AIRLINE REVENUE MANAGEMENT:
MODELS FOR CAPACITY CONTROL OF A SINGLE LEG AND A
NETWORK OF FLIGHTS

DISSERTATION

Presented in Partial Fulfillment of the Requirement for
The Degree of Doctor of Philosophy in the Graduate
School of the Ohio State University

By

Laila Haerian, M.S.

*****

The Ohio State University

2007

Dissertation Committee

Approved by

Dr. Clark Mount Campbell, Advisor

Dr. Keely Croxton

Dr. Mark McCord

-----------

Adviser

Industrial and Systems Engineering
Graduate program
ABSTRACT

Dynamic programming (DP) is one of the most powerful tools for finding the optimal booking policy for capacity control of a single leg flight. However, the extension of it to a network of flights is impractical due to the exponential growth of the size of the model with number of legs in the network. In this work we develop and use an approximate DP model to find the optimal protection levels on a single leg flight and extend it to a network of flights as well. We develop a Markov chain to calculate the expected revenue that is generated under implementation of a fixed policy at each stage of the approximate DP model and for any remaining capacity and then search for the optimal policy. We use large time chunks in the proposed DP model to decrease the computational effort and show that the resulting expected revenue converges to the expected revenue that is generated under implementation of the original DP approach. Unlike many of the existing models, in our proposed method, nesting is incorporated into the optimization procedure. Furthermore, by using the proposed Markov chain model, the expected generated revenue can be calculated directly and without using simulation.
Dedicated to My Dear Parents, My beloved sister Laleh,

My Dear Husband Amir and

My beautiful daughter Roxana
ACKNOWLEDGMENTS

First I would like to thank my advisor Dr. Clark Mount Campbell for all his guidance, help and support. I am so grateful for the privilege of having him as my advisor. I really enjoyed working under his supervision. I would also like to thank Dr. Tito Homem de Mello who was my co advisor when we started this research and before he left OSU. His vast knowledge and experience in this area was quite helpful. He is a great teacher and the course that I took with him, stochastic process, is truly the best and most beautiful course that I have ever taken in my entire academic life. I wish to express my gratitude to my committee members Dr. Keely Croxton and Dr. Mark McCord for reviewing this document and for their helpful suggestions. I would like to thank my colleagues Lijian chen and Weining Tsai for participating in our research meetings and for the helpful discussions that we had. I wish to thank my dear friends Laleh and Behrad Zand, Maryam and Masoud Larijani and Sanam and Yasaman Shakibi for their help and warm hospitality in my last two quarters of long distance research.

I wish to express my heartfelt gratitude to my dear parents for their endless love and support and for always believing in me and encouraging me toward higher education. I would also like to thank my beloved sister Laleh for
always being such an inspiring friend to me and for all the great things that I have learned from her. I wish to thank my dear husband Amir for all his love, kindness, help and support. Finally I wish to thank my beloved baby girl Roxana for cooperating with me and letting me finish this work, for uplifting my spirit with her beautiful smile in the difficult days of my research and for all the love, joy and blessing that she has brought into my life.
VITA

August 18, 1974 ....................... Born - Glasgow, Scotland
1998 ................................. B.S. Electrical Engineering,
Sharif University of Technology,
Tehran, Iran

2000 ................................. M.S. Applied and Industrial
Mathematics,
Towson State University,
Towson, MD

2000-present ........................ Graduate Teaching and Research
assistant, The Ohio State University

PUBLICATIONS


FIELD OF STUDY

Major: Industrial and Systems Engineering
TABLE OF CONTENTS

Abstract ........................................................................................................... ii
Dedication ..................................................................................................... iii
Acknowledgments ......................................................................................... v
Vita ............................................................................................................... vii
List of Tables................................................................................................ xii
List of Figures............................................................................................... xiii

Chapters:

1. Introduction............................................................................................. 1
   1.1. Revenue-based booking control method (bid price control method)........ 6
   1.2. Class-based booking control method.................................................. 7
       1.2.1. Partitioned and nested allocation of seats on a single leg .............. 8
       1.2.2. Partitioned and nested allocation on a network of flights ............. 10
   1.3. Existing nested allocation methods.................................................... 11
       1.3.1. Standard nesting control............................................................... 11
       1.3.2. Theft nesting control ................................................................. 12
   1.4. Problem definition ........................................................................... 13

