepted. See Figure 3-1 for an illustration.
3.3.1 Properties of the Revenue Function and the Threshold Times

The revenue function $R(n, t)$ has some interesting and useful properties. These properties are intuitive and can simplify the algorithm. Most importantly, they allow us to define the notion of the threshold time. In what follows, we state and prove each of the properties and afterwards provide an intuitive meaning of the property.

**Proposition 3.1** $R(n, t)$ is non-decreasing in $t$.

**Proof:**

\[
R(n, t) = \sum_{i=1}^{K} p_i \cdot \max\{f_i + R(n - 1, t - 1), R(n, t - 1)\} + (1 - \sum_{i=1}^{K} p_i) \cdot R(n, t - 1)
\]

\[
\geq \sum_{i=1}^{K} p_i \cdot R(n, t - 1) + (1 - \sum_{i=1}^{K} p_i) \cdot R(n, t - 1) = R(n, t - 1).
\]

**Proposition 3.2** $R(n, t)$ is non-decreasing in $n$.

**Proof:** We use induction on $t$.

For $t = 0$, we have $R(n + 1, 0) = R(n, 0) = \sum_{i=1}^{K} p_i f_i$, $\forall n$. Suppose $R(n + 1, t - 1) \geq$
\[ R(n, t - 1), \forall n. \text{ We want to show } R(n + 1, t) \geq R(n, t), \forall n. \]

Now

\[
R(n + 1, t) = \sum_{i=1}^{K} p_i \cdot r(n + 1, i, t) + (1 - \sum_{i=1}^{K} p_i) \cdot R(n + 1, t - 1)
\geq \sum_{i=1}^{K} p_i \cdot r(n + 1, i, t) + (1 - \sum_{i=1}^{K} p_i) \cdot R(n, t - 1).
\]

We only need to show

\[ r(n + 1, i, t) \geq r(n, i, t). \]

Recall that,

\[
r(n + 1, i, t) = \max\{f_i + R(n, t - 1), R(n + 1, t - 1)\},
\]

\[
r(n, i, t) = \max\{f_i + R(n - 1, t - 1), R(r, t - 1)\}.
\]

In general, if \( x \geq s \) and \( y \geq t \), then \( \max(x, y) \geq \max(s, t) \). In fact, if \( y \geq x \), then \( y \geq x \geq s, y \geq t \), thus \( y \geq \max(s, t) \); else if \( y < x \), then \( x > y \geq t, x \geq s \), thus \( x \geq \max(s, t) \).

Thus implies

\[ r(n + 1, i, t) \geq r(n, i, t). \]

Therefore

\[
R(n + 1, t) \geq \sum_{i=1}^{K} p_i \cdot r(n, i, t) + (1 - \sum_{i=1}^{K} p_i) \cdot R(n, t - 1) \equiv R(n, t).
\]

This proposition says that the more seats we have available, the more revenue we can expect.

**Proposition 3.3** \( R(n, t) \) is concave in \( n \).

The proof of the theorem is rather lengthy and can be found in Appendix A.4.
The proposition means that given \( t \) time units from departure the more seats there are available the less marginal expected revenue an additional seat will bring. Essentially it implies that given a fixed amount of time, the more seats we have the less likely we can fully utilize them, thus the less useful each additional seat is.

**Proposition 3.4** Incremental revenue is non-decreasing in time, i.e.,

\[
R(n, t + 1) - R(n - 1, t + 1) \geq R(n, t) - R(n - 1, t).
\]

The proof of the proposition can be found in Appendix A.5.

This proposition means that given \( n \), the number of seats available, the more time we have the more likely an additional seat will be utilized and thus more valuable.

**Proposition 3.5** If at time \( t \) we give away a seat to a class \( i \) customer, so do we at time \( t - 1 \), \( n \) being the same.

**Proof.** Suppose at time \( t \), we have \( f_i + R(n - 1, t - 1) \geq R(n, t - 1) \). By diminishing return,

\[
f_i \geq R(n, t - 1) - R(n - 1, t - 1) \geq R(n, t - 2) - R(n - 1, t - 2).
\]

So

\[
f_i + R(n - 1, t - 2) \geq R(n, t - 2).
\]

Intuitively, the less time we have the less likely we will be able to fill \( n \) seats with passengers of classes higher than \( i \). If we accommodate a class \( i \) request at time \( t \) because we find the prospect of filling the \( n \) seats with passengers of classes higher than \( i \) is rather bleak, that prospect will only get worse at time \( t - 1 \) and thus we might as well accommodate a class \( i \) request.

**Proposition 3.6** If for \( n \), we accommodate a class \( i \) request, we do the same for \( n + 1, t \) being the same.
Proof: We want to show that if

\[ f_i + R(n - 1, t - 1) \geq R(n, t - 1), \]

then

\[ f_i + R(n, t - 1) \geq R(n + 1, t - 1). \]

For the purposes of deriving a contradiction, if \( f_i + R(n, t - 1) < R(n + 1, t - 1) \), then by concavity

\[ f_i < R(n + 1, t - 1) - R(n, t - 1) \leq R(n, t - 1) - R(n - 1, t - 1) \leq f_i. \]

Intuitively, if we accommodate a class \( i \) request with \( n \) seats available because we find the prospect of filling all \( n \) seats with passengers of classes higher than \( i \) is rather bleak, that prospect will only get worse with \( n + 1 \) seats available and with the same amount of time and thus we might as well accommodate a class \( i \) request.

Proposition 3.5 guarantees the uniqueness of the threshold time and thus makes the threshold time well-defined. Furthermore, with the threshold time the computation is greatly simplified. Without the threshold time, we will need to figure out individual decisions for all \( t \), given \( n \), the number of seats available. That is, given \( n \), for any \( t \), we need to compute and store the value of a binary variable indicating whether a request occurring at \( t \) will be accepted or not. With the threshold time, all we need to do is to compute the threshold time. If a request occurs before the threshold time of its class, it is rejected. Otherwise it is accepted.

The next proposition states that threshold times are non-increasing in class, i.e., the higher the fare, the larger the threshold time.

**Proposition 3.7** If at time \( t \) with \( n \) seats available, we accommodate a class \( i \) request, we will also accommodate any request of class higher than \( i \).
Chapter 3. Single-Leg Dynamic Models

Proof: By definition,

\[ f_i + R(n - 1, t_n^i) - 1 \geq R(n, t_n^i - 1). \]  \hspace{1cm} (3.3)
\[ f_i + R(n - 1, t_n^i) < R(n, t_n^i). \]  \hspace{1cm} (3.4)

For any class \( k \) such that \( f_k \geq f_i \), we have

\[ f_k + R(n - 1, t_n^i - 1) \geq R(n, t_n^i - 1), \]

which means at time \( t_n^i \) we should also accommodate a class \( k \) request.

3.3.2 Algorithm for Computing Threshold Times

In the following algorithm, the binary array \( \text{found}(n, i) \) indicates whether class \( i \) threshold time given \( n \) seats remaining has been found. It takes value \( 0 \) if the threshold time has not been found; \( 1 \) otherwise.

Algorithm 3.1 (Threshold Time Algorithm)

BEGIN

Initialization:

1. For \( t = 0 \) to \( T \), do:
   \[ R(0, t) = 0 \]

2. For \( n = 0 \) to \( C \), do:
   \[ R(n, 0) = 0 \]
   For \( i = 1 \) to \( K \), do:
   \[ \text{found}(n, i) = 0 \]

Iteration:

For \( n = 1 \) to \( C \), do:

For \( t = 1 \) to \( T \), do:
\[ R(n, t) = R(n, t - 1) \]
For $i = 1$ to $K$, do:

if $\text{found}(n, i) = 0$ then

if $f_i + R(n - 1, t - 1) \geq R(n, t - 1)$ then

$R(n, t) = p_i \cdot [f_i + R(n - 1, t - 1)] + (1 - p_i)R(n, t - 1)$

threshold$(n, i) = t$

else

$\text{found}(n, i) = 1$

endif

else

break (out of the For $i$ loop)

endif

END

Notice that we are able to break out of the innermost loop when $\text{found}(n, i)$ is equal to 1 because by Proposition 3.7 if a class is closed so must be any class lower than it.

**Computational Complexity**

Clearly in the worst case, the running time is $O(C \cdot T \cdot K)$.

### 3.4 A Dynamic Model with Cancelations and Overbookings

In this section we extend the dynamic model of the previous section to cases where cancelations and overbookings are allowed.

As before, we partition the time horizon into time intervals. Arrivals occur at the beginning of intervals, and there can be at most one arrival for each interval, with $p_i$ being the probability that the arrival is of class $i$, where $i = 1, \ldots, K$. As before the decision to be taken is whether the incoming request will be accepted or not.
Chapter 3. Single-Leg Dynamic Models

The complication we address here is the existence of cancelations. As is the case with arrivals, there can be at most one cancelation for each class, i.e., there can be multiple cancelations, but at most one for each class. Also the arrival that occurred at the beginning of an interval will not cancel in the same interval.

We assume that reservations of any class are memoryless in terms of their cancelation behavior. Whether a reservation of a certain class will cancel itself at time $t$ is independent of its past history. Also within any class, reservations are independent of each other and identical in terms of their cancelation behavior. Let $N_i$ be the number of class $i$ requests that are canceled within an interval. Then the probability that there is one class $i$ cancelation is

$$P(N_i = 1|N_i \leq 1) = \frac{n_i r_i}{n_i r_i + 1 - r_i}$$ (3.5)

where $n_i$ is the total number of class $i$ requests accepted and $r_i$ is the probability that a reservation will be cancelled in a time interval.

To buffer possible cancelations airlines usually overbook their flights. We introduce some additional notation in order to study the effects of cancelations and overbookings.

- $w$: penalty for one overbooking. We assume the penalty is the same across all classes.

- $p_i$: probability that an arrival within an interval is of class $i$, $i = 1, \ldots, K$. Thus, $1 - \sum_{i=1}^{K} p_i$ is the probability that no arrival occur in the interval.

- $R(n_1, n_2, \ldots, n_K, C, t)$: expected revenue at time $t$ (which is also the time remaining till departure) with $n_i$ seats booked for class $i$, $i = 1, 2, \ldots, K$.

We next derive equations that $R(n_1, \ldots, n_K, C, t)$ satisfies. First we introduce the
Chapter 3. Single-Leg Dynamic Models

boundary conditions.

\[
R(n_1, n_2, \ldots, n_K, C, 0) = w \cdot \min\{0, C - \sum_{i=1}^{K} n_i\} \\
R(0, 0, \ldots, 0, C, t) = 0, \ \forall t
\]

(3.6)

Notice that the overbooking penalty is incorporated in \( R(n_1, n_2, \ldots, n_K, C, 0) \).

Let \( X_i \) be a random variable:

\[
X_i = \begin{cases} 
1 & \text{if there is a cancelation from class } i \\
0 & \text{otherwise}
\end{cases}
\]

Note that

\[
P(X_i = 1) = \frac{n_i r_i}{n_i r_i + 1 - r_i}
\]

The dynamic programming recursion is as follows:

\[
R(n_1, n_2, \ldots, n_K, C, t + 1) = \\
\sum_{i=1}^{K} p_i \cdot \max_{y=0,1} \left\{ x_1, \ldots, x_K \right\} \mathbb{E} \left[ y f_i - \sum_{i=1}^{K} f_i X_i + R(n_1 - X_1, \ldots, n_i + y - X_i, \ldots, n_K - X_K, C, t) \right] \\
+ (1 - \sum_{i=1}^{K} p_i) \cdot \mathbb{E} \left[ - \sum_{i=1}^{K} f_i X_i + R(n_1 - X_1, \ldots, n_K - X_K, C, t) \right]
\]

(3.7)

A natural question is whether the notion of the threshold time established for the case without cancelations generalizes to the case with cancelations. We demonstrate through a counter example that a threshold time is not well-defined any more in the case of cancelations and overbookings.

Consider the following 2-class example where the capacity is 30, and the penalty cost is $300.00, as shown in Table 3.1. The algorithm defined by Eqs. (3.7) and (3.6) is applied to the example. The output that we are interested in is the decisions on requests at different times given that the capacity is 30 and that 16 and 15 are reserved for classes 1 and 2, respectively. The decisions are shown in Table 3.2.

If a threshold time exists for class 1, then if at time \( t + 1 \) we accept a class 1
Table 3.1: A counter-example: 2 classes, 30 capacity, $300 penalty

<table>
<thead>
<tr>
<th></th>
<th>Arr. Rate ($p_i$)</th>
<th>Can. Rate ($r_i$)</th>
<th>Fare ($f_i$)</th>
<th>Seats Sold ($n_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>0.4</td>
<td>0.01</td>
<td>200.0</td>
<td>16</td>
</tr>
<tr>
<td>Class 2</td>
<td>0.5</td>
<td>0.01</td>
<td>100.0</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 3.2: Decisions on Class 1

<table>
<thead>
<tr>
<th>Time</th>
<th>Decision</th>
<th>Time</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>reject</td>
<td>11</td>
<td>accept</td>
</tr>
<tr>
<td>2</td>
<td>reject</td>
<td>12</td>
<td>accept</td>
</tr>
<tr>
<td>3</td>
<td>reject</td>
<td>13</td>
<td>accept</td>
</tr>
<tr>
<td>4</td>
<td>reject</td>
<td>14</td>
<td>accept</td>
</tr>
<tr>
<td>5</td>
<td>reject</td>
<td>15</td>
<td>accept</td>
</tr>
<tr>
<td>6</td>
<td>reject</td>
<td>16</td>
<td>accept</td>
</tr>
<tr>
<td>7</td>
<td>reject</td>
<td>17</td>
<td>accept</td>
</tr>
<tr>
<td>8</td>
<td>reject</td>
<td>18</td>
<td>accept</td>
</tr>
<tr>
<td>9</td>
<td>accept</td>
<td>19</td>
<td>accept</td>
</tr>
<tr>
<td>10</td>
<td>accept</td>
<td>20</td>
<td>accept</td>
</tr>
</tbody>
</table>
request we should accept one at time \( t \). The output of this counter-example clearly shows that is not the case. Up until time 9, we will accept class 1 requests and we reject them beginning at time 8.

In this particular case, the flight is overbooked (by 1 seat) with 16 and 15 reservations for classes 1 and 2, respectively. When we are relatively distant from the departure time, we still accept class 1 requests in the hope that there will be cancellations from class 2. The net result will then be that the overbookings will be eliminated by class 2 cancellations and the revenue will be increased by filling in with class 1 request the vacancies created by the class 2 cancellations. Essentially this is a substitution process. This substitution mechanism does not exist in the case where there are no cancellations.

As we approach the departure time, we will eventually stop accepting class 1 requests because there will not be enough time for enough class 2 cancellations to occur to eliminate the overbookings.

We take a small digression and take a broader look at the general booking behavior of class 1. Early in the booking process when there are sufficient remaining capacity, we accept class 1 requests to utilize the remaining capacity. When the flight is mildly overbooked but still far from the departure time, we continue to accept class 1 requests because of the substitution mechanism mentioned earlier. Finally when the flight is heavily overbooked or we are too close to the departure time, we stop accepting class 1 requests at all.

The conclusion we draw is that we cannot define the threshold time any more when we allow cancellations and overbookings. However, we do have the following conjecture. In place of the threshold time, we have the threshold window. By the threshold window of a class, we mean a period of continual time intervals, within which requests for the class are accepted and outside which requests for the class are rejected. For example, if the threshold window of a class is time 35 through time 10, then only requests for the class that occur between time 35 and time 10 will be
accepted. The threshold window can be closed at both ends or open at either end of
of the booking horizon as illustrated in Figure 3-2.

Based on numerical experimentation, we believe (but have not formally proved)
that the following properties hold.

1. The revenue function $R(n_1, n_2, \ldots, n_K, C, t)$ is a concave function in $C$. In
other words, the expected marginal revenue brought by an additional seat is
non-increasing.

2. If the decision is to accept a class $i$ request when the capacity is $C$, then the
same decision applies when we have $C + 1$ seats, everything else being the same.

3. If the decision is to accept a class $i$ request when the state is $(n_1, n_2, \ldots, n_i +
1, \ldots, n_K, C, t)$, the same decision applies at state $(n_1, n_2, \ldots, n_i, \ldots, n_K, C, t)$. 

Figure 3-2: Possible scenarios of the threshold window
3.5 An Alternative Dynamic Model with Cancelations and Overbookings

The model of the previous section clearly has dimensionality difficulties. For this reason, we propose another approach, in which we are not concerned with the timing of cancelations but rather assume that all cancelations occur in the very end. We now have the following recursion.

\[
R(n_1, n_2, \ldots, n_K, C, t) = \sum_{i=1}^{K} p_i \max_{y=0,1} \{ r_i f_i + R(n_1, n_2, \ldots, n_i + y, \ldots, n_K, C, t - 1) \} + (1 - \sum_{i=1}^{K} p_i) R(n_1, n_2, \ldots, n_i, \ldots, n_K, C, t - 1) \tag{3.8}
\]

where \(n_1, n_2, \ldots, n_K\) are reservations made for classes 1 through \(K\), respectively. Cancelations are captured in the boundary condition:

\[
R(n_1, n_2, \ldots, n_K, C, 0) = w \cdot E[\min(0, C - \sum_{i=1}^{K} \sum_{j=1}^{n_i} X_{ij})] \tag{3.9}
\]

where \(X_{ij}\) are Bernoulli random variables:

\[
X_{ij} = \begin{cases} 
1 & \text{with probability } r_i \\
0 & \text{with probability } 1-r_i 
\end{cases} \quad \text{for } i = 1, \ldots, K \text{ and } j = 1, \ldots, n_i
\]

Again, \(r_i\) is the show-up rate for class \(i\).

3.5.1 An Approximation

As we did before in Chapter 2 in approximating the penalty cost, we approximate the boundary condition in the following way:

\[
\tilde{R}(n_1, n_2, \ldots, n_K, C, 0) = \min(0, C - \sum_{i=1}^{K} r_i n_i). \tag{3.10}
\]
In doing so, we can see that $\tilde{R}(n_1, n_2, \ldots, n_K, C, 0)$ can be rewritten as $\tilde{R}(C - \sum_{i=1}^K r_in_i, 0)$. In other words, the approximated boundary condition is a function of expected surplus capacity, i.e., the capacity minus the sum of expected materialized reservations. This can be generalized to any $t$, i.e., for any $t$ the revenue can be written as $\tilde{R}(x, t)$ where $x$ is the expected surplus capacity. With the approximation, recursion (3.8) is simplified as:

$$
\tilde{R}(x, t) = \sum_{i=1}^K p_i \cdot \max_{y=0,1}\{f_ir_iy + \tilde{R}(x - r_iy, t - 1)\} + (1 - \sum_{i=1}^K p_i) \cdot \tilde{R}(x, t - 1) \quad (3.11)
$$

In what follows, we show that if $r_i = r$ for all $i$, i.e., the show-up rates are identical among all fare classes, we can establish some results similar to those in cases where no cancellations are allowed, such as threshold times. After establishing the necessary properties we introduce an algorithm for finding the threshold times.

**Proposition 3.8** $\tilde{R}(x, t)$ is non-decreasing in $x$.

**Proof:** We use induction on $t$.

For $t = 0$, the proposition obviously holds.

Suppose $\tilde{R}(x + z, t - 1) \geq R(x, t - 1), \forall z \geq 0$. We need to show $\tilde{R}(x + z, t) \geq \tilde{R}(x, t)$, $\forall z \geq 0$.

Now

$$
\tilde{R}(x + z, t) = \sum_{i=1}^K p_i \cdot \max_{y=0,1}\{f_ir_iy + \tilde{R}(x + z - r_iy, t - 1)\} + (1 - \sum_{i=1}^K p_i) \cdot \tilde{R}(x + z, t - 1)
$$

$$
\geq \sum_{i=1}^K p_i \cdot \max_{y=0,1}\{f_ir_iy + \tilde{R}(x + z - r_iy, t - 1)\} + (1 - \sum_{i=1}^K p_i) \cdot \tilde{R}(x, t - 1).
$$

We only need to show

$$
\max_{y=0,1}\{f_ir_iy + \tilde{R}(x + z - r_iy, t - 1)\} \geq \max_{y=0,1}\{f_ir_iy + \tilde{R}(x - r_iy, t - 1)\}.
$$

In general, if $u \geq s$ and $v \geq t$, then $\max(u, v) \geq \max(s, t)$. 

This implies
\[
\max_{y=0,1} \{ f_i r y + \tilde{R}(x + z - ry, t - 1) \} \geq \max_{y=0,1} \{ f_i r y + \tilde{R}(x - ry, t - 1) \}.
\]

Thus
\[
\tilde{R}(x + z, t) \geq \sum_{i=1}^{K} p_i \max_{y=0,1} \{ f_i r y + \tilde{R}(x - ry, t - 1) \} + (1 - \sum_{i=1}^{K} p_i) \cdot \tilde{R}(x, t - 1)
\]
\[= \tilde{R}(x, t)
\]

**Proposition 3.9** \( \tilde{R}(x, t) \) is non-decreasing in \( t \).

**Proof:** We have
\[
\tilde{R}(x, t) = \sum_{i=1}^{K} p_i \cdot \max_{y=0,1} \{ f_i r y + \tilde{R}(x - ry, t - 1) \} + (1 - \sum_{i=1}^{K} p_i) \tilde{R}(x, t - 1)
\]
But
\[
\max_{y=0,1} \{ f_i r_i y + \tilde{R}(x - ry, t - 1) \} \geq \tilde{R}(x, t - 1).
\]
Therefore,
\[
\tilde{R}(x, t) \geq \sum_{i=1}^{K} p_i \cdot \tilde{R}(x, t - 1) + (1 - \sum_{i=1}^{K} p_i) \cdot \tilde{R}(x, t - 1) = \tilde{R}(x, t - 1).
\]

**Proposition 3.10** \( \tilde{R}(x + r, t) + \tilde{R}(x - r, t) \leq 2\tilde{R}(x, t) \).

**Proof:** Let \( x' = \frac{x}{r} \). Then
\[
\tilde{R}(x, t) = \tilde{R}(r x', t) \equiv \tilde{R}(x', t)
\]
The boundary condition becomes:

$$\tilde{R}(x', 0) = w \cdot r \cdot \min(0, x')$$

For any $t$ we have

$$\tilde{R}(x', t) = \tilde{R}(rx', t)$$
$$= \sum_{i=1}^{K} p_i \cdot \max_{y=0,1} \{ f_i r y + \tilde{R}(rx' - ry, t - 1) \} + (1 - \sum_{i=1}^{K} p_i) \cdot \tilde{R}(rx', t - 1)$$

By Proposition (3.3),

$$\tilde{R}(x' + 1, t) + \tilde{R}(x' - 1, t) \leq 2\tilde{R}(x' - 1, t)$$

Equivalently,

$$\tilde{R}(x + r, t) + \tilde{R}(x - r, t) \leq 2\tilde{R}(x, t)$$

**Proposition 3.11** \( \tilde{R}(x, t) - \tilde{R}(x - r, t) \) is non-decreasing in $t$.

**Proof:** Define \( \tilde{R}(x, t) \equiv \tilde{R}(rx, t) \). By Proposition 3.4,

$$\tilde{R}(\frac{x}{r}, t) - \tilde{R}(\frac{x}{r} - 1, t)$$

is non-decreasing in $t$. Equivalently,

$$\tilde{R}(x, t) - \tilde{R}(x - r, t)$$

is non-decreasing in $t$.
Proposition 3.12 If at time $t$ we give away a seat to a class $i$ customer, so do we at time $t - 1$, $x$ being the same.

Proof: Suppose at time $t$, we have

$$f_i r + \tilde{R}(x - r, t - 1) \geq \tilde{R}(x, t - 1)$$

By Proposition 3.11, we have:

$$f_i r \geq \tilde{R}(x, t - 1) - \tilde{R}(x - r, t - 1) \geq \tilde{R}(x, t - 2) - \tilde{R}(x - r, t - 2)$$

i.e.,

$$f_i r + \tilde{R}(x - r, t - 2) \geq \tilde{R}(x, t - 2)$$

Proposition 3.12 guarantees the existence of the threshold time.

3.5.2 Algorithm for Computing Threshold Times

The algorithm for computing threshold times we introduce below is very similar to the one in Section 3.3.2. In this case we need to scale the capacity by the show-up rate $r$. Also we will assume that $r$ has no more than 2 decimal digits.

In the following algorithm, the binary array found($n, i$) indicates whether class $i$ threshold time given $n$ seats remaining has been found. It takes value 0 if the threshold time has not been found; 1 otherwise.

Algorithm 3.2 (Approximate Threshold Time Algorithm)
BEGIN
Initialization:
1. For $t = 0$ to $T$, do:
   $\tilde{R}(0, t) = 0.0$
2. For $x = 0$ to $\left\lfloor \frac{C}{r} \right\rfloor$, do:

   $\tilde{R}(x, 0) = 0.0$

   For $i = 1$ to $K$, do:

   $\text{found}(x, i) = 0$

**Iteration:**

For $x = 1$ to $\left\lfloor \frac{C}{r} \right\rfloor$, do:

   For $t = 1$ to $T$, do:

   $\tilde{R}(x, t) = \tilde{R}(x, t - 1)$

   For $i = 1$ to $K$, do:

   if $\text{found}(x, i) = 0$ then

   if $f_i + \tilde{R}(x - 1, t - 1) \geq \tilde{R}(x, t - 1)$ then

   $\tilde{R}(x, t) = p_i \cdot [f_i + \tilde{R}(x - 1, t - 1)] + (1 - p_i)\tilde{R}(x, t - 1)$

   $\text{threshold}(x, i) = t$

   else

   $\text{found}(x, i) = 1$

   endif

   else

   break (out of the For $i$ loop)

   endif

END

3.5.3 Computational Complexity

Clearly in the worst case, the running time is $O\left(\frac{C}{r} \cdot T \cdot K\right)$. Since we assume that $r$ has at most two decimal digits, $\frac{1}{r} \leq 160$. So the running time can be written as $O(C \cdot T \cdot K)$. 
Chapter 4

Static Network Models

4.1 Introduction

Modern airlines operate over a large network of flights. It is commonplace to find passengers aboard the same aircraft originated from different locations and headed for different destinations. Airlines typically adopt a "hub-and-spoke" system where one or several airports are designated as "hubs". If a passenger wants to travel from point A to point B, typically he or she will fly inbound to one of the hubs from point A and then fly outbound (possibly aboard a different flight) to point B. Obviously this passenger can make the trip from point A to point to B only if he or she can be accommodated on both the inbound flight from point A into the hub and the outbound flight from the hub to point B. Competition for seats exists not only among passengers of identical itineraries who pay different fares, but also among passengers of different itineraries. Figure 4-1 provides an illustration of the hub-and-spoke system.

Within a hub-and-spoke system, there are three general types of itineraries:

1. Itineraries that begin at an origin and end at the hub;
2. Itineraries that begin at the hub and end at a destination;
3. Connecting itineraries that begin at an origin, make a stop at the hub and end
at a destination;

We are ignoring itineraries with more than two legs. Clearly, network effects are important in the practice of airline yield management.

The current practice in the airline industry to handle network effect is to use a heuristic method called "virtual nesting". In virtual nesting, all fare classes of all itineraries aboard a physical flight leg are aggregated into several "fare buckets" according to their fares only, regardless of their origins and destinations. Typically fares that are in a relatively close range of each other are put into the same fare bucket. Then some single-leg model is used to determine the "virtual" booking limits for the aggregated classes, or the fare buckets. The aggregation is done on all physical flight legs in the system. When a request is made for a certain fare class on a certain itinerary, the following will happen. Each flight leg of every itinerary is examined. For each flight leg, its corresponding fare bucket is determined. Then the virtual booking limit on the corresponding fare bucket is checked to decide if the request should be accepted or denied by this particular leg. The request will be accepted only if it is accepted by all the flight legs of the itinerary.

Let us consider the example in Figure 4-2, where there are 3 itineraries: Boston
to Chicago, Chicago to Los Angeles and Boston to Los Angeles via Chicago. Within each itinerary there are two classes, classes 1 and 2, with class 1 being the higher class. The lower and higher fares from Boston to Chicago are $150.00 and $300.00, respectively. The lower and higher fares from Chicago to Los Angeles are $150.00 and $300.00, respectively. Finally, the lower and higher fares from Boston to Los Angeles are $300.00 and $500.00, respectively.

Aboard the leg from Boston to Chicago, 3 fare buckets are created. The lower fare of the Boston-Chicago itinerary by itself is fare bucket 1. The higher fare of the Boston-Chicago itinerary and the lower fare of the Boston-Los Angeles itinerary make up fare bucket 2. The higher fare of the Boston-Los Angeles itinerary by itself stands for fare bucket 3. Aboard the leg from Chicago to Los Angeles, 3 fare buckets are created. The lower fare of the Chicago-Los Angeles itinerary by itself is fare bucket 1. The higher fare of the Chicago-Los Angeles itinerary and the lower fare of the Boston-Los Angeles itinerary make up fare bucket 2. The higher fare of the Boston-Los Angeles itinerary by itself stands for fare bucket 3. See Figure 4-3 for illustration of the virtual fare buckets.

After virtual fare buckets are created, booking limits are then calculated for the 3 fare buckets. We illustrate how fare requests are handled using a few examples. Suppose a request for the lower fare of the Boston-Chicago itinerary is made. The lower fare of the Boston-Chicago itinerary falls into virtual fare bucket 3 aboard the Boston-Chicago leg. The booking limit on virtual fare bucket 3 aboard the Boston-Chicago leg is checked to decide whether to accept or reject the request. If a request for the higher fare of the Boston-Chicago itinerary, we first find the virtual fare bucket it belongs to. In this case the fare bucket is fare bucket 2 on the Boston-Chicago flight leg. Then the booking limit of the fare bucket is checked to decide whether to accept the request or not. Finally suppose a request is made for the lower fare of the Boston-Los Angeles itinerary. Since the itinerary traverses both the Boston-Chicago leg and the Chicago-Los Angeles leg, we need to find the virtual fare buckets it belongs to.
Figure 4-2: A simple virtual nesting example

(1) Virtual fare buckets aboard the Boston-Chicago leg

(2) Virtual fare buckets aboard the Chicago-Los Angeles leg

Figure 4-3: Creation of virtual fare buckets

on both legs. In this case, they are virtual fare bucket 2 on both the Boston-Chicago leg and the Chicago-Los Angeles leg. So we check the booking limits on both virtual fare buckets 3. Only if the request is accepted by both virtual fare buckets will the request be accepted for the itinerary.

A recent development in network models is the so-called “bid price” approach. One such model [17] can be stated as follows. Let the triplet $ijk$ be a itinerary-class combination where $ij$ represents the itinerary from origin $i$ to destination $j$ and $k$ represents the fare class. For each itinerary-class combination $ijk$, define the expected marginal revenue (EMR) of potentially selling each seat $s$ to $ijk$ as:

$$EMR_{ijk}(s) = f_{ijk} \cdot P_{ijk}(s)$$
where \( f_{ijk} \) is the fare of itinerary-class combination \( ijk \) and \( \tilde{P}_{ijk} \) is the probability of selling the \( s \)-th seat to itinerary-class combination \( ijk \), or equivalently, the probability of having \( s \) or more requests for itinerary-class combination \( ijk \). For each itinerary \( ij \), it can possibly traverse multiple legs, each of which has a capacity. Let \( N_{ij} \) be the capacity of itinerary \( ij \). Apparently \( N_{ij} \) is simply the smallest of the capacities of all the legs that itinerary \( ij \) traverses. Let \( C_l \) be the capacity of leg \( l \). The model formally stated is:

\[
\max \sum_{ijk} \sum_{s=1}^{N_{ij}} EMR_{ijk}(s) \cdot x_{s,ijk}
\]

subject to:

\[
\sum_{\{ijk : ij \in I_l\}} \sum_{s=1}^{N_{ij}} x_{s,ijk} \leq C_l \quad \text{where } I_l = \{ij : ij \text{ traverses leg } l\}, \text{ for all } l
\]

\[
x_{s,ijk} \leq 1 \quad \text{for all } ijk \text{ and } s = 1, 2, \ldots, N_{ij}
\]

where \( x_{s,ijk} \) is a binary decision variable:

\[
x_{s,ijk} = \begin{cases} 
1 & \text{if at least } s \text{ seats are allocated to itinerary-class combination } ijk \\
0 & \text{otherwise}
\end{cases}
\]

The objective function in the bid price model is a discrete linearization of the EMSR model. It is claimed that optimal \( x_{s,ijk} \)'s are integer. Associated with each capacity constraint for each leg \( l \), there is an optimal dual solution (multiplier). This multiplier is called the "bid price" of leg \( l \). The proposed control is that a seat on leg \( l \) is sold to a passenger only if he or she is willing to pay at least the bid price of the leg.

The bid price approach has an element of arbitrariness in it in that bid prices in this context are simply the dual optimal solutions corresponding to constraints. In this sense one can come up with any mathematical programming model and use the dual solutions as bid prices.

Another concern with the bid price approach is that it seems to suggest that fares
Chapter 4. Static Network Models

should be negotiable. Moreover, bid prices will be updated as a booking process progresses, which implies the fares will fluctuate even within one fare class. It is not clear if this is implementable.

There has been preliminary work on developing models that combine deterministic network models and single-leg stochastic models [6]. In this chapter we will consider the "hub-and-spoke" type of networks and develop a hierarchical approach that incorporates both network flow algorithms and single-leg nesting approaches.

4.2 A Network Model for Stochastic Demands

In the single-leg case, the standard approach for handling stochastic demands is nesting approaches. One could potentially propose a global nesting approach for the network case, i.e., take all the classes of all the itineraries and nest them according to their fare value just as in the single-leg case. There are conceptual (as well as implementation) difficulties with such an approach. The nesting approach relies on the assumption that lower fare classes book up before higher fare classes. While the assumption often holds within an itinerary, it is not clear why it will hold across itineraries. Consider a flight from Boston to Los Angeles via Chicago, one can argue that among local passengers who travel from Boston to Chicago passengers who fly lower fare classes make reservations earlier than passengers who fly higher classes. The same behavior can be argued for local customers who travel from Chicago to Los Angeles. But there is no ground for making any claim about the order of bookings between the Boston-Chicago passengers and the Chicago-Los Angeles passengers. In fact it is reasonable to think of any particular class on both the Boston-Chicago leg and the Chicago-Los Angeles leg as booking up in about the same time frame.

This difficulty motivates the following hierarchical approach we would like to propose. At a higher level, flight capacities of a hub-and-spoke system are divided up among all itineraries. We will show later that this indeed can be transformed into a
convex network flow problem. At the lower level, within each itinerary we take the allocated capacity and decide how to optimally set booking limits on different classes using a nesting approach.

For simplicity, we assume that there are exactly $n$ aircraft, with aircraft $i$ flying from origin $i$ to destination $i$ via the hub $(i = 1, \ldots, n)$. Aboard any aircraft can be customers of as many as $n + 1$ different itineraries. Within all itineraries, we assume there are identically $K$ fare classes.

We denote the hub as origin/destination 0. Let:

$C_i$: capacity of aircraft $i$, $i = 1, \ldots, n$. This is data.

$X_{ij}$: capacity allocated to itinerary $ij$, $i = 0, \ldots, n$; $j = 0, \ldots, n$ and $i + j \neq 0$. This is the decision variable at the higher level.

$S_{ijk}(X_{ij})$: booking limit on class $k$ of itinerary $ij$, $i = 0, \ldots, n$; $j = 0, \ldots, n$, $i + j \neq 0$ and $k = 1, \ldots, K$. This is the decision variable at the lower level.

At the higher level, the optimization problem is:

$$\max \sum_{i+j \neq 0} R_{ij}(X_{ij})$$

$$\text{s.t.} \sum_{j=0}^{n} X_{ij} \leq C_i, \ i = 1, \ldots, n \quad (4.1)$$

$$\sum_{i=0}^{n} X_{ij} \leq C_j, \ j = 1, \ldots, n$$

where $R_{ij}(X_{ij})$ is the expected revenue from itinerary $ij$ when $X_{ij}$ seats are allocated to it. Typically, $R_{ij}(\cdot)$ does not have a closed form. Later we will discuss how to compute $R_{ij}(X_{ij})$.

The above problem can be transformed into a transportation problem. As shown in Figure 4-4, we create $n$ supply nodes each with supply $C_i$, $i = 1, \ldots, n$, respectively. These supply nodes essentially represent $n$ different origins. We create a $n+1$st supply node, which we will call a dummy supply node and which represents the hub. On
the right, we create \( n \) demand nodes each with demand \( C_i, i = 1, \ldots, n \), respectively. These demand nodes essentially represent \( n \) different destinations. We create a \( n+1 \)st node, which we will call a dummy demand node and which also represents the hub. We link every supply node to each every one of the demand nodes and thus create a fully connected bipartite graph. The revenue of arc \( ij \) is \( R_{ij}(. \cdot) \). The revenue of arc from supply node \( n + 1 \) to demand node \( n + 1 \) is 0. The original problem is now reduced to a non-linear max-revenue flow problem on this bipartite network.

Figure 4-4: Transformation of the hub-and-spoke system into a transportation network
4.2.1 Revenue Function $R_{ij}(\cdot)$

In Chapter 2, we proved the wait-and-see approach revenue function is concave. In this chapter we utilize this property. At the lower level, we will use the wait-and-see approach to handle the randomness within each individual itinerary and to generate the function value needed for the higher level optimization. To solve the higher level optimization problem, we need to know $R_{ij}(x_{ij})$ for given $x_{ij}$. Recall in Chapter 2 the improved wait-and-see algorithm (see Algorithm 2.3) generates the expected revenue given the number of seats available. In other words, $R_{ij}(\cdot)$ are evaluated at integer points. However, the concavity of each $R_{ij}(\cdot)$ implies that the objective function of optimization problem (4.1) is concave since it is the sum of all $R_{ij}(\cdot)$ over all $ij$. Thus optimization problem (4.1) is a concave max-cost problem with integer optimal solutions that can be found in polynomial time (see pages 556–560 of [1]).

4.2.2 A Hierarchical Algorithm

In this section we describe in more detail the concave flow algorithm. The input to the algorithm includes:

- flight network information, i.e., number of origins, number of destinations, network topology
- capacity of each flight in the network
- information of each individual itinerary, including fares and demand distributions

There are two major components to the hierarchical algorithm:

1. Construct revenue function for all itineraries.

2. Take the revenue functions for all itineraries, run the concave flow algorithm that determines the optimal amount of capacity allocated to each itinerary.
3. For each itinerary, take the capacity assigned to it in step 2 and determine the booking limits using the wait-and-see approach of Chapter 2.

In constructing the revenue function for each itinerary, the wait-and-see algorithm developed in Chapter 2 is called as a subroutine. Without loss of generality, we assume that the capacity is identically $C$ among all physical flights and the number of fare classes is identically $n$ among all itineraries. The wait-and-see algorithm will compute $R_{ij}(X_{ij})$, the revenue for itinerary $ij$ when $X_{ij}$ seats are assigned to it, for all $ij$ (i.e., all itineraries) and for $X_{ij} = 0, 1, \ldots, C$. Simultaneously, optimal booking limits are computed for each level of $X_{ij}$. For each itinerary $ij$, its revenues are stored in an array of size $C + 1$. For each level of $X_{ij}$, an array of size $n$ is created to store the optimal booking limits. We showed in Chapter 2 that $R_{ij}(\cdot)$ is a concave function. What each of the arrays of size $C + 1$ holds is essentially a function evaluated at integer points at which the function are concave.

In the network optimization stage, the aforementioned concave flow algorithm (see pages 556–560 of [1]) will be applied to determine optimal capacities $X_{ij}$ assigned to all $ij$ itineraries such that the sum of all $R_{ij}(X_{ij})$ are maximized subject to constraints that for each physical flight the sum of capacities allocated to the itineraries on board the flight must not exceed its cabin capacity, i.e., the concave flow algorithm will solve optimization problem (4.1). Each time an evaluation of the objective function is needed, the arrays filled by the wait-and-see algorithm for all $ij$ will be visited.

In the final stage, optimal $X_{ij}$'s, for all itinerary $ij$, are taken back to each itinerary. For each itinerary $ij$, the wait-and-see approach determines $S_{ijk}$, the booking limits on class $k$, for all $k$'s, based on $X_{ij}$.

Figure 4-5 provides a graphic illustration of the hierarchical approach.

The hierarchical algorithm can be formally stated as follows.

**Algorithm 4.1 (Divide-and-Nest Algorithm)**

BEGIN

**Computing revenue function:**
Figure 4-5: Different stages of the hierarchical algorithm

For all $ij$, do:

For all $s = 0, 1, \ldots, \min(C_i, C_j)$, do:

Compute and store all $R_{ij}(s)$ using the improved wait-and-see algorithm

**Allocating capacities:** call the aforementioned concave flow algorithm (see pages 556–560 of [1]) to compute $X_{ij}$, optimal capacity allocated to itinerary $ij$, for all $ij$.

**Computing booking limits:**

For all $ij$, do:

Based on $X_{ij}$, use the improved wait-and-see algorithm to obtain $S_{ijk}$, the optimal booking

limit on class $k$ of itinerary $ij$, for all $k$.

END

The complexity of this algorithm is the running time for computing the revenue functions and the running time of the concave flow algorithm. We do not count the running time for computing the booking limits because it is included in the running time of the revenue functions (this is the nature of the wait-and-see approach). The
running time for computing all the revenue functions is

\[ n(n + 2) \cdot O(KC^3) = O(n^2KC^3) \]

where \( C = \max_i C_i \). The running time for the concave flow algorithm in general is \( O((m \log U)T(m, n)) \), where \( U \) is largest possible capacity, \( m \) is the number of arcs in the network and \( n \) is the number of nodes in the network, and \( T(\cdot) \) is the polynomial running time for solving a shortest path problem. In our case it the number of arcs is equal to the number of itineraries, which is \( n(n + 2) \). The number of nodes is simply the total number of origins and destinations, which is \( 2n + 1 \). is \( O(n(n + 2) \log C)T(2n + 1, n(n + 2)) \), or equivalently \( O(n^2 \log C)T(n, n^2) \). The total running time for the hierarchical algorithm is then

\[ O(n^2KC^3) + O(n^2 \log C)T(n, n^2) \]

As we can see the bottleneck is the computation of revenue functions.

### 4.2.3 Computational Issues

A significant part of computation in the above hierarchical algorithm is spent on constructing the revenue functions. The revenue functions as well as optimal booking limits need not be calculated on-line each time the algorithm is run. They can be calculated off-line and stored \textit{a priori}. 

.
Chapter 5

Dynamic Network Models

5.1 Introduction

In this chapter, we introduce dynamic network models and propose two general approaches for the problem. The first one is a hierarchical approach introduced in the last chapter, which philosophically keeps the dynamic element local to individual itineraries and use network flow algorithms to capture the interaction between itineraries. Within each itinerary we use different single-leg dynamic models developed in Chapter 3. The second approach uses global dynamic programming and uses quadratic approximations to overcome the rather severe dimensionality problem.