2. Existing optimization models for finding the optimal booking policy ......... 17
2.1. Literature review .........................................................18
2.2. Some of the existing optimization models for finding
the optimal booking policy.................................................20
  2.2.1. Littlewood’s two class model.........................21
  2.2.2. Deterministic linear programming (DLP) model......23
  2.2.3. Dynamic programming (DP) model..................25
3. Introduction to proposed method.....................................29
  3.1. Big Picture .............................................................30
  3.2. General background and technical tools.....................31
    3.2.1. The nesting table ..............................................32
    3.2.2. Fill From the right (R) control method ............33
    3.2.3. Fill From the left (L) control method ............35
    3.2.4. L and R Markov chain Models .......................36
      3.2.4.1. L Markov chain model .........................37
      3.2.4.2. R Markov chain model .........................39
    3.2.5. Calculating the expected revenue .............41
  3.3. Summary .............................................................46
4. Proposed methodology ................................................48
  4.1. Proposed method for finding the optimal policy for
capacity control of a single leg .....................................51
    4.1.1. Proposed dynamic programming (PDP) model ....53
    4.1.2. Decreasing the size of the set of protection levels ....59
      4.1.2.1. Rule 1: a lower bound for number of
      seats that should be protected from
each fare class ...........................................60

4.1.2.2. Rule 2: an upper bound on number of
seats that should be protected from each
fare class .........................................................62

4.1.2.3. Multi dimensional line search.................66

4.1.2.4. Empirical study on effect of the
three rules on reducing the size of
the set of eligible protection levels.............68

4.1.3. Proposed method for finding the optimal
protection levels for capacity control of a
single leg flight under the L control method ........73

4.2. Numerical experiments ................................75

4.2.1. Using the nesting table and “L Markov chain’
to model the booking process under
implementation of bid price control ..........77

4.2.2. Design of experiment .............................80

4.2.3. Comparing the expected total revenue
versus rate of arrival under three different
methods ............................................................84

4.2.4. Comparing the expected total revenue
versus expected load factor under three
different control methods .............................91

4.2.4.1. Calculating the expected load factor
by using the L Markov chain model .....92

4.2.4.2. Graphs of expected revenue versus
expected load factor ...................................94

4.2.5. Expected total revenue versus number of stages ....100

4.2.6. Average over all 12 samples ..................102

4.3. Summary and Conclusion .........................103

5. Capacity control on a network of flights ..............105
5.1. Solving the network by decomposing it into individual single leg problems ..................................107

5.1.1. Displacement-adjusted revenue prorating (DARP) .........................................................107

5.1.2. Proposed prorating method: remaining capacity pricing (RCP) .........................................108

5.1.3. Proposed method for capacity control of a network of flights ..........................................111

5.2. Numerical experiments on network .........................112

5.2.1. Network methods .................................................114

5.2.1.1. RCP-PDP network control method .............114

5.2.1.2. DARP-PDP network control method ..........115

5.2.1.3. NET-BP control method .........................117

5.2.2. Numerical experiments ......................................118

5.2.2.1. Experiment 1 ..............................................119

5.2.2.2. Experiment 2 ..............................................126

5.2.2.3. Experiment 3 ..............................................133

5.2.2.4. Computational effort .................................136

5.3. Summary ...............................................................140

6. Conclusion and future study ....................................142

Bibliography .................................................................147