5.2 A Hierarchical Dynamic Model

We propose for the hub-and-spoke system a hierarchical approach similar to that in Chapter 4. At the higher level we allocate the flight capacities over the network among all the itineraries. At the lower level, within an itinerary we take the allocated capacity and use a single-leg dynamic model from Chapter 3 to conduct seat inventory control.

As before, the objective is to optimally allocate network capacities among all the
Chapter 5. Dynamic Network Models

itineraries such that the expected revenue over the entire network is maximized.

$$\max \sum_{i+j \neq 0} R_{ij}(S_{ij})$$

$$\text{s.t.} \sum_{j=0}^{m} S_{ij} \leq C_{i0}, \ i = 1, \ldots, m$$

$$\sum_{i=0}^{m} S_{ij} \leq C_{0j}, \ j = 1, \ldots, m.$$ (5.1)

where $R_{ij}(\cdot)$ is the revenue function for itinerary $ij$ derived from a single-leg dynamic model with or without cancelations for itinerary $ij$; $C_{i0}$ is the capacity of the flight leg from origin $i$ to the hub and $C_{0j}$ is the capacity of the flight leg from the hub to destination $j$; and $S_{ij}$ are the decision variables that specify how many seats are to be allocated to itinerary $ij$. This problem can be transformed into a network flow problem with a separable objective function. We proved in Chapter 3 that without cancelations, the single-leg revenue function is concave in capacity. For the case with cancelations we conjectured the same. The discussion below assumes that the single-leg revenue function is concave.

Without loss of generality, suppose the number of fare classes is identically $n$ among all the itineraries and the capacity is identically $C$. The algorithm for the hierarchical dynamic model can be described as follows.

1. Use a single-leg dynamic model to compute $R_{ij}(x_{ij})$ for all $ij$ and for $x_{ij} = 0, 1, \ldots, C$.

2. Apply a network concave flow algorithm to solve for optimal capacities assigned to the itineraries.

3. Take the output from step 2 and within each itinerary compute threshold times (for the case without cancelations) or threshold intervals (for the case with cancelations).
5.3 A Global Dynamic Network Model for Two Connecting Flights

We motivate our discussion with a simple network shown in Figure 5-1. In the network, there are three itineraries (from Boston to Chicago, from Chicago to Los Angeles and from Boston to Los Angeles) and two flight legs (from Boston to Chicago and from Chicago to Los Angeles). For simplicity, we denote itineraries Boston-Chicago, Chicago-Los Angeles and Boston-Los Angeles as 1, 2 and 3, respectively, and flight legs Boston-Chicago and Chicago-Los Angeles as 1 and 2, respectively.

![Figure 5-1: A simple network with 3 itineraries and 2 flight legs](image)

In the single-leg case, we need to have the bookings of all the fare classes to characterize the state of the system. In our simple network, we need to know the bookings of all the fare classes of all three itineraries. For simplicity, we assume that all three itineraries have the same number of classes \( K \). In total we have \( 3 \cdot K \) state variables. As in the single-leg case, we slice the entire booking horizon into time intervals so fine that in any interval there can be at most one arrival occurring from an itinerary-class combination.

5.3.1 Notation

We will use the following notation for the discussion of the above simple network.

- \( C_l \): leg capacity, \( l = 1, 2 \), without loss of generality \( C_1 = C_2 = C \);

- \( p_{ij} \): probability of a request of itinerary \( i \), class \( j \), \( i = 1, 2, 3 \) and \( j = 1, \ldots, K \);
• $n_{ij}$: number of reservations for itinerary $i$, class $j$, $i = 1, 2, 3$ and $j = 1, \ldots, K$;

• $X_{ij}$: $i = 1, 2, 3$ and $j = 1, \ldots, K$ and

\[ X_{ij} = \begin{cases} 1 & \text{if an itinerary } i \text{ class } j \text{ reservation is cancelled} \\ 0 & \text{otherwise} \end{cases} \]

For future convenience, let

\[ X = (X_{11}, \ldots, X_{1K}, X_{21}, \ldots, X_{2K}, X_{31}, \ldots, X_{3K}) \]

• $w_{ij}$: penalty cost for denying boarding an itinerary $i$ class $j$ reservation, $i = 1, 2, 3$ and $j = 1, \ldots, K$.

**5.3.2 Recursion and Boundary Conditions**

Similar to the recursion we have in Chapter 3, we establish the dynamic programming recursion:

\[
R(n_{11}, n_{12}, \ldots, n_{1K}, n_{21}, n_{22}, \ldots, n_{2K}, n_{31}, n_{32}, \ldots, n_{3K}, t + 1) = \\
\sum_{i=1}^{3} \sum_{j=1}^{K} p_{ij} \cdot \max_{y=0,1} \left\{ E \left[ y \cdot f_{ij} - \sum_{s=1}^{3} \sum_{t=1}^{K} f_{st} X_{st} + R(n_{11} - X_{11}, \ldots, n_{ij} + y - X_{ij}, \ldots, n_{3K} - X_{3K}, t) \right] \right\} + \\
(1 - \sum_{i=1}^{3} \sum_{j=1}^{K} p_{ij}) \cdot E \left[ - \sum_{s=1}^{3} \sum_{t=1}^{K} f_{st} X_{st} + R(n_{11} - X_{11}, \ldots, n_{3K} - X_{3K}, t) \right] \\
(5.2)
\]

Conditioned on the request, the total revenue expected from time $t + 1$ till departure (time 0) is the sum of the net revenue in time $t + 1$ (which, if the request is accepted, is the fare of the accepted request minus the cancelled revenue, and if the requested is rejected, is simply minus the expected cancelled revenue) and the total revenue expected from time $t$ to departure (time 0). If no request occurs, the total revenue expected from time $(t + 1)$ to departure (time 0) is simply the cancelled revenue plus
the revenue from time \( t \) to departure.

At departure (time 0), the revenue is the sum of the penalty costs resulting from overbookings aboard both flight legs. If overbookings do exist then we will be faced with a decision as to who to board and who to deny boarding such that the overbooking penalty is minimized.

Let \( z_{ij} \) be the decision variable that represents the number of itinerary \( i \) class \( j \) reservations to be denied boarding. \( n_{ij} - z_{ij} \) is the number of itinerary \( i \) class \( j \) reservations that are actually flown. We have the following optimization problem.

\[
\begin{align*}
\text{min} & \quad \sum_{i=1,2,3} \sum_{j=1,\ldots,K} w_{ij} z_{ij} \\
\text{s.t.} & \quad \sum_{i=1,3} z_{ij} \geq \sum_{i=1,3} n_{ij} - C \\
& \quad \sum_{i=2,3} z_{ij} \geq \sum_{i=2,3} n_{ij} - C \\
& \quad z_{ij} \leq n_{ij} \\
& \quad z_{ij} \geq 0
\end{align*}
\]

In (5.3), the objective is the sum of unit penalty penalty cost multiplied by the number of reservations that are denied boarding. The first constraint means that all overbookings for the Boston-Chicago flight leg must be denied boarding. The second constraint means that all overbookings for the Chicago-Los Angeles flight leg must be denied boarding. The third constraint means that we can deny boarding at most all the reservations of itinerary \( i \) class \( k \). Since this a minimization problem, constraints 1 and 2 will try to attain equality.

Note that the network effects are clearly captured in the boundary condition (5.3). Itinerary 3 reservations, both market and denied boarding, appear in both constraints, i.e., if a reservation is to be accommodated, it will be accommodated on both legs; if it is to be denied boarding, it will empty two seats, one on each leg.

Optimization problem (5.3) is apparently a transportation problem and can be
solved quickly.

The solution to above dynamic programming problem (5.2) and (5.3) is a policy that determines whether to accept or deny a request at any given time and state.

5.4 A Global Dynamic Model for the Hub-and-Spoke Network

In general, we consider the hub-and-spoke network discussed in Chapter 4, as shown in Figure 5-2.

![Diagram of a hub-and-spoke network]

Figure 5-2: General hub-and-spoke network

In the general hub-and-spoke network, there are \( m(m + 2) \) itineraries with \( 2m \) flight legs.

5.4.1 Notation for the General Problem

The notation for the general case is more involved than for the two-leg case discussed earlier. We need to keep track of all the itineraries, legs and classes. We use \( ij \) to represent an itinerary that starts at origin/hub \( i \) and ends at hub/destination \( j \).
Throughout we will need to have $i \cdots j \neq 0$, which means that there cannot be an itinerary that starts and ends at the hub. If neither of $i$ and $j$ are zero, we have a connecting itinerary $ij$ from origin $i$ to destination $j$ via the hub. If $i \neq 0$ and $j = 0$, this is a local itinerary that starts at origin $i$ and ends at the hub. If $i = 0$ and $j \neq 0$, this represents a local itinerary that begins at the hub and ends at origin $j$. A leg is defined specifically to be a physical flight either from an origin to the hub or from the hub to a destination. Therefore when we index the legs with $ij$, we must have that exactly one of $i$ and $j$ is $0$. The triplet $ijk$ represents class $k$ of itinerary $ij$.

- $p_{ijk}$: probability of a request of class $k$ of itinerary $ij$;
- $r_{ijk}$: probability of a cancelation of class $k$ of itinerary $ij$;
- $f_{ijk}$: fare of class $k$ of itinerary $ij$;
- $w_{ijk}$: penalty cost of denying boarding an itinerary $ij$ class $k$ reservation;
- $C_{ij}$: capacity of flight leg $ij$, where exactly one of $i$ and $j$ is equal to $0$;
- $n_{ijk}$: state variable that represents the number of bookings for class $k$ of itinerary $ij$;
- $X_{ijk}$: binary random variable, when equal to $1$, representing a cancelation from class $k$ of itinerary $ij$.

### 5.4.2 Generic Recursion

The rationale behind the recursion for the general case is the same to that for the two-leg case we obtained earlier. The only difference is the dimension of the problem. For convenience, define the following three vectors:

$$ n = (n_{101}, n_{102}, \ldots, n_{10K}, n_{111}, n_{112}, \ldots, n_{11K}, \ldots, n_{1m1}, n_{1m2}, \ldots, n_{1mK}, n_{201}, n_{202}, \ldots, n_{20K}, n_{211}, n_{212}, \ldots, n_{21K}, \ldots, n_{2m1}, n_{2m2}, \ldots, n_{2mK}). $$
Chapter 5. Dynamic Network Models

\[ n_{m01}, n_{m02}, \ldots, n_{m0K}, n_{m11}, n_{m12}, \ldots, n_{m1K}, \ldots, n_{mm1}, n_{mm2}, \ldots, n_{mmK}, \]
\[ n_{011}, n_{012}, \ldots, n_{01K}, n_{021}, n_{022}, \ldots, n_{02K}, n_{0m1}, n_{0m2}, \ldots, n_{0mK}, \]
\[ X = \{ X_{101}, X_{102}, \ldots, X_{10K}, X_{111}, X_{112}, \ldots, X_{11K}, \ldots, X_{1m1}, X_{1m2}, \ldots, X_{1mK}, \]
\[ X_{201}, X_{202}, \ldots, X_{20K}, X_{211}, X_{212}, \ldots, X_{21K}, \ldots, X_{2m1}, X_{2m2}, \ldots, X_{2mK}, \]
\[ \ldots \]
boarding. Then the boundary condition can be formulated as the following problem:

\[
\begin{align*}
\min & \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=1}^{K} w_{ijk} z_{ijk} \\
\text{s.t.} & \sum_{j=0}^{m} \sum_{k=1}^{K} z_{ijk} \geq \sum_{j=0}^{m} \sum_{k=1}^{K} n_{ijk} - C_{i0} \quad \text{for } i = 1, \ldots, m \\
& \sum_{i=0}^{m} \sum_{k=1}^{K} z_{ijk} \geq \sum_{i=0}^{m} \sum_{k=1}^{K} n_{ijk} - C_{0j} \quad \text{for } j = 1, \ldots, m \\
& z_{ijk} \leq n_{ijk} \quad \forall i, j, k \in Z \\
& z_{ijk} \geq 0 \quad \forall i, j, k \in Z
\end{align*}
\]  

(5.5)

i.e., in the end given the reservations of all itineraries and classes, we decide how many reservations of which itineraries and classes to deny boarding, if we have to at all, such that the overbooking penalty is minimized. The network effects are captured in the constraint sets 1 and 2 because connecting itineraries (those that start at an origin and end at a destination) appear in both constraint sets while local itineraries (those that start at an origin and end at the hub or start at the hub and end at a destination) only appear in one of the constraint sets.

The solution to dynamic program (5.4) and (5.5) is a policy that determines whether to accept or deny a request for a fare class on an itinerary given \( n \) and \( t \).

### 5.4.3 A Global Dynamic Model without Cancelations and Overbookings

Without cancelations and overbookings, the dynamic network problem is greatly simplified. In particular we do not need to know the bookings of all the itinerary-class combinations. The system can be fully characterized by the number of seats available on the each flight \( l^o \gamma \). This will become clear when we introduce the recursion formula.

In addition to the notation defined in Section 5.4.1, we define:
Chapter 5. Dynamic Network Models

• $s_{i0}$: number of seats available on flight leg $i0$.

• $s_{0j}$: number of seats available on flight leg $0j$.

Furthermore, we define:

$$\mathbf{s} = (s_{10}, s_{20}, \ldots, s_{m0}, s_{01}, s_{02}, \ldots, s_{0m})$$

for all $\mathbf{s}$ such that

$$s_{i0} \leq C_{i0}, \quad i = 1, \ldots, m$$

$$s_{0j} \leq C_{0j}, \quad j = 1, \ldots, m$$

Finally we define for all $0 \leq i, j \leq m$ and $i + j \neq m$:

$$\delta_{ij} = (\tau_{11}, \tau_{2i}, \ldots, \tau_{mi}, \omega_{j1}, \omega_{j2}, \ldots, \omega_{jm})$$

where

$$\tau_{ki} = \begin{cases} 1 & \text{if } k = i \quad \text{and} \quad \omega_{jk} = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{if } k \neq j \end{cases} \\
0 & \text{if } k \neq i \end{cases}$$

Let $R(\mathbf{s}, t)$ denote the expected revenue from time $t$ until 0 given $\mathbf{s}$. We have the following recursion:

$$R(\mathbf{s}, t + 1) = \sum_{i=0}^{m} \sum_{j=0}^{m} \sum_{k=1}^{K} p_{ijk} \cdot \max_{y \in \{0,1\}} \{y \cdot f_{ijk} + R(\mathbf{s} - y \cdot \delta_{ij}, t)\} +$$

$$\left(1 - \sum_{i=1}^{m} \sum_{j=0}^{m} \sum_{k=1}^{K} p_{ijk}\right) \cdot R(\mathbf{s}, t) \quad (5.6)$$

Note that the network effect is captured by the term $y \cdot \delta_{ij}$. If $i \neq 0$ and $j \neq 0$, then $\tau_{ii} = 1$ and $\omega_{jj} = 1$, meaning itinerary $ij$ takes one seat each from leg $i0$ and leg $0j$. If $i \neq 0$ and $j = 0$, then $\tau_{ii} = 1$ and $\omega_{jk} = 0$ for all $k = 1, \ldots, m$, meaning itinerary $i0$ only takes a seat from leg $i0$. Similarly, if $i = 0$ and $j \neq 0$, then itinerary $0j$ only
Chapter 5. Dynamic Network Models

takes a seat from leg 0j.

The boundary condition is:

\[ R(s, 0) = 0 \]  \hspace{1cm} (5.7)

The solution to the dynamic program (5.6) and (5.7) is a policy that determines whether to accept or reject a fare request on an itinerary given \( s \) (number of seats available on all flight legs) and time \( t \).

5.5 An Approximation of Dynamic Models

In order to address the dimensionality problems with dynamic models, we will approximate the revenue function \( R(n, t) \) as follows:

\[ \hat{R}(n, t) = n' \cdot A_t \cdot n + \beta_t \cdot n + \alpha_t. \]  \hspace{1cm} (5.8)

where \( A_t \) (\( m \times m \)), \( \beta_t \) (\( m \times 1 \)) and \( \alpha_t \) are parameters that need to be fit. This essentially is a quadratic approximation.

5.5.1 Solving for \( \alpha_t, \beta_t \) and \( A_t \)

In order to fit the parameters, we randomly generate \( N \) data points, \( n^1, n^2, \ldots, n^N \). \( N \) is a number we choose arbitrarily. Normally the larger the \( N \) the more accurate the approximation is. Using recursion (5.4) and boundary condition (5.5), we can calculate exactly \( R(n^i, 1), i = 1, \ldots, N \). We then do a least-square fitting using \( R(n^i, 1) (i = 1, \ldots, N) \) to find \( A_1, \beta_1 \) and \( \alpha_1 \). So we have an approximation for \( R(\cdot, 1) \)

\[ \hat{R}(n, 1) = n' \cdot A_1 \cdot n + \beta_1 \cdot n + \alpha_1 \]

Next we use \( \hat{R}(n^i, 1) \) and recursion (5.4) to obtain \( N \) approximated values of \( R(\cdot, 2), \hat{R}(n^i, 2), i = 1, 2, \ldots, N \). Then we do a least-square fitting with \( \hat{R}(n^i, 2) (i = \)
1, \ldots, N) to find \( A_2, \beta_2 \) and \( \alpha_2 \). So we have:

\[
\hat{R}(n, 2) = n' \cdot A_2 \cdot n + \beta_2' \cdot n + \alpha_2
\]

In general we use \( \hat{R}(n', t) \) and recursion (5.4) to obtain \( N \) approximated values of \( R(\cdot, t + 1), \hat{R}(n', t + 1), i = 1, 2, \ldots, N \). Then we do a least-square fitting with \( \hat{R}(n', t + 1) \) to find \( A_{t+1}, \beta_{t+1} \) and \( \alpha_{t+1} \). So we have:

\[
\hat{R}(n, t + 1) = n' \cdot A_{t+1} \cdot n + \beta_{t+1}' \cdot n + \alpha_{t+1}
\]

This process continues until we reach \( T \).

The algorithm for determining \( \alpha_t, \beta_t \) and \( A_t \) can be stated formally as follows:

**Algorithm 5.1 (Quadratic Approximation Algorithm)**

**BEGIN**

**Initialization:**

- Randomly generate \( N \) state vectors \( n^1, n^2, \ldots, n^N \);
- Compute \( \hat{R}(n^i, 1) = R(n^i, 1), i = 1, \ldots, N \).

**Iteration:**

For \( t = 1 \) to \( T \), do:

- Compute \( \alpha_t, \beta_t \) and \( A_t \) using least square from \( \hat{R}(n^i, t), i = 1, \ldots, N \); thus obtaining \( \hat{R}(n^i, t) = n'^i \cdot A_t \cdot n^i + \beta_t' \cdot n^i + \alpha_t \).
- Compute \( \hat{R}(n^i, t + 1) \) for \( i = 1, 2, \ldots, N \), using \( \hat{R}(n^i, t) \) and recursion (5.4).

**END**

The motivation for this approximation approach is the belief that the revenue functions \( R(n, t) \) are smooth and thus can be well approximated with a quadratic form.
Chapter 6

Simulation and Computational Results

In this chapter, we evaluate the performance of the models and approaches we developed in earlier chapters through comparisons with current industry practices. We perform several experiments using numerical computation and computer simulation.

In Chapter 2, we derived a “wait-and-see” approach for single-leg static models. We showed that the wait-and-see approach gives optimal booking limits. In this chapter, we will test the relative benefits of the wait-and-see approach over the Expected Marginal Seat Revenue (EMSR) model, which is widely used in practice. The results show that the wait-and-see approach performs consistently better than the EMSR model. The improvement is not only in the expected revenue but also in the load factor and the revenue variability. In any of the situations where the variability of the demands is large, the capacity is tight relative to the demands and the fare range is wide, the improvement is more significant.

Also in Chapter 2, we extended the wait-and-see approach to incorporate cancellations and overbookings. A key component of the model is a simplification scheme for estimating the overbooking penalty. In addition to simplifying the computation, it is also the basis for the analysis we did for the model. In this chapter, we will
justify the simplification using several numerical examples. The results reveal that
the wait-and-see approach with cancelations and overbookings performs well against
the a priori approach (which is implemented in this chapter using a new global op-
timization scheme) and that the error caused by the approximation is minimal.

In Chapter 3, we developed dynamic models for cases where fare requests are no
longer subject to the assumption that lower-class arrivals occur first and are modeled
as dynamic stochastic process. In this chapter, we demonstrate the benefits of the
dynamic models. We compare dynamic models with static ones with and without
re-optimization.

In Chapter 4, we developed various network models. We proposed a hierarchi-
cal approach that combines network algorithms with our single-leg wait-and-see ap-
proach. We perform case studies in this chapter that compare the hierarchical ap-
proach with the widely used industry practice “virtual nesting”.

Next we test network dynamic approaches against virtual nesting (with re-optimization).
The results clearly show that although re-optimization improves the performance of
the EMSR model in a single-leg environment it has limited effect in improving the
performance of the virtual nesting approach, which is a heuristic generalization of
the EMSR model to the network case. In particular, re-optimization is not able to
correct virtual nesting’s weakness in handling network effects.

A primary difficulty with dynamic models in Chapters 3 and 5 is the amount of
computation involved caused by large dimensions. Even for the single-leg case the
computation can become very extensive. To counter the “curse of dimensionality”,
we proposed a quadratic approximation for the dynamic models. We will illustrate
in this chapter the performance of the approximation with numerical examples.
6.1 Wait-and-see versus EMSR

We consider single-leg cases with multiple classes and without cancelations (thus no overbookings). The goal is to compare the wait-and-see approach and the EMSR model ([12] and [2]), which is widely used in the airline industry.

A Case Study

In the following example, there are 5 fare classes, the flight capacity is 195. Demands are normally distributed. Table 6.1 lists the fares, mean demands and demand standard deviations for all classes.

The wait-and-see approach and the EMSR model are applied to the case, respectively, to obtain booking limits and total expected revenue. The results are summarized in Table 6.2.

In addition, simulations are run with respect to the booking limits obtained through the wait-and-see approach as well as those obtained through the EMSR
Table 6.3: Simulation Statistics

<table>
<thead>
<tr>
<th>class</th>
<th>Wait-and-See reqts</th>
<th>Wait-and-See acpts</th>
<th>EMSR reqts</th>
<th>EMSR acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20.05</td>
<td>18.85</td>
<td>20.05</td>
<td>19.42</td>
</tr>
<tr>
<td>2</td>
<td>34.99</td>
<td>34.34</td>
<td>34.99</td>
<td>34.47</td>
</tr>
<tr>
<td>3</td>
<td>40.00</td>
<td>39.38</td>
<td>40.00</td>
<td>39.49</td>
</tr>
<tr>
<td>4</td>
<td>44.92</td>
<td>44.30</td>
<td>44.92</td>
<td>44.42</td>
</tr>
<tr>
<td>5</td>
<td>50.13</td>
<td>32.83</td>
<td>50.13</td>
<td>17.91</td>
</tr>
<tr>
<td>Total</td>
<td>190.09</td>
<td>169.71</td>
<td>190.09</td>
<td>155.71</td>
</tr>
<tr>
<td>Revenue $^a$</td>
<td>89073.5</td>
<td></td>
<td>88310.4</td>
<td></td>
</tr>
<tr>
<td>SD $^b$</td>
<td>10919.6</td>
<td></td>
<td>11922.2</td>
<td></td>
</tr>
</tbody>
</table>

$^a$Sample mean revenue  
$^b$Sample standard deviation

model. In the simulations, we generate normally distributed random demands for the 5 fare classes in the assumed order in which fare requests occur (i.e., lower classes book up first) The generated demands are then accommodated according to their booking limits. The simulations are run a large number of times (20,000 times in this case). In Table 6.3 we include the sample mean and deviation of the revenues, and the number of requests made and accepted.

We observe a 0.86% improvement in terms of total expected revenue of the wait-and-see approach over the EMSR model. Also the sample standard deviation in the case of the wait-and-see approach is 9.18% smaller compared to the case of the EMSR. Under the wait-and-see approach, we carry about 15 more passengers than under EMSR, a 9.7% increase, which implies a larger percentage of filled seats (load factor), an important indicator for an airline’s market share. As we can see, the number of requests accepted for the first 4 classes is almost identical. There is a sizable difference between the wait-and-see approach and the EMSR model in terms of the number of requests accepted for class 5. The EMSR approach discriminates against lower classes, especially the lowest class, which has in this example a significantly lower fare.
Table 6.4: Variation 1: 5 fare classes, capacity = 195

<table>
<thead>
<tr>
<th>class</th>
<th>fare</th>
<th>mean</th>
<th>std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>900.0</td>
<td>20.0</td>
<td>8.0</td>
</tr>
<tr>
<td>2</td>
<td>800.0</td>
<td>35.0</td>
<td>12.0</td>
</tr>
<tr>
<td>3</td>
<td>600.0</td>
<td>40.0</td>
<td>13.0</td>
</tr>
<tr>
<td>4</td>
<td>400.0</td>
<td>45.0</td>
<td>14.0</td>
</tr>
<tr>
<td>5</td>
<td>100.0</td>
<td>50.0</td>
<td>15.0</td>
</tr>
</tbody>
</table>

Table 6.5: Results for Variation 1

<table>
<thead>
<tr>
<th>algorithm</th>
<th>Booking Limits</th>
<th>exp. rev.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Wait-and-See</td>
<td>195</td>
<td>185</td>
</tr>
<tr>
<td>EMSR</td>
<td>195</td>
<td>184</td>
</tr>
</tbody>
</table>

**Variation 1: More Variability**

In this example, we increase the variability of the demands in the previous example (see Table 6.4) and report the results in Table 6.5.

We then simulate the system as before. The advantage of the wait-and-see approach in terms of the expected revenue increases to 1.04%. The gap between sample standard deviations is increased to 10.2%. The gap between the number of passengers carried is 12.2%.

The simulation results in Table 6.6 show that the lowest class is much more heavily discriminated by the EMSR approach. This is because as variability increases more seats need to be set aside to hedge against the randomness in the demands for higher classes. The EMSR is exceedingly aggressive in discriminating the lowest class, thus the drastic difference in the statistics.
Table 6.6: Simulation Statistics: Increased Variability

<table>
<thead>
<tr>
<th>class</th>
<th>Wait-and-See</th>
<th>EMSR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>reqts</td>
<td>acpts</td>
</tr>
<tr>
<td>1</td>
<td>20.07</td>
<td>18.85</td>
</tr>
<tr>
<td>2</td>
<td>35.03</td>
<td>34.26</td>
</tr>
<tr>
<td>3</td>
<td>40.08</td>
<td>39.39</td>
</tr>
<tr>
<td>4</td>
<td>45.18</td>
<td>44.54</td>
</tr>
<tr>
<td>5</td>
<td>49.98</td>
<td>28.39</td>
</tr>
<tr>
<td>Total</td>
<td>190.34</td>
<td>165.27</td>
</tr>
<tr>
<td>Revenue $^a$</td>
<td>88516.3</td>
<td>87600.9</td>
</tr>
<tr>
<td>SD $^b$</td>
<td>13426.4</td>
<td>14837.8</td>
</tr>
</tbody>
</table>

$^a$Sample mean revenue  
$^b$Sample standard deviation

Table 6.7: Variation 2: 5 fare classes, capacity = 235

<table>
<thead>
<tr>
<th>class</th>
<th>fare</th>
<th>mean</th>
<th>std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>900.0</td>
<td>20.0</td>
<td>8.0</td>
</tr>
<tr>
<td>2</td>
<td>800.0</td>
<td>35.0</td>
<td>12.0</td>
</tr>
<tr>
<td>3</td>
<td>600.0</td>
<td>40.0</td>
<td>13.0</td>
</tr>
<tr>
<td>4</td>
<td>400.0</td>
<td>45.0</td>
<td>14.0</td>
</tr>
<tr>
<td>5</td>
<td>100.0</td>
<td>50.0</td>
<td>15.0</td>
</tr>
</tbody>
</table>

**Variation 2: More Capacity**

Our final example involves increasing the capacity in Variation 1. In particular we increase the capacity to 235 (see Table 6.7). The results of applying the wait-and-see approach and the EMSR model are shown in Table 6.8.

Table 6.9 lists the statistics from running simulations using the booking limits from Table 6.8. As we can see, the gaps decrease in all categories: expected revenue, standard deviation and number of passengers carried.

Based on these experiments we conclude the following: The wait-and-see approach performs consistently better than the EMSR, not only in terms of the expected rev-
### Table 6.8: Results for Variation 2

<table>
<thead>
<tr>
<th>algorithm</th>
<th>Booking Limits</th>
<th>exp. rev.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Wait-and-See</td>
<td>235</td>
<td>225</td>
</tr>
<tr>
<td>EMSR</td>
<td>235</td>
<td>224</td>
</tr>
</tbody>
</table>

### Table 6.9: Simulation Statistics: More Capacity

<table>
<thead>
<tr>
<th>class</th>
<th>Wait-and-See</th>
<th>EMSR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>reqts</td>
<td>acpts</td>
</tr>
<tr>
<td>1</td>
<td>20.07</td>
<td>19.28</td>
</tr>
<tr>
<td>2</td>
<td>35.03</td>
<td>34.46</td>
</tr>
<tr>
<td>3</td>
<td>40.08</td>
<td>39.53</td>
</tr>
<tr>
<td>4</td>
<td>45.18</td>
<td>44.66</td>
</tr>
<tr>
<td>5</td>
<td>49.98</td>
<td>48.80</td>
</tr>
<tr>
<td>Total</td>
<td>190.34</td>
<td>186.71</td>
</tr>
<tr>
<td>Revenue(^a)</td>
<td>91371.0</td>
<td>91072.1</td>
</tr>
<tr>
<td>SD(^b)</td>
<td>14608.7</td>
<td>14954.8</td>
</tr>
</tbody>
</table>

\(^a\)Sample mean revenue

\(^b\)Sample standard deviation
enue but also in terms of the load factor. The improvement is more significant with increasing variability, tighter capacity, and wider fare range.

6.2 Static Models with Cancelations

In Chapter 2, we extended the wait-and-see model to the case with cancelations and overbookings and proposed an approximation algorithm. In this section we will try to evaluate the effect of the approximation through numerical experiments.

We conduct our testing as follows. We compute the booking limits using the approximation. To evaluate the booking limits, we simulate the system by generating fare requests as well as cancelations. In each simulation experiment from a total 20,000 runs, the following data is collected: total revenue, number of overbookings and penalty cost. By comparing the revenue derived from the approximation and the simulated revenue, we evaluate the effect of the approximation. For comparison purposes, we used a global optimization algorithm [14] to find the overall best policy. We briefly review the global optimization algorithm next.

Overview of Global Optimization

The global optimization algorithm used in this section is developed by Tang [14]. It is a randomized global optimization scheme that adaptively and intelligently partitions the entire feasible region into subregions within which to conduct random evaluations. Instead of finding the subregion where the global maximum is likely located, it looks for the subregion where the largest improvement is likely with a moderate number of evaluations. The algorithm has minimal requirement on the objective function and can even handle objective functions without closed form expressions. It is particular useful in our context, as simulation is needed to evaluate the objective function. It has proved to be effective and robust.

One specific form of the adaptive partition algorithm deals with the problem with
Table 6.10: Example: 4 classes, capacity = 200, penalty = $600

<table>
<thead>
<tr>
<th>class</th>
<th>fare</th>
<th>mean</th>
<th>std. dev</th>
<th>showup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>500</td>
<td>10</td>
<td>5</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>300</td>
<td>30</td>
<td>15</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>60</td>
<td>25</td>
<td>0.75</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>80</td>
<td>35</td>
<td>0.8</td>
</tr>
</tbody>
</table>

a feasible region defined by:

\[ S = \{ (x_1, x_2, \ldots, x_n) : 0 \leq x_n \leq x_{n-1} \leq \ldots \leq x_1 \leq C \} \]

where \( x_i \)'s are decision variables and \( C \) is positive constant. In our context, \( x_i \)'s represent the booking limits.

**Numerical Examples**

Let us consider the case shown in Table 6.10. The expected revenue derived using the approximation is 21404.98. Interestingly, simulation with these booking limits gives a simulated revenue of 21397.25. The algorithm in [14] gives a solution with a revenue of 21289.53. The running time of the approximation algorithm is less than 3 minutes, while the global optimization algorithm takes 3 to 7 hours to run depending on the number of simulations run for each function evaluation.

In another case, we reduce the capacity by 50, everything else being the same. The approximate revenue is 20301.90. The simulated revenue is 20106.02. The difference is 195.88. The revenue generated using global optimization is 19891.13.

If we further deduct capacity by 50 to 100, the approximated revenue becomes 16465.30. The simulated revenue is 16234.02. The revenue generated by global optimization 16195.60.

Finally we consider an example of a larger scale, as shown in Table 6.11. The approximated revenue is 125131.06. The simulated revenue is 125082.18. The revenue
Table 6.11: Example: 5 classes, capacity = 280, penalty = $1250

<table>
<thead>
<tr>
<th>class</th>
<th>fare</th>
<th>mean</th>
<th>std. dev</th>
<th>showup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1150</td>
<td>20</td>
<td>7.0</td>
<td>0.25</td>
</tr>
<tr>
<td>3</td>
<td>980</td>
<td>40</td>
<td>13.5</td>
<td>0.46</td>
</tr>
<tr>
<td>3</td>
<td>750</td>
<td>70</td>
<td>30.0</td>
<td>0.65</td>
</tr>
<tr>
<td>4</td>
<td>600</td>
<td>85</td>
<td>35.0</td>
<td>0.84</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>55</td>
<td>20</td>
<td>0.90</td>
</tr>
</tbody>
</table>

generated by the global optimization algorithm is 124945.79.

The previous examples illustrate the following points:

1. Our approximate dynamic programming algorithm gives better solutions than
global optimization. It is also very fast.

2. The approximation gives revenues which are very close to simulation results.
   This is particularly useful as the approximate dynamic programming algorithm
   is quite efficient.

### 6.3 Single-Leg Dynamic Models versus Static Models

In Chapter 3, we developed dynamic models for situations where fare requests do
not satisfy the assumption that lower classes always arrive early. Indeed in practice
lower classes do not arrive strictly before higher classes, but static models are used
anyway. Static models as a seat inventory control mechanism are well-defined even
if fare requests violate the static assumption, i.e., one can always use booking limits
to decide whether to accept or reject fare requests. In this section we compare the
dynamic models developed in Chapter 3 with current industry practices, where a
static model is rerun periodically.
We consider the single-leg case with multiple fare classes and without cancelations. We are given a certain booking horizon. The booking horizon is partitioned into fine booking intervals so that the number of booking intervals is large. Within each booking interval there can have at most one request for any of the fare classes. For each fare class, there is a homogeneous probability that a request for the fare class will occur in any booking interval. Threshold times are calculated using the dynamic model (without cancelations) in Chapter 3. The threshold times are then fed to a simulator. The simulator starts at time horizon and runs backwards till time 0 (departure time). It generates requests for the fare classes for each booking interval of the booking horizon according to the arrival probabilities of the fare classes. For each generated request, the threshold time of its fare class is checked. If the current booking interval (current time) is greater than the threshold time, the request is disregarded; otherwise it is accepted.

On the static side, we aggregate fare requests over the entire booking horizon. In this case, for each fare class the number of requests is a binomial random variable $B(T, p)$, where $T$ is the horizon and $p$ is the arrival probability for each booking interval. The numbers of requests for different fare classes are independent of each other. Once we obtain the aggregated demands, we can use a static model to determine the booking limits on individual fare classes. The booking limits then are fed to another simulator. This simulator is similar to the previous one. The requests are generated in exactly the same manner. Only the control is different. Regardless of the arrival time of a request, it is accepted as long as the booking limits of its class and all higher classes are not violated.

**An Example**

Consider the example shown in Table 6.12. The data contained in the table are arrival probabilities within each time interval and fares.

We apply the EMSR algorithm and the dynamic algorithm to the example in
Table 6.12: Example: 4 classes, horizon = 1000, capacity = 150

<table>
<thead>
<tr>
<th>class</th>
<th>arr. prob.</th>
<th>fare</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.020</td>
<td>1000.0</td>
</tr>
<tr>
<td>2</td>
<td>0.060</td>
<td>700.0</td>
</tr>
<tr>
<td>3</td>
<td>0.045</td>
<td>550.0</td>
</tr>
<tr>
<td>4</td>
<td>0.055</td>
<td>250.0</td>
</tr>
</tbody>
</table>

Table 6.13: Results for the example in Table 6.12.

<table>
<thead>
<tr>
<th></th>
<th>EMSR</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>BK1</td>
<td>150</td>
<td>92207.80</td>
</tr>
<tr>
<td>BK2</td>
<td>131</td>
<td>4958.60</td>
</tr>
<tr>
<td>BK3</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>BK4</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>revenue</td>
<td>89513.25</td>
<td>6446.31</td>
</tr>
<tr>
<td>SD</td>
<td>6446.31</td>
<td>4958.60</td>
</tr>
</tbody>
</table>

Table 6.12 and simulate the policies proposed by the two algorithms. We collect the following data:

1. sample mean revenue,
2. sample standard deviation of revenue,
3. fare requests for each class,
4. fare requests accepted for each class,
5. total number of fare requests and total number of fare requests accepted.

Table 6.13 shows that the dynamic model yields 2,694.55 or 3.01% more revenue than the EMSR model. Also the standard deviation of the revenue generated by the dynamic model is 1,487.71 or 23.08% less than that of the revenue generated by the EMSR model. The best improvement observed of the wait-and-see approach, the optimal static model, over the EMSR model is around 1%. An improvement of 3.01% can only be attributed to the dynamic nature of fare requests.
Table 6.14: Simulation Statistics for the example in Table 6.12.

<table>
<thead>
<tr>
<th>class</th>
<th>EMSR</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>reqs</td>
<td>acpts</td>
</tr>
<tr>
<td>1</td>
<td>19.69</td>
<td>19.69</td>
</tr>
<tr>
<td>2</td>
<td>59.55</td>
<td>59.33</td>
</tr>
<tr>
<td>3</td>
<td>44.70</td>
<td>44.55</td>
</tr>
<tr>
<td>4</td>
<td>54.80</td>
<td>15.00</td>
</tr>
<tr>
<td>Total</td>
<td>178.74</td>
<td>138.63</td>
</tr>
</tbody>
</table>

Table 6.14 shows that a 10-seat difference between the numbers of passengers carried under the dynamic model and the EMSR model, respectively. The difference lies in the number of requests accepted for the lowest class while the requests accommodated for classes 1, 2 and 3 are almost identical between the dynamic model and the EMSR model. From this we can draw two conclusions. First, the dynamic model does have a nesting flavor. Second, it handles the lowest class more intelligently by showing more lenience towards it. As a result, the dynamic model not only produces more revenue than the EMSR model, but also achieves a better load factor.

To make our study more comprehensive, we also use the wait-and-see algorithm to generate booking limits. Then we run the same simulation as we did on the EMSR. The results are summarized in Tables 6.15 and 6.16. The results show that the dynamic model performs better than the wait-and-see approach in all three areas: revenue, variability of revenue and load factor. From these results as well as from other experiments we performed extensive studies it can also be seen that the wait-and-see is a better heuristic for handling dynamic arrivals since the differences in all the three areas are narrowed. This can be explained by the fact that the difference in performance lies mainly in the handling of the lowest fare class. The wait-and-see approach has demonstrated that it is more lenient to the lowest class both in a static setting and a dynamic setting.
Table 6.15: Results for the example in Table 6.12 with the wait-and-see approach applied

<table>
<thead>
<tr>
<th></th>
<th>Wait-and-See</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BK1&lt;sup&gt;a&lt;/sup&gt;</td>
<td>BK2</td>
</tr>
<tr>
<td>150</td>
<td>132</td>
<td>74</td>
</tr>
</tbody>
</table>

<sup>a</sup>BK: booking limit

Table 6.16: Simulation statistics for the example in Table 6.12 with the wait-and-see approach applied

<table>
<thead>
<tr>
<th>class</th>
<th>Wait-and-See</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>reqts</td>
<td>acpts</td>
</tr>
<tr>
<td>1</td>
<td>19.59</td>
<td>19.59</td>
</tr>
<tr>
<td>2</td>
<td>59.26</td>
<td>59.01</td>
</tr>
<tr>
<td>3</td>
<td>44.47</td>
<td>44.16</td>
</tr>
<tr>
<td>4</td>
<td>54.53</td>
<td>19.00</td>
</tr>
<tr>
<td>Total</td>
<td>177.85</td>
<td>141.76</td>
</tr>
</tbody>
</table>

Re-Optimization Using Static Models

In practice, static models are used in a re-optimization manner. Every certain period of time booking limits are re-calculated and the new booking limits are used for seat inventory control. In this study, re-optimization is implemented as follows. In the beginning, the demand is aggregated over the entire booking horizon. A certain number of re-optimization points equally apart are established on the booking horizon. As simulation proceeds and reaches a re-optimization point, the booking limits will be recomputed. First the demands need to be re-aggregated. In particular, if \( T \) is the entire booking horizon and \( d \) is the re-optimization interval, the demand at re-optimization point \( i \) will be binomial \( B(T - (i - 1) \cdot d, p) \). Another input that needs to be updated is the capacity. It is simply the flight capacity minus bookings. The new demands and the new capacity will be passed to the static model to compute the booking limits that will be in use between the current re-optimization point and the
next one.

We take the example in the previous section and do re-optimization. In our implementation a parameter called "update interval" which defines the interval between consecutive re-optimizations. In the example in Table 6.12, the booking horizon is 1000. If we think of one interval as one hour, a booking horizon of 1000 is approximate 40 calendar days. If the update interval is 1000, one re-optimization (the initial optimization itself) is done. If the update interval is 500, two re-optimizations is done, etc. If the update interval is 125, then 4 re-optimizations are done. In our case, the re-optimization interval is selected such that in the beginning the number of re-optimizations increases by one until it reaches 10, and afterwards it increases by 5 until it reaches 1000. In other words,

$$\text{update interval} = \left\lceil \frac{\text{booking horizon}}{\text{number of re-optimizations}} \right\rceil.$$

Table 6.17 selectively lists the results for different re-optimization intervals. The results demonstrate that the re-optimization of the EMSR model does narrow the gaps between itself and the dynamic model. When the update interval is 22, i.e., 45 re-optimizations are done, the difference in revenue becomes 0.56% in favor of the dynamic model, the difference in the number of passengers carried is about 1 in favor of the dynamic model. The difference in revenue standard deviation becomes 5.34% in favor of the EMSR with re-optimization.

Even in the face of re-optimization of the EMSR model, the dynamic model retains its advantages. From a theoretical point of view, it provides the optimal solution for the situations where the static assumption does not hold. It can serve as a measuring yardstick for the development of heuristics since it gives an upper bound. From a practical point of view, for an airline with a revenue of 7 to 8 billion dollars, a 0.56% improvement can be significant.