Appendixes .................................................................149

A: Supporting information for chapter 3 ..................149

B: Supporting information for chapter 4 .................154
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Summary of notation ..........................</td>
</tr>
</tbody>
</table>
| 4.2   | Solution of stage “s-1” of the PDP model  
       In a tabular format .......................... | 60 |
| 4.3   | Optimal PL’s on stage s of the PDP model ........ | 63 |
| 4.4   | Optimal protection levels on stage 1 and 2 in  
       a tabular format .......................... | 74 |
| 4.5   | Optimal protection levels on stage S-1 and S  
       in a tabular format ........................ | 75 |
<p>| 4.6   | 12 samples for the numerical result .......... | 84 |
| 5.1   | Table of notation for DARP prorating method ... | 108 |
| 5.2   | Table of notation for RCP prorating method .... | 109 |
| 5.3   | Capacity of the legs of network 1 ............ | 120 |
| 5.4   | ODF’s of network 1 ............................ | 121 |
| 5.5   | ODF’s of network 2 ............................ | 128 |
| 5.6   | Capacity of the legs of network 2 ............. | 128 |
| 5.7   | ODF’s of network 2 for experiment 3 ............ | 135 |
| 5.8   | Capacities of the legs of network 2 for experiment 3 | 135 |</p>
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>A sample Nesting Table, for an airplane with $C=5$, $n=3$ and $PL=(0,2,4)$</td>
</tr>
<tr>
<td>3.2</td>
<td>Applying R method to control the booking process for example 3.1</td>
</tr>
<tr>
<td>3.3</td>
<td>Applying L method to control the booking process for example 3.1</td>
</tr>
<tr>
<td>3.4</td>
<td>The L Markov chain model for $C=4$, $n=3$ and $PL=(0,1,2)$</td>
</tr>
<tr>
<td>3.5</td>
<td>The R Markov chain for an airplane with $C=4$, $n=3$ and $PL=(0,1,2)$</td>
</tr>
<tr>
<td>3.6</td>
<td>Expected revenue gained from each of the columns of the nesting table for $C=4$, $n=3$ and $PL=(0,2,3)$</td>
</tr>
<tr>
<td>3.7</td>
<td>The L Markov chain model with shaded filled columns for $C=4$, $n=3$ and $PL=(0,2,3)$</td>
</tr>
<tr>
<td>4.1</td>
<td>Size of the set of eligible protection levels versus capacity for $n=3$</td>
</tr>
<tr>
<td>4.2</td>
<td>Graph of $N$ versus $n$ for $C=20$</td>
</tr>
<tr>
<td>4.3</td>
<td>The size of the set of eligible protection levels as a fraction of the original set when applying rule 1 for different capacities</td>
</tr>
<tr>
<td>4.4</td>
<td>The size of the set of eligible protection levels as a fraction of the original set when applying rules 1 and 2 for different capacities</td>
</tr>
<tr>
<td>4.5</td>
<td>The size of the set of eligible protection levels as a fraction of the original set when applying all three rules</td>
</tr>
<tr>
<td>4.6</td>
<td>A nesting table with a specific format of protection levels that could be used to model booking process</td>
</tr>
</tbody>
</table>
under bid price control with $r_i \geq \text{Bid Price} > r_{i+1}$ .................78

4.7 Patterns for $p(i)$ with respect to $i$ .............................................83

4.8 Patterns for $r(i)$ with respect to $i$ .............................................83

4.9 Expected revenue under implementation of DLP, PDP and DP methods where the optimal policy is updated 3 times .........................................................87

4.10 Expected revenue under implementation of DLP, PDP and DP methods for sample 1 of table 4.6 where the optimal policy is updated 12 times ........................................88

4.11 Expected revenue under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 6 times .................................88

4.12 Percentage difference in expected revenue with the DP method under implementation of DLP and PDP methods for sample 3 of table 4.6 where the optimal policy is updated 12 times ..................................................89

4.13 Expected revenue under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 60 times ........................................89

4.14 Percentage difference in expected revenue with the DP method under implementation of DLP and PDP methods for sample 3 of table 4.6 where the optimal policy is updated 60 times ..................................................90

4.15 Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated once ..........95

4.16 Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 3 times ..........95

4.17 Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 6 times ..........96
4.18 Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 12 times ………96

4.19 Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 30 times……….97

4.20 Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 60 times……….97

4.21 Average percentage difference in the expected revenue with DP method where the average is taken over all the 12 samples and different arrival rates………………………………………101

4.22 Average revenue under implementation of each of the three methods versus the arrival rate where, the average is taken over all 12 samples and different stage numbers………………102

5.1 Network 1 ………………………………………………………120

5.2 Average revenue of network 1 during the booking period for arrival rate of 0.01 under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP………..122

5.3 Average revenue of network 1 during the booking period for arrival rate of 0.015 under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP………..123