The conclusions we can draw are that dynamic effect cannot be ignored and using static models to handle dynamic cases will yield non-optimal results.
Table 6.17: Re-optimization of the example in Table 6.12

<table>
<thead>
<tr>
<th>interval</th>
<th>revenue</th>
<th>SD(^a)</th>
<th>class</th>
<th>reqs(^b)</th>
<th>acpts(^c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>89513.25</td>
<td>6446.31</td>
<td>1</td>
<td>19.69</td>
<td>19.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.55</td>
<td>59.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.70</td>
<td>44.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>54.80</td>
<td>15.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>178.74</td>
<td>138.63</td>
</tr>
<tr>
<td>500</td>
<td>90497.25</td>
<td>5542.07</td>
<td>1</td>
<td>19.69</td>
<td>19.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.56</td>
<td>59.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.70</td>
<td>44.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>54.80</td>
<td>19.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>178.75</td>
<td>142.77</td>
</tr>
<tr>
<td>333</td>
<td>90914.30</td>
<td>5103.13</td>
<td>1</td>
<td>19.70</td>
<td>19.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.57</td>
<td>59.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.71</td>
<td>44.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>54.82</td>
<td>21.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>178.80</td>
<td>144.87</td>
</tr>
<tr>
<td>250</td>
<td>91026.95</td>
<td>4875.53</td>
<td>1</td>
<td>19.72</td>
<td>19.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.63</td>
<td>59.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.76</td>
<td>43.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>54.88</td>
<td>22.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>178.99</td>
<td>145.45</td>
</tr>
<tr>
<td>200</td>
<td>91097.90</td>
<td>4990.64</td>
<td>1</td>
<td>19.75</td>
<td>19.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.68</td>
<td>59.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.79</td>
<td>44.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>54.93</td>
<td>22.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.15</td>
<td>145.59</td>
</tr>
<tr>
<td>166</td>
<td>91179.95</td>
<td>4850.22</td>
<td>1</td>
<td>19.74</td>
<td>19.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.67</td>
<td>59.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.78</td>
<td>43.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>54.91</td>
<td>22.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.10</td>
<td>145.75</td>
</tr>
<tr>
<td>142</td>
<td>91200.00</td>
<td>4994.37</td>
<td>1</td>
<td>19.77</td>
<td>19.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.75</td>
<td>59.49</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.85</td>
<td>43.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>54.99</td>
<td>22.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.35</td>
<td>145.77</td>
</tr>
</tbody>
</table>
Table 6.17: Re-optimization of the example in Table 6.12 (continued)

<table>
<thead>
<tr>
<th>interval</th>
<th>revenue</th>
<th>SD</th>
<th>class</th>
<th>reqs</th>
<th>acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>91326.20</td>
<td>4933.33</td>
<td>1</td>
<td>19.77</td>
<td>19.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.75</td>
<td>59.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.85</td>
<td>43.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>54.99</td>
<td>23.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.35</td>
<td>146.26</td>
</tr>
<tr>
<td>50</td>
<td>91479.20</td>
<td>4670.93</td>
<td>1</td>
<td>19.77</td>
<td>19.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.75</td>
<td>59.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.86</td>
<td>43.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>54.99</td>
<td>24.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.37</td>
<td>147.18</td>
</tr>
<tr>
<td>33</td>
<td>91592.00</td>
<td>4567.74</td>
<td>1</td>
<td>19.79</td>
<td>19.79</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.77</td>
<td>59.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.88</td>
<td>43.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.03</td>
<td>24.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.47</td>
<td>147.43</td>
</tr>
<tr>
<td>28</td>
<td>91638.95</td>
<td>4663.73</td>
<td>1</td>
<td>19.80</td>
<td>19.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.80</td>
<td>59.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.91</td>
<td>43.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.05</td>
<td>24.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.56</td>
<td>147.49</td>
</tr>
<tr>
<td>25</td>
<td>91603.70</td>
<td>4716.25</td>
<td>1</td>
<td>19.80</td>
<td>19.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.81</td>
<td>59.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.92</td>
<td>43.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.06</td>
<td>24.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.60</td>
<td>147.46</td>
</tr>
<tr>
<td>22</td>
<td>91697.40</td>
<td>4693.20</td>
<td>1</td>
<td>19.81</td>
<td>19.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.83</td>
<td>59.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.92</td>
<td>43.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.07</td>
<td>24.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.62</td>
<td>147.56</td>
</tr>
<tr>
<td>20</td>
<td>91628.60</td>
<td>4703.79</td>
<td>1</td>
<td>19.81</td>
<td>19.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.86</td>
<td>59.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.95</td>
<td>43.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.11</td>
<td>24.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td>179.73</td>
<td>147.49</td>
</tr>
</tbody>
</table>
Table 6.17: Re-optimization of the example in Table 6.12 (continued)

<table>
<thead>
<tr>
<th>interval</th>
<th>revenue</th>
<th>SD</th>
<th>class</th>
<th>reqs</th>
<th>acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>91570.10</td>
<td>4713.03</td>
<td></td>
<td>1</td>
<td>19.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td></td>
<td>179.71</td>
</tr>
<tr>
<td>10</td>
<td>91630.30</td>
<td>4594.19</td>
<td></td>
<td>1</td>
<td>19.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td></td>
<td>179.73</td>
</tr>
<tr>
<td>8</td>
<td>91643.20</td>
<td>4631.27</td>
<td></td>
<td>1</td>
<td>19.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td></td>
<td>179.68</td>
</tr>
<tr>
<td>5</td>
<td>91694.00</td>
<td>4541.49</td>
<td></td>
<td>1</td>
<td>19.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td></td>
<td>179.83</td>
</tr>
<tr>
<td>3</td>
<td>91664.95</td>
<td>4604.70</td>
<td></td>
<td>1</td>
<td>19.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td></td>
<td>179.74</td>
</tr>
<tr>
<td>2</td>
<td>91658.25</td>
<td>4605.85</td>
<td></td>
<td>1</td>
<td>19.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td></td>
<td>179.73</td>
</tr>
<tr>
<td>1</td>
<td>91688.05</td>
<td>4581.52</td>
<td></td>
<td>1</td>
<td>19.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>59.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>44.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>55.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total</td>
<td></td>
<td>179.83</td>
</tr>
</tbody>
</table>
6.4 Divide-and-Nest versus Virtual Nesting

In this section we compare the divide-and-nest approach in Chapter 4 with the airline industry’s practice, virtual nesting. We study the simple network example in Chapter 4, that consist of 3 cities (see Figure 6-1). The fare classes are created the same way as in Chapter 4. We use the EMSR model to compute the booking limits for all 3 fare buckets. Because of the aggregate nature of virtual nesting, we have to make some assumptions about about how virtual nesting is organized. In particular, we assume that lower classes book up first within each itinerary. Furthermore, we assume:

- On the Boston-Chicago itinerary, the lower class books first. On the Chicago-Los Angeles itinerary, the lower class books up first. No order is assumed between the two lower classes.

- The lower class of the Boston-Los Angeles itinerary books up before the higher class of the Boston-Chicago itinerary and before the higher class of the Chicago-Los Angeles itinerary.

- The higher class of the Boston-Los Angeles itinerary books up last.

With these assumptions, we have well-defined arrival orders for both legs. For the Boston-Chicago leg, the order of arrivals is:

1. The lower class of the Boston-Chicago itinerary
2. The lower class of the Boston-Los Angeles itinerary

3. The higher class of the Boston-Chicago itinerary

4. The higher class of the Boston-Los Angeles itinerary

For the Chicago-Los Angeles leg, the order of arrivals is:

1. The lower class of the Chicago-Los Angeles itinerary

2. The lower class of the Boston-Los Angeles itinerary

3. The higher class of the Chicago-Los Angeles itinerary

4. The higher class of the Boston-Los Angeles itinerary

Virtual fare buckets are defined according to fares only, regardless of the itineraries. In all the cases below, the buckets are created as shown in Table 6.18.

In the following numerical testing, the fares are fixed. We generate different cases by changing the capacity, the demands and the variability in demands. For these cases, we compare the virtual nesting approach and the divide-and-nest approach we developed in Chapter 4. In the divide-and-nest approach, we use the wait-and-see approach with individual itineraries.

For the virtual nesting approach, we apply the EMSR model to compute the booking limits on the virtual fare buckets. Given the booking limits and based on the assumptions about the orders of arrivals, we calculate the expected revenue.

The divide-and-nest approach is implemented precisely as described in the divide-and-nest algorithm (Algorithm 4.1) on page 81.

**Case 1**

Let us consider the case in Table 6.19. The table contains all the input needed for the virtual nesting approach and the divide-and-nest approach. In the end we see a 1.26% improvement of the divide-and-nest approach over the virtual nesting approach.

The computational results of Case 1 are shown in Table 6.20 and Table 6.21.
Table 6.18: Definition of virtual fare buckets

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Boston-Chicago Leg</th>
<th>Chicago-Los Angeles Leg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket 1</td>
<td>Boston-Chicago lower class</td>
<td>Chicago-Los Angeles lower class</td>
</tr>
<tr>
<td>Bucket 2</td>
<td>Boston-Chicago higher class</td>
<td>Chicago-Los Angeles higher class</td>
</tr>
<tr>
<td>Bucket 3</td>
<td>Boston-Los Angeles lower class</td>
<td>Boston-Los Angeles lower class</td>
</tr>
<tr>
<td></td>
<td>Boston-Los Angeles higher class</td>
<td>Boston-Los Angeles higher class</td>
</tr>
</tbody>
</table>

Table 6.19: Case 1: capacity = 100

<table>
<thead>
<tr>
<th></th>
<th>Boston-Chicago</th>
<th>Chicago-Los Angeles</th>
<th>Boston-Los Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Fare</td>
<td>Mean</td>
<td>Std. Dev.</td>
</tr>
<tr>
<td>1</td>
<td>400</td>
<td>30.0</td>
<td>9.5</td>
</tr>
<tr>
<td>2</td>
<td>150</td>
<td>40.0</td>
<td>13.0</td>
</tr>
</tbody>
</table>

Table 6.20: Virtual nesting booking limits for Case 1 – Revenue = 48092.33

<table>
<thead>
<tr>
<th></th>
<th>Boston-Chicago Leg</th>
<th>Chicago-Los Angeles Leg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket 1</td>
<td>17</td>
<td>28</td>
</tr>
<tr>
<td>Bucket 2</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>Bucket 3</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 6.21: Results of Case 1 using divide-and-nest – Revenue = 48139.62

<table>
<thead>
<tr>
<th></th>
<th>Boston-Chicago</th>
<th>Chicago-Los Angeles</th>
<th>Boston-Los Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cap. Allocated</td>
<td>57</td>
<td>57</td>
<td>41</td>
</tr>
<tr>
<td>Revenue</td>
<td>14443.84</td>
<td>14744.38</td>
<td>18951.40</td>
</tr>
</tbody>
</table>
Table 6.22: Virtual nesting booking limits for Case 2 – Revenue = 41274.53

<table>
<thead>
<tr>
<th>Boston-Chicago Leg</th>
<th>Chicago-Los Angeles Leg</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket 1</td>
<td>bucket 2</td>
</tr>
<tr>
<td>0</td>
<td>65</td>
</tr>
<tr>
<td>bucket 3</td>
<td>bucket 1</td>
</tr>
<tr>
<td>80</td>
<td>8</td>
</tr>
<tr>
<td>bucket 2</td>
<td>bucket 3</td>
</tr>
<tr>
<td>65</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 6.23: Results of Case 2 using divide-and-nest – Revenue = 41367.17

<table>
<thead>
<tr>
<th>Boston-Chicago</th>
<th>Chicago-Los Angeles</th>
<th>Boston-Los Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td>cap. allocated</td>
<td>revenue</td>
<td>cap. allocated</td>
</tr>
<tr>
<td>40</td>
<td>11942.03</td>
<td>40</td>
</tr>
<tr>
<td>revenue</td>
<td>cap. allocated</td>
<td>revenue</td>
</tr>
<tr>
<td>11482.42</td>
<td>40</td>
<td>17942.72</td>
</tr>
</tbody>
</table>

Case 2

We present another example, where demands and fares are identical with those in Case 1. The only difference is the capacity. In this case, we reduce the capacity of 100 in Case 1 by 20, to 80. We see an 1.85% improvement of the divide-and-nest approach over virtual nesting. The computational results of Case 2 are shown in Table 6.22 and Table 6.23.

Case 3

In this case we raise the capacity by 20 to 120. The computational results of Case 3 are shown in Table 6.24 and Table 6.25.

Table 6.24: Virtual nesting booking limits for Case 3 – Revenue = 54051.04

<table>
<thead>
<tr>
<th>Boston-Chicago Leg</th>
<th>Chicago-Los Angeles Leg</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket 1</td>
<td>bucket 2</td>
</tr>
<tr>
<td>37</td>
<td>105</td>
</tr>
<tr>
<td>bucket 3</td>
<td>bucket 1</td>
</tr>
<tr>
<td>120</td>
<td>48</td>
</tr>
<tr>
<td>bucket 2</td>
<td>bucket 3</td>
</tr>
<tr>
<td>105</td>
<td>120</td>
</tr>
</tbody>
</table>
Table 6.25: Results of Case 3 using divide-and-nest – Revenue = 53838.32

<table>
<thead>
<tr>
<th>Boston-Chicago</th>
<th>Chicago-Los Angeles</th>
<th>Boston-Los Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td>cap. allocated</td>
<td>revenue</td>
<td>cap. allocated</td>
</tr>
<tr>
<td>71</td>
<td>16214.67</td>
<td>71</td>
</tr>
</tbody>
</table>

Table 6.26: Case 4: capacity = 120

<table>
<thead>
<tr>
<th></th>
<th>Boston-Chicago</th>
<th>Chicago-Los Angeles</th>
<th>Boston-Los Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>fare</td>
<td>mean</td>
<td>std. dev.</td>
</tr>
<tr>
<td>1</td>
<td>400</td>
<td>30.0</td>
<td>7.5</td>
</tr>
<tr>
<td>2</td>
<td>150</td>
<td>40.0</td>
<td>13.0</td>
</tr>
</tbody>
</table>

Case 4

We modify the variability of the higher classes of the Boston-Chicago itinerary and the Chicago-Los Angeles itinerary. The modified numbers are underlined in Table 6.26.

The computational results of Case 4 are shown in Table 6.27 and Table 6.28. Compared to Case 3, the difference between the virtual nesting solution and the divide-and-nest solutions narrows, in terms of both absolute values and percentage.

Case 5

Finally we consider an extreme case – Case 5 as shown in Table 6.29. The computational results of Case 5 are shown in Table 6.30 and Table 6.31.

The divide-and-nest and virtual nesting approaches produce comparable solutions.

Table 6.27: Virtual nesting booking limits for Case 4 – Revenue = 54428.18

<table>
<thead>
<tr>
<th>Boston-Chicago Leg</th>
<th>Chicago-Los Angeles Leg</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket 1</td>
<td>bucket 2</td>
</tr>
<tr>
<td>38</td>
<td>105</td>
</tr>
</tbody>
</table>
Table 6.28: Results of Case 4 using divide-and-nest – Revenue = 54269.47

<table>
<thead>
<tr>
<th>Boston-Chicago</th>
<th>Chicago-Los Angeles</th>
<th>Boston-Los Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td>cap. allocated</td>
<td>revenue</td>
<td>cap. allocated</td>
</tr>
<tr>
<td>70</td>
<td>16330.21</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17030.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20908.83</td>
</tr>
</tbody>
</table>

Table 6.29: Case 5: capacity = 120

<table>
<thead>
<tr>
<th></th>
<th>Boston-Chicago</th>
<th>Chicago-Los Angeles</th>
<th>Boston-Los Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fare</td>
<td>mean</td>
<td>std. dev.</td>
</tr>
<tr>
<td>class 1</td>
<td>400</td>
<td>30.0</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>25.0</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>600</td>
<td>18.0</td>
<td>6.0</td>
</tr>
<tr>
<td>class 2</td>
<td>150</td>
<td>40.0</td>
<td>13.0</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>46.0</td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>32.0</td>
<td>10.0</td>
</tr>
</tbody>
</table>

Table 6.30: Virtual nesting booking limits for Case 5 – Revenue = 55029.38

<table>
<thead>
<tr>
<th>Boston-Chicago Leg</th>
<th>Chicago-Los Angeles Leg</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket 1</td>
<td>bucket 2</td>
</tr>
<tr>
<td>bucket 2</td>
<td>bucket 3</td>
</tr>
<tr>
<td>bucket 3</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>105</td>
</tr>
<tr>
<td>120</td>
<td>48</td>
</tr>
<tr>
<td>105</td>
<td>120</td>
</tr>
</tbody>
</table>

Table 6.31: Results of Case 5 using divide-and-nest – Revenue = 55260.92

<table>
<thead>
<tr>
<th>Boston-Chicago</th>
<th>Chicago-Los Angeles</th>
<th>Boston-Los Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td>cap. allocated</td>
<td>revenue</td>
<td>cap. allocated</td>
</tr>
<tr>
<td>68</td>
<td>16605.50</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>17311.10</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21344.32</td>
</tr>
</tbody>
</table>
The divide-and-nest approach captures the changes in variability well and makes the "right" adjustment. In this particular case it keeps lowering the capacity allocated to the two short itineraries and increasing the capacity allocated to the long itinerary. This is because as the variability of the two short legs decrease, fewer seats are needed to achieve the same revenue for the two short legs. On the contrary, virtual nesting makes little adjustment to the change.

Discussion and Conclusions

From the above cases, we draw the following conclusions. In most cases,

1. The divide-and-nest approach and the virtual nesting approach are comparable in most cases. In some cases, virtual nesting performs better and in other cases the divide-and-nest approach performs better.

2. Virtual nesting favors long-haul passengers.

3. The divide-and-nest approach adjusts its behavior when system parameters change (such as increasing demand variability, capacity and etc), while the virtual nesting approach is not as adaptive.

6.5 Dynamic Network Model versus Virtual Nesting

In this section we will study the benefits of network dynamic models compared with the virtual nesting approach, which is the airline industry’s standard practice.

Again, consider the network shown in Figure 6-1. For convenience, we will label the Boston-Chicago leg as leg 1 and the Chicago-Los Angeles leg as leg 2. Suppose on all three origin-destination (OD) pairs there are 3 fare classes. For each origin-destination/fare class (ODF) we specify its arrival probability $p$, fare $f$, and
Table 6.32: Example 1: horizon = 1500, 9 ODFs, capacity = 120

<table>
<thead>
<tr>
<th>ODF</th>
<th>( p )</th>
<th>( f )</th>
<th>( \delta_1 )</th>
<th>( \delta_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.010</td>
<td>1200.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.015</td>
<td>950.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.018</td>
<td>600.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0.013</td>
<td>950.00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.020</td>
<td>800.00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.021</td>
<td>550.00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.016</td>
<td>900.00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0.015</td>
<td>750.00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0.022</td>
<td>550.00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>total</td>
<td>0.150(^a)</td>
<td>0.097(^b)</td>
<td>0.096(^c)</td>
<td></td>
</tr>
</tbody>
</table>

\( a \sum p_i \)
\( b \sum p_i \cdot \delta_{i1} \)
\( c \sum p_i \cdot \delta_{i2} \)

its itinerary. Table 6.32 lists all the input data. The way an itinerary is defined is by two binary variables (columns 4 and 5 in Table 6.32). If the items in both columns are 1 for an ODF, i.e., it covers both leg 1 and leg 2, the ODF is of the Boston-Los Angeles itinerary. If for an ODF the item in column 4 is 1 and the item in column 5 is 0, i.e., it only covers leg 1, the ODF is of the Boston-Chicago itinerary. If for an ODF the item in column 4 is 0 and the item in column 5 is 1, i.e., it only covers leg 2, the ODF is of the Chicago-Los Angeles itinerary.

**Dynamic Model**

We first run the global dynamic model (5.6) and (5.7). The decision is whether to accept a class \( i \) fare request that occurs at time \( t \) given \( m \) and \( n \) seats remain on legs 1 and 2, respectively. In the network case unlike the single-leg case, threshold times do not exist any more. So, when solving the model we need to store the decision for each individual combination of \( m, n, t \) and \( i \) for later reference in the simulation stage.
Table 6.33: Seats requested and accepted by ODF

<table>
<thead>
<tr>
<th>ODF</th>
<th>reqs</th>
<th>acpts</th>
<th>$\delta_1$</th>
<th>$\delta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14.73</td>
<td>14.54</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>22.41</td>
<td>21.53</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>26.95</td>
<td>4.54</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>19.39</td>
<td>19.25</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>30.05</td>
<td>29.71</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>31.56</td>
<td>28.07</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>23.95</td>
<td>23.80</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>22.26</td>
<td>22.02</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>33.03</td>
<td>30.55</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.34: Seats requested and accepted by legs

<table>
<thead>
<tr>
<th>leg</th>
<th>reqs</th>
<th>acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>145.09</td>
<td>117.64</td>
</tr>
<tr>
<td>2</td>
<td>143.32</td>
<td>116.98</td>
</tr>
<tr>
<td>total</td>
<td>224.32</td>
<td>194.01</td>
</tr>
</tbody>
</table>

The results for the dynamic model is summarized in Tables 6.33, 6.34 and 6.35. Table 6.33 lists the requests made and accepted by ODF. Table 6.34 lists the requests made and accepted by legs. Table 6.35 shows the revenue, standard deviation and total number of passengers carried. Note that in Table 6.34, the row “total” lists the grand total number of requests made and accepted. It is not equal to the sum of the legs because that sum double-counts connecting passengers.

Table 6.35: Revenue, standard deviation and total number of passengers carried

<table>
<thead>
<tr>
<th>revenue</th>
<th>SD</th>
<th>total acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>152858.45</td>
<td>5702.74</td>
<td>194.01</td>
</tr>
</tbody>
</table>
Table 6.36: Definition of virtual fare classes

<table>
<thead>
<tr>
<th>Leg 1</th>
<th>Leg 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC1a</td>
<td>VC1</td>
</tr>
<tr>
<td>1</td>
<td>2, 7</td>
</tr>
<tr>
<td>VC2</td>
<td>VC2</td>
</tr>
<tr>
<td>2, 4</td>
<td>8</td>
</tr>
<tr>
<td>VC3</td>
<td>VC3</td>
</tr>
<tr>
<td>5</td>
<td>3, 9</td>
</tr>
<tr>
<td>VC4</td>
<td>VC4</td>
</tr>
<tr>
<td>3, 6</td>
<td></td>
</tr>
</tbody>
</table>

* VC: virtual class

Virtual Nesting

In order to run the virtual nesting algorithm for the previous case, we first need to define fare buckets (virtual fare classes) on both legs and define fares for the virtual fare classes. Finally, we need to aggregate demand for all the virtual fare classes.

On each leg, we define buckets according to the fares of the ODF’s that traverses the leg. In this particular case, we define 4 virtual fare classes on both legs. Table 6.36 lists the ODFs in each virtual fare classes on both legs.

The demands for the virtual classes are aggregated as a binomial \( B(n, p_{vc}) \), where \( n \) is the length of the (remaining) booking horizon and \( p_{vc} \) is the sum of arrival probabilities of all ODFs assigned to this virtual class. The virtual fare for a virtual class is simply the sum of the fares of all ODFs weighted by their arrival probabilities.

After preparing all the necessary inputs, we start running the EMSR model on the example. Virtual booking limits for the virtual classes on both legs are calculated, respectively. Then the virtual booking limits are passed to a simulator. The simulator generates requests for all ODFs according to their arrival probabilities. To decide whether a request for an ODF should be accepted or not, we first decide the itinerary, i.e., which leg(s) it traverses. Then it is determined which virtual class of the leg(s) it belongs to. Finally, all relevant virtual booking limits are checked to reach the decision as to accept or reject the request.

The results of simulation on virtual nesting are shown in Tables 6.37, 6.38 and 6.39. The difference in revenue is \$7,775, or 5.36% in favor of the dynamic model. The statistics on fare requests are more revealing. Overall, the difference in total number
Table 6.37: Seats requested and accepted by ODF

<table>
<thead>
<tr>
<th>ODF</th>
<th>reqs</th>
<th>acpts</th>
<th>δ₁</th>
<th>δ₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14.56</td>
<td>13.84</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>22.15</td>
<td>20.54</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>26.63</td>
<td>15.56</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>19.18</td>
<td>18.49</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>29.73</td>
<td>28.39</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>31.21</td>
<td>19.39</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>23.69</td>
<td>22.81</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>22.02</td>
<td>20.58</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>32.68</td>
<td>23.14</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.38: Seats requested and accepted by legs

<table>
<thead>
<tr>
<th>leg</th>
<th>reqs</th>
<th>acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>143.46</td>
<td>116.20</td>
</tr>
<tr>
<td>2</td>
<td>141.74</td>
<td>116.46</td>
</tr>
<tr>
<td>total</td>
<td>221.85</td>
<td>182.73</td>
</tr>
</tbody>
</table>

of passengers carried is about 11 seats. But the number of passengers carried on each leg are roughly identical. The principal observations is that under virtual nesting, more connecting passengers are carried than under the dynamic model. The more detailed breakdown shows that this indeed is the case. The dynamic model carries considerably more local customers and fewer connecting customers than the virtual nesting approach. The difference is most significant in the lowest classes of all three itineraries. The dynamic model takes about 11 fewer the lowest class requests of the Boston-Los Angeles itinerary than virtual nesting, but 9 and 7 more requests for the Boston-Chicago itinerary and the Chicago-Los Angeles itinerary. As a matter of fact, the sum of the lowest Boston-Chicago fare and the lowest Chicago-Los Angeles fare ($1,100) is much larger than the lowest Boston-Los Angeles fare ($600).
Table 6.39: Revenue, standard deviation and total number of passengers carried

<table>
<thead>
<tr>
<th>revenue</th>
<th>SD</th>
<th>total acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>145083.80</td>
<td>5188.45</td>
<td>182.73</td>
</tr>
</tbody>
</table>

Re-Optimization of Virtual Nesting

As in the single-leg case, we run the virtual nesting algorithm in a re-optimization mode. Re-optimization points are specified on the booking horizon. Whenever a simulation reaches a re-optimization point, the future demands for virtual fare classes are re-aggregated. In particular at re-optimization \( i \) (with \( i = 1 \) being the initial optimization) the demand for a virtual fare class can be seen as \( B(T - (i - 1) \cdot d, p_{vc}) \), where \( T \) is the entire booking horizon, \( d \) is the interval between two consecutive re-optimizations and \( i \) represents \( i \)th re-optimization (with zero being the initial optimization). Virtual booking limits are re-calculated based on the re-aggregated demands and the remaining capacities on both legs. The interval between consecutive re-optimizations is determined by the following equation:

\[
\text{update interval} = \left\lfloor \frac{\text{booking horizon}}{\text{number of re-optimizations}} \right\rfloor.
\]

In our case, 41 runs with different number of re-optimizations are made. For the first 10 runs, the number of re-optimizations increases by 1, i.e., the number of re-optimizations are from 1 up to 10. Beginning from 11th run the number of re-optimization increases by 5. A brief summary of all the runs is in Table 6.40.

The results show that when the re-optimization is at 23, the revenue attains its maximum value, 147810.60, with a total of 185.69 requests accepted. The difference in revenue is $5,048. or 3.41%, in favor of the dynamic model. In addition, the dynamic model still yields a load about 10 passengers more than virtual nesting with re-optimization. Table 6.41 and Table 6.42 contain the breakdown of the requests made and accepted and show that re-optimization has limited effect in correcting the
Table 6.40: Summary of re-optimization results

<table>
<thead>
<tr>
<th>interval</th>
<th>revenue</th>
<th>SD(^a)</th>
<th>acpts(^b)</th>
<th>acpts(^c)</th>
<th>total(^d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>750</td>
<td>146534.20</td>
<td>4726.27</td>
<td>117.18</td>
<td>117.21</td>
<td>184.13</td>
</tr>
<tr>
<td>500</td>
<td>146750.05</td>
<td>4458.81</td>
<td>117.64</td>
<td>117.67</td>
<td>184.58</td>
</tr>
<tr>
<td>375</td>
<td>147074.10</td>
<td>4446.69</td>
<td>117.67</td>
<td>117.64</td>
<td>184.64</td>
</tr>
<tr>
<td>300</td>
<td>146999.35</td>
<td>4305.95</td>
<td>117.59</td>
<td>117.77</td>
<td>184.56</td>
</tr>
<tr>
<td>250</td>
<td>147420.45</td>
<td>4225.39</td>
<td>118.16</td>
<td>118.07</td>
<td>185.35</td>
</tr>
<tr>
<td>214</td>
<td>147432.75</td>
<td>4145.70</td>
<td>118.19</td>
<td>117.98</td>
<td>185.36</td>
</tr>
<tr>
<td>187</td>
<td>147367.00</td>
<td>4213.03</td>
<td>117.96</td>
<td>117.78</td>
<td>184.92</td>
</tr>
<tr>
<td>166</td>
<td>147331.10</td>
<td>4181.51</td>
<td>118.13</td>
<td>117.99</td>
<td>185.23</td>
</tr>
<tr>
<td>150</td>
<td>147417.85</td>
<td>4134.19</td>
<td>118.34</td>
<td>118.01</td>
<td>185.22</td>
</tr>
<tr>
<td>100</td>
<td>147661.25</td>
<td>4093.65</td>
<td>118.56</td>
<td>118.13</td>
<td>185.48</td>
</tr>
<tr>
<td>75</td>
<td>147647.90</td>
<td>4093.21</td>
<td>118.43</td>
<td>118.20</td>
<td>185.40</td>
</tr>
<tr>
<td>60</td>
<td>147764.20</td>
<td>3973.35</td>
<td>118.67</td>
<td>118.34</td>
<td>185.75</td>
</tr>
<tr>
<td>50</td>
<td>147767.80</td>
<td>3993.59</td>
<td>118.64</td>
<td>118.34</td>
<td>185.74</td>
</tr>
<tr>
<td>42</td>
<td>147675.70</td>
<td>4026.98</td>
<td>118.58</td>
<td>118.27</td>
<td>185.52</td>
</tr>
<tr>
<td>37</td>
<td>147801.10</td>
<td>4008.49</td>
<td>118.61</td>
<td>118.42</td>
<td>185.83</td>
</tr>
<tr>
<td>33</td>
<td>147728.20</td>
<td>3992.44</td>
<td>118.61</td>
<td>118.31</td>
<td>185.73</td>
</tr>
<tr>
<td>30</td>
<td>147799.50</td>
<td>3923.91</td>
<td>118.64</td>
<td>118.39</td>
<td>185.79</td>
</tr>
<tr>
<td>27</td>
<td>147747.50</td>
<td>3998.81</td>
<td>118.55</td>
<td>118.39</td>
<td>185.64</td>
</tr>
<tr>
<td>25</td>
<td>147782.15</td>
<td>3990.93</td>
<td>118.57</td>
<td>118.36</td>
<td>185.68</td>
</tr>
<tr>
<td>23</td>
<td>147810.60</td>
<td>4007.05</td>
<td>118.58</td>
<td>118.33</td>
<td>185.69</td>
</tr>
<tr>
<td>21</td>
<td>147792.40</td>
<td>4030.96</td>
<td>118.62</td>
<td>118.35</td>
<td>185.78</td>
</tr>
<tr>
<td>20</td>
<td>147717.05</td>
<td>3947.15</td>
<td>118.55</td>
<td>118.34</td>
<td>185.67</td>
</tr>
<tr>
<td>18</td>
<td>147700.35</td>
<td>4022.07</td>
<td>118.54</td>
<td>118.32</td>
<td>185.65</td>
</tr>
<tr>
<td>17</td>
<td>147702.45</td>
<td>4010.33</td>
<td>118.61</td>
<td>118.34</td>
<td>185.65</td>
</tr>
<tr>
<td>16</td>
<td>147796.45</td>
<td>4081.96</td>
<td>118.67</td>
<td>118.35</td>
<td>185.77</td>
</tr>
<tr>
<td>15</td>
<td>147801.70</td>
<td>3999.80</td>
<td>118.61</td>
<td>118.34</td>
<td>185.78</td>
</tr>
<tr>
<td>14</td>
<td>147774.55</td>
<td>3981.42</td>
<td>118.63</td>
<td>118.34</td>
<td>185.76</td>
</tr>
<tr>
<td>13</td>
<td>147776.40</td>
<td>3984.58</td>
<td>118.61</td>
<td>118.31</td>
<td>185.67</td>
</tr>
<tr>
<td>12</td>
<td>147804.10</td>
<td>3962.46</td>
<td>118.56</td>
<td>118.33</td>
<td>185.79</td>
</tr>
<tr>
<td>11</td>
<td>147735.55</td>
<td>3937.24</td>
<td>118.56</td>
<td>118.34</td>
<td>185.76</td>
</tr>
</tbody>
</table>

\(^a\) standard deviation of revenue  
\(^b\) number of passengers carried on leg 1  
\(^c\) number of passengers carried on leg 2  
\(^d\) total number of passengers carried
Table 6.40: Summary of re-optimization results (continued)

<table>
<thead>
<tr>
<th>interval</th>
<th>revenue</th>
<th>SD</th>
<th>acpts1</th>
<th>acpts2</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>147612.50</td>
<td>3991.02</td>
<td>118.53</td>
<td>118.33</td>
<td>185.59</td>
</tr>
<tr>
<td>9</td>
<td>147742.40</td>
<td>4027.41</td>
<td>118.56</td>
<td>118.31</td>
<td>185.69</td>
</tr>
<tr>
<td>8</td>
<td>147683.55</td>
<td>4107.38</td>
<td>118.58</td>
<td>118.39</td>
<td>185.64</td>
</tr>
<tr>
<td>7</td>
<td>147748.80</td>
<td>3997.77</td>
<td>118.57</td>
<td>118.33</td>
<td>185.74</td>
</tr>
<tr>
<td>6</td>
<td>147772.25</td>
<td>4071.16</td>
<td>118.55</td>
<td>118.32</td>
<td>185.69</td>
</tr>
<tr>
<td>5</td>
<td>147749.70</td>
<td>4067.20</td>
<td>118.56</td>
<td>118.32</td>
<td>185.68</td>
</tr>
<tr>
<td>4</td>
<td>147733.05</td>
<td>3987.44</td>
<td>118.58</td>
<td>118.34</td>
<td>185.76</td>
</tr>
<tr>
<td>3</td>
<td>147720.90</td>
<td>3979.83</td>
<td>118.57</td>
<td>118.35</td>
<td>185.73</td>
</tr>
<tr>
<td>2</td>
<td>147647.05</td>
<td>3915.82</td>
<td>118.52</td>
<td>118.33</td>
<td>185.56</td>
</tr>
<tr>
<td>1</td>
<td>147659.30</td>
<td>3924.47</td>
<td>118.49</td>
<td>118.37</td>
<td>185.56</td>
</tr>
</tbody>
</table>

**tendency of virtual nesting to favor long-haul requests.**

We end the discussion with a more extreme example, in which the capacity is considerably tight relative to demands. The problem is stated in Table 6.43. The dynamic solutions are summarized in Tables 6.44, 6.45 and 6.46. The purpose of this example is to show that when capacities are tight the global dynamic model performs even better than virtual nesting with re-optimization.

For virtual nesting, virtual fare classes are defined in Table 6.47. The results without re-optimization are summarized in Tables 6.48, 6.49 and 6.50. The dynamic model enjoys $6,791, or 8.37% advantage over the EMSR model in terms of revenue. In addition the number of passengers carried differ by almost 20 in favor of the dynamic model. A breakdown of requests shows that local requests are more severely discriminated against in this case where capacity is tight.

Next we introduce re-optimization. As in the last example, 41 test runs (including the initial optimization) are made. The results are shown in Table 6.51. In the example, when the re-optimization interval is 1, virtual nesting achieves its maximum revenue, 84407. The revenue generated by dynamic model is $3,530, or 4.18% better than that generated by virtual nesting with re-optimization. Moreover, it maintains an advantage of almost 12 in the number of passengers carried. Again, re-optimization
Table 6.41: Seats requested and accepted by ODF, update interval = 23

<table>
<thead>
<tr>
<th>ODF</th>
<th>reqs</th>
<th>acpts</th>
<th>$\delta_1$</th>
<th>$\delta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14.72</td>
<td>14.52</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>22.40</td>
<td>21.55</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>26.92</td>
<td>15.15</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>19.38</td>
<td>18.90</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>30.03</td>
<td>27.95</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>31.54</td>
<td>20.50</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>23.94</td>
<td>23.41</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>22.24</td>
<td>20.62</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>33.01</td>
<td>23.09</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.42: Seats requested and accepted by legs, update interval = 23

<table>
<thead>
<tr>
<th>leg</th>
<th>reqs</th>
<th>acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>144.99</td>
<td>118.58</td>
</tr>
<tr>
<td>2</td>
<td>143.22</td>
<td>118.33</td>
</tr>
<tr>
<td>total</td>
<td>224.17</td>
<td>185.69</td>
</tr>
</tbody>
</table>

Table 6.43: Example 2: horizon = 1500, 9 ODFs, capacity = 80

<table>
<thead>
<tr>
<th>ODF</th>
<th>$p$</th>
<th>$f$</th>
<th>$\delta_1$</th>
<th>$\delta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.007</td>
<td>1200.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.013</td>
<td>950.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.010</td>
<td>550.00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0.009</td>
<td>950.00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.014</td>
<td>550.00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.013</td>
<td>400.00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.007</td>
<td>950.00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0.013</td>
<td>550.00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0.014</td>
<td>400.00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>total</td>
<td>0.100$^a$</td>
<td>0.066$^b$</td>
<td>0.064$^c$</td>
<td></td>
</tr>
</tbody>
</table>

$^a \sum p_i$

$^b \sum p_i \cdot \delta_{1i}$

$^c \sum p_i \cdot \delta_{2i}$
Table 6.44: Example 2: Seats requested and accepted by ODF

<table>
<thead>
<tr>
<th>ODF</th>
<th>reqs</th>
<th>acpts</th>
<th>δ₁</th>
<th>δ₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.40</td>
<td>10.28</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>19.15</td>
<td>18.57</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>15.01</td>
<td>2.33</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>13.37</td>
<td>13.28</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>20.80</td>
<td>20.14</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>19.48</td>
<td>14.03</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>10.45</td>
<td>10.38</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>19.49</td>
<td>19.03</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>20.99</td>
<td>17.61</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.45: Seats requested and accepted by legs

<table>
<thead>
<tr>
<th>leg</th>
<th>reqs</th>
<th>acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98.21</td>
<td>78.63</td>
</tr>
<tr>
<td>2</td>
<td>95.50</td>
<td>78.20</td>
</tr>
<tr>
<td>total</td>
<td>149.15</td>
<td>125.65</td>
</tr>
</tbody>
</table>

Table 6.46: Revenue, standard deviation and total number of passengers carried

<table>
<thead>
<tr>
<th>revenue</th>
<th>SD</th>
<th>total acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>87937.40</td>
<td>4174.22</td>
<td>125.65</td>
</tr>
</tbody>
</table>

Table 6.47: Definition of Virtual Fare Classes for Example 2

<table>
<thead>
<tr>
<th>Leg 1</th>
<th>Leg 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC₁ᵃ</td>
<td>VC₁</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2, 4</td>
<td>2, 7</td>
</tr>
<tr>
<td>3, 5</td>
<td>3, 8</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

ᵃ VC: virtual class
Table 6.48: Example 2: Seats requested and accepted by ODF

<table>
<thead>
<tr>
<th>ODF</th>
<th>reqs</th>
<th>acpts</th>
<th>$\delta_1$</th>
<th>$\delta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.41</td>
<td>10.12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>19.16</td>
<td>18.49</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>15.02</td>
<td>13.34</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>13.37</td>
<td>13.10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>20.81</td>
<td>19.22</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>19.48</td>
<td>0.00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>10.45</td>
<td>10.29</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>19.50</td>
<td>18.35</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>21.00</td>
<td>3.00</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.49: Example 2: Seats requested and accepted by legs

<table>
<thead>
<tr>
<th>leg</th>
<th>reqs</th>
<th>acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98.25</td>
<td>74.28</td>
</tr>
<tr>
<td>2</td>
<td>95.53</td>
<td>73.61</td>
</tr>
<tr>
<td>total</td>
<td>149.20</td>
<td>105.93</td>
</tr>
</tbody>
</table>

Table 6.50: Revenue, standard deviation and total number of passengers carried

<table>
<thead>
<tr>
<th>revenue</th>
<th>SD</th>
<th>total acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>82763.85</td>
<td>4515.86</td>
<td>105.93</td>
</tr>
</tbody>
</table>
has done little to correct virtual nesting's tendency to overly favor long-haul requests, as the breakdown in Table 6.52 and Table 6.53.

### 6.6 Quadratic Approximation of Dynamic Models

A major difficulty with our dynamic model is the large dimension of the state space. In Chapter 5, we proposed a quadratic approximation for estimating the revenue function. In this section we will run some simple example to show that the approximation works reasonably well.

We consider a 3-class example. Table 6.54 lists arrival probabilities, fares and probabilities that a request will eventually show up.

The test is conducted as follows. First of all, the exact dynamic model is run to obtain the exact expected revenues for combinations of bookings for the 3 classes at time $t$, i.e., the revenue we can expect between time $t$ and time 0 given that certain numbers of seats have been booked for the 3 classes, respectively. Algebraically, the revenue is the function of combination $(n_1, n_2, n_3, t)$, where $n_i$ is the bookings for class $i$, $i = 1, 2, 3$ and $t$ is time.

The estimated revenues are computed as follows. We first randomly generate a specific number of data points which will later be used for least-square fitting. Revenues at these data points for $t = 1$ are calculated (exactly). Matlab is then called to perform a least-square fitting on the data points and the corresponding revenues at $t = 1$. The least-square fitting will yield a quadratic form, which will serve as the approximate revenue function for $t = 1$. This quadratic form is then used, together with the dynamic programming recursion, to approximate the revenues at the data points for $t = 2$. The data points and the corresponding approximated revenues for $t = 2$ are input to Matlab for a least-square fitting. The quadratic form resulting from the least-square fitting will serve as the approximate revenue function
Table 6.51: Summary of re-optimization results

<table>
<thead>
<tr>
<th>inter.val</th>
<th>revenue</th>
<th>SD</th>
<th>acpts(^b)</th>
<th>acpts(^c)</th>
<th>total(^d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>750</td>
<td>82763.85</td>
<td>4515.85</td>
<td>76.00</td>
<td>75.61</td>
<td>109.71</td>
</tr>
<tr>
<td>500</td>
<td>83137.85</td>
<td>4086.18</td>
<td>76.25</td>
<td>76.30</td>
<td>110.76</td>
</tr>
<tr>
<td>375</td>
<td>83253.55</td>
<td>4142.27</td>
<td>76.38</td>
<td>76.28</td>
<td>110.87</td>
</tr>
<tr>
<td>300</td>
<td>83541.40</td>
<td>3981.80</td>
<td>76.66</td>
<td>76.85</td>
<td>111.59</td>
</tr>
<tr>
<td>250</td>
<td>83737.80</td>
<td>3796.32</td>
<td>76.95</td>
<td>77.01</td>
<td>112.20</td>
</tr>
<tr>
<td>214</td>
<td>83947.75</td>
<td>3850.52</td>
<td>77.33</td>
<td>77.00</td>
<td>112.43</td>
</tr>
<tr>
<td>187</td>
<td>3960.45</td>
<td>3851.65</td>
<td>77.24</td>
<td>77.12</td>
<td>112.39</td>
</tr>
<tr>
<td>166</td>
<td>83986.35</td>
<td>3782.32</td>
<td>77.35</td>
<td>77.15</td>
<td>112.55</td>
</tr>
<tr>
<td>150</td>
<td>84049.25</td>
<td>3750.53</td>
<td>77.39</td>
<td>77.22</td>
<td>112.67</td>
</tr>
<tr>
<td>100</td>
<td>84056.65</td>
<td>3676.38</td>
<td>77.43</td>
<td>77.30</td>
<td>112.80</td>
</tr>
<tr>
<td>75</td>
<td>84291.80</td>
<td>3661.55</td>
<td>77.78</td>
<td>77.47</td>
<td>113.26</td>
</tr>
<tr>
<td>60</td>
<td>84240.40</td>
<td>3595.31</td>
<td>77.71</td>
<td>77.43</td>
<td>113.19</td>
</tr>
<tr>
<td>50</td>
<td>84327.35</td>
<td>3516.42</td>
<td>77.80</td>
<td>77.54</td>
<td>113.44</td>
</tr>
<tr>
<td>42</td>
<td>84297.85</td>
<td>3602.79</td>
<td>77.73</td>
<td>77.50</td>
<td>113.28</td>
</tr>
<tr>
<td>37</td>
<td>84312.40</td>
<td>3586.02</td>
<td>77.78</td>
<td>77.52</td>
<td>113.34</td>
</tr>
<tr>
<td>33</td>
<td>84317.55</td>
<td>3595.94</td>
<td>77.84</td>
<td>77.58</td>
<td>113.39</td>
</tr>
<tr>
<td>30</td>
<td>84292.45</td>
<td>3579.84</td>
<td>77.78</td>
<td>77.57</td>
<td>113.30</td>
</tr>
<tr>
<td>27</td>
<td>84357.00</td>
<td>3560.97</td>
<td>77.84</td>
<td>77.67</td>
<td>113.50</td>
</tr>
<tr>
<td>25</td>
<td>84398.60</td>
<td>3575.76</td>
<td>77.89</td>
<td>77.67</td>
<td>113.64</td>
</tr>
<tr>
<td>23</td>
<td>84370.20</td>
<td>3556.83</td>
<td>77.83</td>
<td>77.68</td>
<td>113.49</td>
</tr>
<tr>
<td>21</td>
<td>84385.25</td>
<td>3584.29</td>
<td>77.89</td>
<td>77.70</td>
<td>113.62</td>
</tr>
<tr>
<td>20</td>
<td>84334.55</td>
<td>3599.99</td>
<td>77.83</td>
<td>77.62</td>
<td>113.45</td>
</tr>
<tr>
<td>18</td>
<td>84349.30</td>
<td>3560.91</td>
<td>77.87</td>
<td>77.60</td>
<td>113.46</td>
</tr>
<tr>
<td>17</td>
<td>84378.15</td>
<td>3572.62</td>
<td>77.90</td>
<td>77.66</td>
<td>113.55</td>
</tr>
<tr>
<td>16</td>
<td>84318.40</td>
<td>3629.47</td>
<td>77.86</td>
<td>77.58</td>
<td>113.41</td>
</tr>
<tr>
<td>15</td>
<td>84367.10</td>
<td>3586.50</td>
<td>77.90</td>
<td>77.66</td>
<td>113.58</td>
</tr>
<tr>
<td>14</td>
<td>84365.25</td>
<td>3585.78</td>
<td>77.91</td>
<td>77.65</td>
<td>113.58</td>
</tr>
<tr>
<td>13</td>
<td>84331.80</td>
<td>3569.78</td>
<td>77.83</td>
<td>77.66</td>
<td>113.53</td>
</tr>
<tr>
<td>12</td>
<td>84435.35</td>
<td>3497.54</td>
<td>77.92</td>
<td>77.73</td>
<td>113.70</td>
</tr>
<tr>
<td>11</td>
<td>84380.65</td>
<td>3550.91</td>
<td>77.91</td>
<td>77.70</td>
<td>113.56</td>
</tr>
</tbody>
</table>

\(^a\) standard deviation of revenue  
\(^b\) number of passengers carried on leg 1  
\(^c\) number of passengers carried on leg 2  
\(^d\) total number of passengers carried
for $t = 2$. The approximate revenue function for $t = 2$ is used, together with the dynamic programming recursion, to approximate the revenues at the data points for $t = 3$. The data points and the corresponding approximated revenues for $t = 3$ are input to Matlab for a least-square fitting. The quadratic form resulting from the least-square fitting will serve as the approximate revenue function for $t = 3$. This process continues until $t$ is equal the booking horizon.

We choose Matlab as our least-square fitting tool because it features a nice interface with the C programming language in which all our implementation is done.

In Table 6.55, we list and compare the revenues for data points $(0, 0, 0, t)$. These data points are particularly interesting because in a distributed dynamic situation, i.e., the dynamic model is used within each itinerary while network capacities are assigned for individual itineraries using network flow algorithms. The revenues at these data points will be used for initial allocation of capacities.

We observe the agreement between the exact revenues and the approximated revenues.
Table 6.51: Summary of re-optimization results (continued)

<table>
<thead>
<tr>
<th>interval</th>
<th>revenue</th>
<th>SD</th>
<th>acpts1</th>
<th>acpts2</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>84368.85</td>
<td>3540.05</td>
<td>77.88</td>
<td>77.66</td>
<td>113.55</td>
</tr>
<tr>
<td>9</td>
<td>84378.10</td>
<td>3516.17</td>
<td>77.89</td>
<td>77.65</td>
<td>113.55</td>
</tr>
<tr>
<td>8</td>
<td>84350.95</td>
<td>3562.29</td>
<td>77.89</td>
<td>77.64</td>
<td>113.53</td>
</tr>
<tr>
<td>7</td>
<td>84375.15</td>
<td>3562.91</td>
<td>77.93</td>
<td>77.67</td>
<td>113.58</td>
</tr>
<tr>
<td>6</td>
<td>84363.35</td>
<td>3570.60</td>
<td>77.90</td>
<td>77.65</td>
<td>113.52</td>
</tr>
<tr>
<td>5</td>
<td>84383.50</td>
<td>3498.32</td>
<td>77.88</td>
<td>77.67</td>
<td>113.59</td>
</tr>
<tr>
<td>4</td>
<td>84356.45</td>
<td>3517.04</td>
<td>77.87</td>
<td>77.62</td>
<td>113.57</td>
</tr>
<tr>
<td>3</td>
<td>84390.70</td>
<td>3513.10</td>
<td>77.88</td>
<td>77.68</td>
<td>113.60</td>
</tr>
<tr>
<td>2</td>
<td>84404.20</td>
<td>3482.33</td>
<td>77.90</td>
<td>77.70</td>
<td>113.63</td>
</tr>
<tr>
<td>1</td>
<td>84407.70</td>
<td>3466.60</td>
<td>77.91</td>
<td>77.71</td>
<td>113.65</td>
</tr>
</tbody>
</table>

Table 6.52: Seats requested and accepted by ODF, update interval = 1

<table>
<thead>
<tr>
<th>ODF</th>
<th>reqs</th>
<th>acpts</th>
<th>δ₁</th>
<th>δ₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.45</td>
<td>10.36</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>19.23</td>
<td>18.68</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>15.07</td>
<td>12.93</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>13.42</td>
<td>13.17</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>20.89</td>
<td>18.85</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>19.57</td>
<td>3.92</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>10.48</td>
<td>10.31</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>19.56</td>
<td>18.00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>21.07</td>
<td>7.43</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.53: Seats requested and accepted by legs, update interval = 1

<table>
<thead>
<tr>
<th>leg</th>
<th>reqs</th>
<th>acpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98.62</td>
<td>77.91</td>
</tr>
<tr>
<td>2</td>
<td>95.86</td>
<td>77.71</td>
</tr>
<tr>
<td>total</td>
<td>149.74</td>
<td>113.65</td>
</tr>
</tbody>
</table>
Table 6.54: A 3-class example: horizon = 100, capacity = 80

<table>
<thead>
<tr>
<th>class</th>
<th>arr. prob.</th>
<th>fare</th>
<th>showup prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.15</td>
<td>500.0</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>400.0</td>
<td>0.20</td>
</tr>
<tr>
<td>3</td>
<td>0.40</td>
<td>300.0</td>
<td>0.35</td>
</tr>
</tbody>
</table>

Table 6.55: Comparison of exact and approximated revenues for (0,0,0,t): number of data points is 100

<table>
<thead>
<tr>
<th>time</th>
<th>exact rev.</th>
<th>approx. rev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2926.572</td>
<td>2868.164</td>
</tr>
<tr>
<td>20</td>
<td>5811.213</td>
<td>5786.192</td>
</tr>
<tr>
<td>30</td>
<td>8654.854</td>
<td>8653.202</td>
</tr>
<tr>
<td>40</td>
<td>11458.392</td>
<td>11470.465</td>
</tr>
<tr>
<td>50</td>
<td>14222.698</td>
<td>14239.215</td>
</tr>
<tr>
<td>60</td>
<td>16948.613</td>
<td>16960.651</td>
</tr>
<tr>
<td>70</td>
<td>19636.952</td>
<td>19635.932</td>
</tr>
<tr>
<td>80</td>
<td>22288.507</td>
<td>22266.187</td>
</tr>
<tr>
<td>90</td>
<td>24903.848</td>
<td>24852.509</td>
</tr>
<tr>
<td>100</td>
<td>27406.022</td>
<td>27395.961</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusions

7.1 Contributions of this Dissertation

The goal of this research is to conduct a comprehensive study on the airline yield management problem, with emphasis on its dynamic and network aspects. We have proposed models and algorithms for static single-leg models (with and without cancelations), single-leg dynamic models (with and without cancelations), static network models (mainly without cancelations) and dynamic network models (with and without cancelations). In all cases we provided strong support for the models and algorithms with extensive computational results.

In the area of single-leg static models, we developed the “wait-and-see” model. Its advantage is guaranteed by the underlying idea of postponing decisions as much as possible. We proved the optimality of the wait-and-see approach. We also provide an easily implementable algorithm that uses the wait-and-see approach to compute the optimal booking limits. Furthermore, we conducted extensive computational experiments. We found that the wait-and-see approach consistently performs better than the industry practice, the EMSR model, both in terms of the total revenue and the load factor.

We also developed a single-leg static model which directly incorporates cancela-
tions and overbookings by introducing a penalty cost for overbookings. Traditional approaches have been along the line of scaling the capacity. To accurately address the issue of cancelations and overbookings, one must associate with overbookings a penalty cost. We established a model based on the wait-and-see approach. In order to be able to practically implement the model, we introduced an approximation scheme. Through our computational testing, we have found that the approximation yields results close to the exact solutions.

The wait-and-see approach was extended to the network case and a hierarchical approach was developed. We were able to formulate a problem as a mathematical programming problem and transform it into a convex flow problem, for which a fast algorithm exists. We implemented the hierarchical approach for the case without cancelations and overbookings and compared it against the industry standard approach "virtual nesting". We have found that the hierarchical approach yields comparable results to the virtual nesting approach.

In our experiments we have found that existing models used in the industry are overly conservative in that they excessively favor high-fare classes. Using our models we have observed that in most cases we achieve better load factors and in many cases higher total revenues.

For dynamic models without cancelations and overbookings, we were able to establish solutions in the form of "threshold times", which not only lead to nice analytical results, but also greatly simplify the solutions themselves. In the case of cancelations and overbookings, we conjecture that there exist "threshold intervals".

We carried out extensive experimental study on the dynamic models. For the single-leg case, the results have shown that substantial improvements can be gained by using the dynamic models. The improvements are seen both in the total revenue and in the load factor. We also found that by re-applying a static model periodically one can narrow the difference between the results of the dynamic models and the static model.
Chapter 7. Conclusions

We also developed dynamic models for the network case. We compared the global dynamic model with the industry practice "virtual nesting". The difference in this case is more significant than in the single-leg case. Moreover, we found that although re-applying the virtual nesting approach periodically reduces the difference between the dynamic model results and the virtual nesting approach results, the reduction is lower than in the single-leg case. This clearly implies that the dynamic model is intrinsically a better model for handling the network case.

For dynamic network models, we also proposed a approximation approach to tackle the dimensionality difficulty in solving the dynamic program. The approximation is essentially a quadratic approximation scheme. Our preliminary computational experiments have produced results that demonstrate a close agreement between the approximated solutions and the exact solutions.

7.2 Future Research Directions

We summarize below the most promising research directions:

1. Development of further analytical insight on dynamic network models.

   An analytical study was done on single-leg dynamic models. Similar study needs to be done for network dynamic models. Given the difficulty associated with network dynamic models, the primary goal of an analytical study is not to find closed-form solutions, instead it is to discover useful properties and achieve insights on the nature of the solutions.

2. Better understanding of the quadratic approximation used.

   The quadratic approximation is at its preliminary stage. Issues that need further study include the number of data points needed, general physical distribution of data points generated.
3. Further experimentation using real data.

Our numerical experiments were carried out with manufactured data.
It is rather interesting to conduct the experiments using real data.
Appendix A

Proofs of Theorems and Propositions

A.1 Proof of Theorem 2.5

Theorem 2.5: $G(\cdot)$ is concave.

Proof: Let

\[
G(S) = g(S, t^*), \\
G(S + 1) = g(S + 1, u^*), \\
G(S - 1) = g(S - 1, v^*).
\]

By Proposition 2.4, $u^* = t^*$ or $u^* = t^* + 1$.

If $u^* = 0$, then $t^* = v^* = 0$. Trivially, $G(S + 1) = E(S + 1)$, $G(S) = E(S)$ and $G(S - 1) = E(S - 1)$.

In what follows, $u^* > 0$. 

140
If \( u^* = t^* \), then

\[
G(S + 1) - G(S) = g(S + 1, u^*) - g(S, u^*)
= \sum_{x=0}^{u^*} [E(S + 1 - x) - E(S - x)] \cdot p(x) + \sum_{x=u^*+1}^{\infty} [E(S + 1 - u^*) - E(S - u^*)] \cdot p(x)
\]

We consider two cases:

1. \( t^* = v^* (= u^*) \). In this case,

\[
G(S) - G(S - 1) = g(S, u^*) - g(S - 1, u^*)
= \sum_{x=0}^{u^*} [E(S - x) - E(S - 1 - x)] \cdot p(x) + \sum_{x=u^*+1}^{\infty} [E(S - u^*) - E(S - 1 - u^*)] \cdot p(x).
\]

From concavity of \( E(\cdot) \), we have

\[
E(S + 1 - x) - E(S - x) \leq E(S - x) - E(S - 1 - x),
E(S + 1 - u) - E(S - u) \leq E(S - u) - E(S - 1 - u).
\]

This implies

\[
G(S + 1) - G(S) \leq G(S) - G(S - 1).
\]

2. \( t^* = v^* + 1 \). In this case,

\[
G(S) - G(S - 1) = g(S, t^*) - g(S - 1, v^*) = g(S, v^* + 1) - g(S - 1, v^*)
= \sum_{x=0}^{v^*+1} [f \cdot x + E(S - x)] \cdot p(x) + \sum_{x=v^*+2}^{\infty} [f \cdot (v^* + 1) + E(S - 1 - v^*)] \cdot p(x) -
\sum_{x=0}^{v^*} [f \cdot x + E(S - 1 - x)] \cdot p(x) - \sum_{x=v^*+1}^{\infty} [f \cdot v^* + E(S - 1 - v^*)] \cdot p(x)
= \sum_{x=0}^{v^*} [E(S - x) - E(S - 1 - x)] \cdot p(x) + f \sum_{x=v^*+1}^{\infty} p(x).
\]
Appendix A. Proofs of Theorems and Propositions

\[ G(S + 1) - G(S) = g(S + 1, u^*) - g(S, t^*) = g(S + 1, u^*) - g(S, u^*) \]

\[ = \sum_{x=0}^{u^*} [E(S + 1 - x) - E(S - x)] \cdot p(x) + \sum_{x=u^*+1}^{\infty} [E(S + 1 - u^*) - E(S - u^*)] \cdot p(x) \]

\[ = \sum_{x=0}^{v^*} [E(S + 1 - x) - E(S - x)] \cdot p(x) + \sum_{x=v^*+1}^{\infty} [E(S - v^*) - E(S - 1 - v^*)] \cdot p(x). \]

From concavity of \( E(\cdot) \),

\[ E(S + 1 - x) - E(S - x) \leq E(S - x) - E(S - 1 - x). \]

From optimality of \( v^* \) for \( g(S - 1, \cdot) \),

\[ E(S - v^*) - E(S - 1 - v^*) \leq f. \]

Putting the above together, we have

\[ G(S + 1) - G(S) \leq G(S) - G(S - 1). \]

We have just proved concavity for the case where \( u^* = t^* \). We next prove concavity for the case where \( u^* = t^* + 1 \).

First of all, if \( t^* = 0, v^* = 0 \). We easily have

\[ G(S + 1) - G(S) = [E(S + 1) - E(S)]p(0) + f \sum_{x=1}^{\infty} p(x), \]

\[ G(S) - G(S - 1) = [E(S) - E(S - 1)]. \]

All we need to show is \( f \leq E(S) - E(S - 1) \). In fact, \( u^* = 0 \) implies

\[ g(S, 1) - g(S, 0) = [f + E(S - 1) - E(S)] \sum_{x=1}^{\infty} p(x) < 0. \]

Thus

\[ f < E(S) - E(S - 1). \]
Appendix A. Proofs of Theorems and Propositions

If $t^* > 0$, then $v^* = t^* - 1$. This is because the optimality of $t^*$ implies

$$f \geq E(S - t^* + 1) - E(S - t^*),$$

$$f < E(S - t^*) - E(S - t^* - 1).$$

$v^* = t^*$ would imply $f < E(S - v^*) - E(S - v^* - 1)$, contradicting the optimality of $v^*$.

We are now ready to complete our proof. We know

$$G(S + 1) - G(S) = g(S + 1, u^*) - g(S, t^*) = g(S + 1, t^* + 1) - g(S, t^*)$$

$$= \sum_{x=0}^{t^*+1} [f \cdot x + E(S + 1 - x)] \cdot p(x) + \sum_{x=t^*+2}^{\infty} [f \cdot (t^* + 1) + E(S - t^*)] \cdot p(x) -$$

$$\sum_{x=0}^{t^*} [f \cdot x + E(S - x)] \cdot p(x) - \sum_{x=t^*+1}^{\infty} [f \cdot t^* + E(S - t^*)] \cdot p(x)$$

$$= \sum_{x=0}^{t^*} [E(S + 1 - x) - E(S - x)] \cdot p(x) + f \sum_{x=t^*+1}^{\infty} p(x),$$

$$G(S) - G(S - 1) = g(S, t^*) - g(S - 1, t^* - 1)$$

$$= \sum_{x=0}^{t^*} [f \cdot x + E(S - x)] \cdot p(x) + \sum_{x=t^*+1}^{\infty} [f \cdot t^* + E(S - t^*)] \cdot p(x) -$$

$$\sum_{x=0}^{t^*-1} [f \cdot x + E(S - 1 - x)] \cdot p(x) - \sum_{x=t^*}^{\infty} [f \cdot (t^* - 1) + E(S - t^*)] \cdot p(x)$$

$$= \sum_{x=0}^{t^*} [E(S - x) - E(S - 1 - x)] \cdot p(x) + f \sum_{x=t^*}^{\infty} p(x).$$

By concavity of $E(\cdot)$, we have $E(S + 1 - x) - E(S - x) \leq E(S - x) - E(S - 1 - x)$ for $x = 1, \ldots, t^* - 1$. It remains to show that $E(S + 1 - t^*) - E(S - t^*) \leq f$. This, however, is implied by the optimality of $t^*$ for $g(S, \cdot)$. We thus have

$$G(S + 1) - G(S) \leq G(S) - G(S - 1).$$
A.2 Proof of Theorem 2.15

To prove Theorem 2.15, we introduce the following three propositions.

Let $E(S)$ be a function defined on $(-\infty, +\infty)$ and has the following properties:

(A1) $E$ is continuous;

(A2) $E$ is nondecreasing;

(A3) $E$ is concave;

(A4) $\exists M > 0$ and $w > 0$, such that $\forall S < -M, E'(S) = w$; i.e., when $S$ is far less than zero, $E(S)$ is a straight line with the slope $w$.

(A5) $E(S)$ is upper-bounded.

For every nonnegative integer $t$, let

$$g(S, t) = \sum_{x=0}^{t} [frx + E(S - rx)]p(x) + \sum_{x=t+1}^{\infty} [frt + E(S - rt)]p(x)$$

where

$$0 < f < w, \ 0 < r \leq 1.$$

Define

$$G(S) = \max_{t \geq 0} g(S, t),$$

We have the following propositions.

**Proposition A.1** $G(S)$ is nondecreasing.

**Proof:** Since $E(S)$ is nondecreasing, for fixed $t \geq 0, g(S, t)$ is nondecreasing in $S$.

Thus $\forall S < S^*$,

$$g(S, t) \leq g(S^*, t).$$

Then

$$\max_{t \geq 0} g(S, t) \leq \max_{t \geq 0} g(S^*, t).$$
Appendix A. Proofs of Theorems and Propositions

That is

\[ G(S) \leq G(S') \]

Proposition A.2 For every \( S \), let

\[ t^*(S) = \min\{ t : t \geq 0, E(S - rt) - E(S - r(t + 1)) \geq fr \}, \]

then

\[ G(S) = g(S, t^*(S)). \]

Proof: It is easy to show, for every fixed \( S \),

\[ g(S, t) - g(S, t + 1) = \sum_{x=t+1}^{\infty} [E(S - rt) - E(S - r(t + 1)) - fr]g(x). \]

For every \( t \geq t^* = t^*(S) \), since \( E \) is concave,

\[ E(S - rt) - E(S - rt - r) \geq E(S - rt^*) - E(S - rt^* - r) \geq fr. \]

Thus

\[ g(S, t) - g(S, t + 1) \geq 0. \]

This implies that

\[ g(S, t^*) \geq g(S, t^* + 1) \geq g(S, t^* + 2) \geq \cdots \quad (A.1) \]

By definition of \( t^* \), for every \( t \leq t^* - 1 \), we have

\[ E(S - rt) - E(S - rt - r) < fr. \]

Then

\[ g(S, t) - g(S, t + 1) \leq 0. \]
This implies that

\[ g(S, t^*) \geq g(S, t^* - 1) \geq g(S, t^* - 2) \geq \ldots \]  \hspace{1cm} (A.2)

(A.1) and (A.2) imply that

\[ g(S, t^*) = \max_{t \geq 0} g(S, t) = G(S) \]

Proposition A.3 For any \( S \),

\[ t^*(S + r) - t^*(S) = 0 \text{ or } 1 \]

and

\[ t^*(S + r) - t^*(S) = 0 \]

if and only if both \( t^*(S + r) \) and \( t^*(S) \) are equal to 0.

Proof: By definition of \( t^*(S) \), we have

\[ E[S - rt^*(S)] - E[S - rt^*(S) - r] \geq fr. \]

The inequality can be rewritten as

\[ E[S + r - r(t^*(S) + 1)] - E[S + r - r(t^*(S) + 1) - r] \geq rf. \]

Then by definition of \( t^*(S + r) \),

\[ t^*(S + r) \leq t^*(S) + 1. \]
We need to show if
\[ t^*(S + r) \leq t^*(S), \]
then
\[ t^*(S) = t^*(S + r) = 0. \]

First we show \( t^*(S + r) = 0 \). Assume \( t^*(S + r) \geq 1 \), then from
\[ E[S + r - rt^*(S + r)] - E[S + r - r(t^*(S + r) + 1)] \geq rf. \]

We have
\[ E[S - r(t^*(S + r) - 1)] - E[S - r(t^*(S + r))] \geq rf. \]

Thus
\[ t^*(S) \leq t^*(S + r) - 1 \]
which is a contradiction.

Now \( t^*(S + r) = 0 \), then we have
\[ E(S + r) - E(S) \geq fr. \]

Since \( E \) is concave,
\[ E(S) - E(S - r) \geq E(S + r) - E(S) \geq fr. \]
Thus we get
\[ t^*(S) = 0. \]

**Theorem 2.15:**
\[ G(S + r) + G(S - r) \leq 2G(S). \]
Appendix A. Proofs of Theorems and Propositions

Proof: Denote

\[ v^* = t^*(S - r), \]
\[ t^* = t^*(S), \]
\[ u^* = t^*(S + r). \]

By Proposition A.3, we have 3 possible cases:

1. \( v^* < t^* < u^* \)
2. \( v^* = t^* = u^* = 0 \)
3. \( v^* = t^* = 0 \) and \( u^* = 1 \)

(1) For the case where \( v^* < t^* \), we must have \( t^* < u^* \) and \( t^* = v^* + 1, u^* = t^* + 1 \),

\[
G(S) - G(S - r) = g(S, t^*) - g(S - r, t^* - 1)
\]
\[
= \sum_{x=0}^{t^*} [frx + E(S - rx)]p(x) + \sum_{x=t^*+1}^{\infty} [frt^* + E(S - rt^*)]p(x)
\]
\[
- \sum_{x=0}^{t^*-1} [frx + E(S - r - rx)]p(x) + \sum_{x=t^*}^{\infty} [fr(t^* - 1) + E(S - rt^*)]p(x)
\]
\[
= \sum_{x=0}^{t^*} [E(S - rx) - E(S - r - rx)]p(x) + fr \sum_{x=t^*+1}^{\infty} p(x).
\]

Similarly,

\[
G(S + r) - G(r) = \sum_{x=0}^{t^*} [E(S + r - rx) - E(S - rx)]p(x) + fr \sum_{x=t^*+1}^{\infty} p(x).
\]

By definition of \( t^* \) and concavity of \( E(S) \)

\[
E(S - r(t^* - 1)) - E(S - rt^*) < fr
\]
and

\[ E(S + r - rx) - E(S - r) \leq E(S - r) - E(S - r - rx), \]

we have

\[
G(S + r) - G(r) \leq \sum_{x=0}^{t^*} [E(S + r - rx) - E(S - r) p(x)] + fr \sum_{x=t^*+1}^{\infty} p(x)
\]

\[
\leq G(S) - G(S - r).
\]

(2) For the case with \( v^* = t^* = u^* = 0 \), we have

\[
G(S) - G(S - r) = g(S, 0) - g(S - r, 0)
\]

\[
= E(S) - E(S - r)
\]

and

\[
G(S + r) - G(S) = E(S + r) - E(S).
\]

Thus

\[
G(S) - G(S - r) \geq G(S + r) - G(S).
\]

(3) For the case with \( 0 = v^* = t^* < u^* = 1 \), we have

\[
G(S) - G(S - r) = E(S) - E(S - r)
\]

and

\[
G(S + r) - G(S) = g(S + r, 1) - g(S, 0)
\]

\[
= [E(S + r) - E(S)]p(0) + fr \sum_{x=1}^{\infty} p(x).
\]

Note \( t^* = 0 \), we have

\[
E(S) - E(S - r) \geq fr.
\]
By concavity, \( E(S + r) - E(S) \leq E(S) - E(S - r) \). Then
\[
G(S + r) - G(S) \leq E(S) - E(S - r) = G(S) - G(S - r).
\]

A.3 Proof of Theorem 2.16

To prove Theorem 2.16, we introduce the following two propositions.

**Proposition A.4** \( \tilde{G}(S) \) satisfies (A1) – (A5), where \( \tilde{G}(S) \) is defined in Eq. (2.19).

**Proof:** (A1) By definition, \( \tilde{G}(S) \) is continuous on \( (-\infty, S_0) \) and on \( [S_0, +\infty) \). Additionally \( \tilde{G}(S_0) = G(S_0) \). We only need to show \( \tilde{G}(S) \) is continuous at \( S_0 \) from the left, i.e.,
\[
\lim_{S \to S_0^-} \tilde{G}(S) = G(S_0).
\]

We know that for \( S < S_0 \),
\[
\tilde{G}(S) = E(S).
\]

Then
\[
\lim_{S \to S_0^-} \tilde{G}(S) = \lim_{S \to S_0^-} E(S) = E(S_0).
\]

Since
\[
E(S_0) - E(S_0 - r) \geq E(S_0 + r) - E(S_0) = fr.
\]

We have
\[
t^*(S_0) = 0.
\]

Thus
\[
G(S_0) = E(S_0) = \lim_{S \to S_0^-} \tilde{G}(S)
\]
(A2) We have seen that $G(S)$ is nondecreasing, especially,

$$G(S_0) \leq G(S_0 + r) \leq G(S_0 + 2r) \leq \ldots$$

By definition of $\tilde{G}(S)$, for $S_0 + nr \leq S < (n + 1)r$, $n \geq 0$,

$$\tilde{G}(S) = G(S_0 + nr) + \frac{S - (S_0 + nr)}{r}[G(S_0 + (n + 1)r) - G(S_0 + nr)] \quad (A.3)$$

From this equation we can easily see that $\tilde{G}(S)$ is nondecreasing on $[S_0, \infty)$. On $(-\infty, S_0)$, $\tilde{G}(S) = E(S)$. But $E(S)$ is non-decreasing and at $S_0$, $\tilde{G}(S)$ is continuous. All these imply that $\tilde{G}(S)$ is non-decreasing.

(A3) By Eq. (A.3), we have, for every $S_0 + nr < S < (n + 1)r$, $n \geq 0$,

$$\tilde{G}'(S) = \frac{1}{r}[G(S_0 + (n + 1)r) - G(S_0 + nr)]$$

and at $S = S_0 + nr$, $n \geq 1$

$$\tilde{G}_-(S) = \frac{1}{r}[G(S_0 + nr) - G(S_0 + (n - 1)r)]$$

$$\tilde{G}_+(S) = \frac{1}{r}[G(S_0 + (n + 1)r) - G(S_0 + nr)]$$

since

$$G(S_0 + (n + 1)r) - G(S_0 + nr) \leq G(S_0 + nr) - G(S_0 + (n - 1)r)$$

and

$$\tilde{G}_+(S_0) \leq \tilde{G}_-(S_0)$$

Thus, $\tilde{G}'(S)$ is non-increasing. So, $\tilde{G}(S)$ is concave on $(S_0, +\infty)$. On $(-\infty, S_0)$, $\tilde{G}(S) = E(S)$, which is concave. Therefore we only need to check that $\tilde{G}(S)$ is concave at $S_0$.

First we get,

$$\tilde{G}_+(S_0) = \frac{1}{r}[G(S_0 + r) - G(S_0)]$$
since
\[ E(S_0) - E(S_0 - r) \geq E(S_0 + r) - E(S_0) = fr. \]

We have
\[ t^*(S_0 + r) = 0 \quad \text{and} \quad t^*(S_0) = 0, \]
thus
\[ G(S_0 + r) - G(S_0) = E(S_0 + r) - E(S_0). \]
so
\[ \tilde{G}'_+(S_0) = \frac{1}{r}[E(S_0 + r) - E(S_0)]. \]
By concavity of \( E(S) \),
\[ E'_-(S_0) \geq \frac{1}{r}[E(S_0 + r) - E(S_0)] \]
But \( E'_-(S_0) = \tilde{G}'_-(S_0) \). Therefore
\[ \tilde{G}'_-(S_0) = E'_-(S_0) \geq \frac{1}{r}[E(S_0 + r) - E(S_0)] = \tilde{G}'_+(S_0) \]
This implies \( \tilde{G} \) is concave at \( S_0 \).

(A4) This is clear, since \( \forall S < S_0, \tilde{G}(S) = E'(S) \).

(A5) \( \tilde{G}(S) \) is nondecreasing, and \( \forall S_0 + nr \leq S < (n + 1)r, n \geq 0, \)
\[ \tilde{G}(S) \leq \tilde{G}[S_0 + (n + 1)r] = G[S_0 + (n + 1)r]. \]
Since
\[ t^*[G(S_0 + (n + 1)r)] = n, \]
\[ G[S_0 + (n + 1)r] = \sum_{x=0}^{n}[frx + E(S_0 + (n + 1)r - rx)p(x)] + \sum_{x=n+1}^{\infty}[frn + E(S_0 + r)p(x)]. \]
$E(S)$ is upper-bounded, suppose $E(S) < L$, then

\[
G(S_0 + (n + 1)r) \leq L + \sum_{x=0}^{n} f_r x p(x) + \sum_{x=n+1}^{\infty} f_r n p(x) \\
\leq L + \sum_{x=0}^{\infty} f_r x p(x)
\]

for the expectation demands $\sum_{x=0}^{n} x p(x) < \infty$.

**Proposition A.5** Let $E, G, S_0$ as defined before. $\forall S$, if $nr + S_0 \leq (n + 1)r + S_0$, then $t^*(S) = \max(0, n)$.

**Proof:** If $n \leq 0$, then $S \leq S_0 + r$. We have

\[
E(S) - E(S - r) \geq E(S_0 + r) - E(S_0) = f_r,
\]

thus $t^*(S) = 0$.

If $n \geq 1$,

\[
nr + S_0 < S \leq (n + 1)r + S_0.
\]

Thus

\[
E(S - nr) - E(S - nr - r) \geq E(S_0 + r) - E(S_0) = f_r \quad (A.4)
\]

On the other hand, $S - nr > S_0$. Note

\[
S_0 = \sup \{ S : E(S + r) - E(S) = f_r \}
\]

thus for every $S > S_0$,

\[
E(S + r) - E(S) < f_r.
\]

Therefore

\[
E(S - (n - 1)r) - E(S - nr) < f_r \quad (A.5)
\]
Eqs. (A.4) and (A.5) imply that
\[ t^*(S) = n. \]

**Theorem 2.16:** Let \( E, G, S_0 \) be as defined in the above proposition, then
\[ 0 \leq G(S) - \tilde{G}(S) \leq fr. \]

**Proof:** \( \forall S, \) if \( S \leq S_0, \) then \( \tilde{G}(S) = E(S) \) and \( G(S) = E(S), \) so the inequality is obvious.
If \( S_0 + nr < S \leq S_0 + (n + 1)r, \) for some \( n \geq 0, \) by the last proposition, \( t^*(S) = n. \)
Since \( G(S) \) and \( \tilde{G}(S) \) are nondecreasing,
\[ G(S_0 + nr) \leq G(S), \]
\[ \tilde{G}(S) \leq G(S_0 + (n + 1)r). \]
We have
\[ |G(S) - \tilde{G}(S)| \leq G(S_0 + (n + 1)r) - G(S_0 + nr). \]
When \( i = 0, \)
\[ t^*(S_0 + r) = t^*(S_0) = 0, \]
then
\[ G(S_0 + r) - G(S_0) = E(S_0 + r) - E(S_0) = fr. \]
When \( n \geq 1, \) we have
\[ G(S_0 + (n + 1)r) - G(S_0 + nr) \leq G(S_0 + nr) - G(S_0 + (n - 1)r) \]
\[ \leq \cdots \]
\[ \leq G(S_0 + r) - G(S_0) \]
Thus

$$|G(S) - \tilde{G}(S)| \leq fr$$

Next we will show $G(S) \geq \tilde{G}(S)$. For $S_0 + nr < S \leq S_0 + (n+1)r$, $t^*(S) = n$, thus

$$G(S) = \sum_{x=0}^{n} [frx + E(S - rx)]p(x) + \sum_{x=n+1}^{\infty} [frn + E(S - nr)]p(x)$$

since $E(S)$ is concave, $G(S)$ is concave on $(S_0 + nr, S_0 + (n+1)r)$. Thus $G(S)$ on $(S_0 + nr, S_0 + (n+1)r]$ lies above the line connecting points $(S_0 + nr, G(S_0 + nr))$ and $(S_0 + (n+1)r, G(S_0 + (n-1)r))$, which is $\tilde{G}(S)$. So $G(S) \geq \tilde{G}(S)$.

\[\square\]

### A.4 Proof of Proposition 3.3

**Proposition 3.3** $R(n,t)$ is concave in $n$.

**Proof:** We need to show

$$R(n+1,t) - R(n,t) \leq R(n,t) - R(n-1,t).$$

We induce on $t$.

For $t = 0$, obvious.

Suppose $R(n+1,t-1) - R(n,t-1) \leq R(n,t-1) - R(n-1,t-1)$, $\forall n$.

Now

$$R(n+1,t) = \sum_{i=1}^{k} p_i \cdot r(n+1,i,t) + (1 - \sum_{i=1}^{k} p_i) \cdot R(n+1,t-1),$$

$$R(n,t) = \sum_{i=1}^{k} p_i \cdot r(n,i,t) + (1 - \sum_{i=1}^{k} p_i) \cdot R(n,t-1).$$
Appendix A. Proofs of Theorems and Propositions

So

\[ R(n+1, t) - R(n, t) = \sum_{i=1}^{k} p_i \cdot [r(n+1, i, t) - r(n, i, t)] + (1 - \sum_{i=1}^{k} p_i) \cdot [R(n+1, t-1) - R(n, t-1)] \]

\[ \leq \sum_{i=1}^{k} p_i \cdot [r(n + 1, i, t) - r(n, i, t)] + (1 - \sum_{i=1}^{k} p_i) \cdot [R(n, t - 1) - R(n - 1, i - 1)]. \]

It remains to be shown that

\[ r(n + 1, i, t) - r(n, i, t) \leq r(n, i, t) - r(n - 1, i, t). \]

\[ \begin{array}{c|c|}
LHS & \text{RHS} \\
\end{array} \]

By definition,

\[ r(n + 1, i, t) = \max\{f_i + R(n, t - 1), R(n + 1, t - 1)\}, \]

\[ r(n, i, t) = \max\{f_i + R(n - 1, t - 1), R(n, t - 1)\}, \]

\[ r(n - 1, i, t) = \max\{f_i + R(n - 2, t - 1), R(n - 1, t - 1)\}. \]

We need to discuss different cases.

1. \( f_i + R(n, t - 1) \leq R(n + 1, t - 1) \). We then have,

\[ f_i \leq R(n + 1, t - 1) - R(n, t - 1). \]

This implies

\[ f_i \leq R(n + 1, t - 1) - R(n, t - 1) \leq R(n, t - 1) - R(n - 1, t - 1) \leq R(n - 1, t - 1) - R(n - 2, t - 1) \]

Thus we have

\[ r(n + 1, i, t) - r(n, i, t) = R(n + 1, t - 1) - R(n, t - 1), \]

\[ r(n, i, t) - r(n - 1, i, t) = R(n, t - 1) - R(n - 1, t - 1). \]
Therefore
\[ r(n+1, i, t) - r(n, i, t) \leq r(n, i, t) - r(n-1, i, t). \]

2. \( R(n+1, t-1) < f_i + R(n, t-1) \). We then have

\[ \text{LHS} = f_i + R(n, t-1) - \max\{f_i + R(n-1, t-1), R(n, t-1)\}. \]

(a) If \( f_i + R(n-1, t-1) \geq R(n, t-1) \), then

\[ \text{LHS} = R(n, t-1) - R(n-1, t-1) \leq f_i. \]

i. If \( R(n-2, t-1) + f_i \geq R(n-1, t-1) \), then

\[ \text{RHS} = R(n-1, t-1) - R(n-2, t-1). \]

We have \( \text{LHS} \leq \text{RHS} \).

ii. Otherwise,

\[ \text{RHS} = f_i + R(n-1, t-1) - R(n-1, t-1) = f_i. \]

We have \( \text{LHS} \leq \text{RHS} \).

(b) If \( f_i + R(n-1, t-1) < R(n, t-1) \), then

\[ \text{LHS} = f_i + R(n, t-1) - R(n, t-1) = f_i. \]

Furthermore,

\[ R(n-1, t-1) - R(n-2, t-1) \geq R(n, t-1) - R(n-1, t-1) > f_i. \]
This implies

\[ R(n - 1, t - 1) > f_i + R(n - 2, t - 1). \]

And so

\[
\text{RHS} = R(n, t - 1) - \max\{f_i + R(n - 2, t - 1), R(n - 1, t - 1)\} \\
= R(n, t - 1) - R(n - 1, t - 1) > f_i = \text{LHS}
\]

That is, \( \text{LHS} < \text{RHS} \).

So we have just proved that

\[ r(n + 1, i, t) - r(n, i, t) \leq r(n, i, t) - r(n - 1, i, t). \]

Therefore,

\[
R(n+1, t) - R(n, t) \leq \sum_{i=1}^{k} p_i [r(n, i, t) - r(n-1, i, t)] + (1 - \sum_{i=1}^{k} p_i) [R(n, t-1) - R(n-1, t-1)] \\
= R(n, t) - R(n - 1, t).
\]

\[ \blacksquare \]

### A.5 Proof of Proposition 3.4

**Proposition 3.4:** (Diminishing return in \( t \)) Incremental revenue is non-decreasing in time, i.e.,

\[ R(n, t + 1) - R(n - 1, t + 1) \geq R(n, t) - R(n - 1, t). \]

**Proof:**

\[
R(n, t+1) - R(n-1, t+1) = \sum_{i=1}^{k} p_i [r(n, i, t) - r(n-1, i, t+1)] + (1 - \sum_{i=1}^{k} p_i) [R(n, t) - R(n-1, t)].
\]
It remains to be shown that

\[ r(n, i, t + 1) - r(n - 1, i, t + 1) \geq R(n, t) - R(n - 1, t). \]

\[ \text{LHS} \quad \text{RHS} \]

1. If \( f_i + R(n - 1, t) \leq R(n, t) \), by concavity, we have

\[ f_i + R(n - 2, t) \leq R(n - 1, t). \]

This implies

\[ \text{LHS} = R(n, t) - R(n - 1, t) = \text{RHS} \]

2. If \( f_i + R(n - 1, t) > R(n, t) \), then

\[ \text{LHS} = f_i + R(n - 1, t) - \max\{f_i + R(n - 2, t), R(n - 1, t)\}. \]

(a) If \( f_i + R(n - 2, t) \leq R(n - 1, t) \), then

\[ \text{LHS} = f_i > R(n, t) - R(n - 1, t) = \text{RHS}. \]

(b) If \( f_i + R(n - 2, t) > R(n - 1, t) \), then by concavity

\[ \text{LHS} = R(n - 1, t) - R(n - 2, t) \geq R(n, t) - R(n - 1, t) = \text{RHS}. \]

We therefore have proved that

\[ r(n, i, t + 1) - r(n - 1, i, t + 1) \geq R(n, t) - R(n - 1, t). \]

Therefore
\[ R(n, t + 1) - R(n - 1, t + 1) \]

\[ = \sum_{i=1}^{k} p_i \cdot [r(n, i, t) - r(n - 1, i, t + 1)] + (1 - \sum_{i=1}^{k} p_i) \cdot [R(n, t) - R(n - 1, t)] \]

\[ \geq \sum_{i=1}^{k} p_i \cdot [R(n, i, t) - R(n - 1, i, t + 1)] + (1 - \sum_{i=1}^{k} p_i) \cdot [R(n, t) - R(n - 1, t)] \]

\[ = R(n, t) - R(n - 1, t). \]
Bibliography