5.4 Average revenue of network 1 during the booking period for arrival rate of 0.02 under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP………..124

5.5 Average computational effort that is required to calculate the optimal booking policy under implementation of three control methods: NET-BP, RCP-PDP and DARP-BP………..125

5.6 Average revenue of network 1 during the booking period under implementation of three control methods: NET-BP, RCP-PDP and DARP-BP……………………………………126

5.7 Network 2 ……………………………………………………..127

5.8 Average revenue of network 2 during the booking period
for arrival rate of 0.01 under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP ...............129

5.9 Average revenue of network 2 during the booking period for arrival rate of 0.012 under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP........130

5.10 Average computational effort that is required to calculate the optimal booking policy under implementation of three control methods: NET-BP, RCP-PDP and DARP-BP on network 2 .................................................................131

5.11 Average revenue of network 2 during the booking period under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP.........................................................132

5.12 Average revenue per seat for network 2 during the booking period under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP..............................133

5.13 Network 2 with more ODF’s for experiment ..................134

5.14 Average revenue of network 3 during the booking period under implementation of three control methods: NET-BP, RCP-PDP and DARP .................................................136

5.15 Network 3, a 2-leg network.........................................137

5.16 Network 4, a 4-leg network.........................................137

5.17 Network 5, a 6-leg network.........................................137

5.18 Network 6, an 8-leg network......................................138

5.19 Network 7, a 10-leg network......................................138

5.20 Computational effort versus number of legs in the network.....139

5.21 Computational effort versus number of fare classes on the legs of network 6.............................................139
INTRODUCTION

After deregulation of the airline industry in the 1970s, airlines were able to set their own prices. They could also sell seats of the same cabin for different prices. That was an early stage for revenue management (RM). This area has evolved considerably since then. It has also expanded into businesses other than the airline industry like the hotel and cruise industries, though the main application is still in the airline industry (Park and Piersma, 2002). Generally speaking, revenue management is to control and manage an inventory of perishable product over a limited time such that the total revenue is maximized. The key point is that in a revenue management problem, customers are willing to pay different prices for the same physical product. The difference in prices is due to different conditions that come along with the product. Important elements of a RM problem are: (i) limited capacity, (ii) limited time (perishable product), (iii) same product can be sold for different prices and (iv) demand for each product is stochastic. Usually, demand for buying the product at each price is assumed to be estimable.
In the airline industry, the perishable product is the ticket for a seat in a cabin of an airplane. The product is perishable simply because an airline ticket is only valuable until the plane takes off. Furthermore, a seat in the same cabin could be sold for different prices due to different conditions such as requirement for Saturday night stay or the ticket being refundable versus non-refundable or changeable versus non-changeable, or it could be a part of different itineraries on multi-leg trips. Broadly speaking, RM deals with various issues including forecasting and demand modeling, pricing, overbooking and finally capacity control (Park and Piersma, 2002). Forecasting is an area in which the demand for each itinerary is forecasted. Pricing is an area in airline revenue management in which different itineraries are defined and the prices of them are set. There is always the possibility that some passengers do not show up at the flight time, so in order to cope with no-shows, airlines usually overbook the flight. The level at which the flight should be overbooked is a broad area of research. The detail of how these functions; pricing, forecasting and overbooking; are done is out of the scope of this research. The focus of this study is on capacity control. Capacity control is also known as seat inventory control. In fact, one can think of all the available seats as an inventory of perishable products and of capacity control as seat inventory management. After itineraries are defined, their fares are set and forecasted and the level of overbooking is determined, the next step is to find a way to control the booking process. Controlling the booking process means making an accept/reject decision for each and every request that arrives during the booking period when the booking period is the duration of time in which the
reservations are made and airline tickets are sold. The requests could be for any of the existing itineraries. The total revenue that is gained during the entire booking period depends on all of the accept/reject decisions that have been made. The goal in booking control is to maximize the total expected revenue. As mentioned in the previous paragraph, the focus of our study is on capacity control of a single leg flight as well as a network of flights.

Booking control consists of a control method such as bid price control or standard nesting control method and a set of control parameters such as a set of bid price values (any seat request with a class fare lower than the bid price is rejected, otherwise it is accepted) or a set of protection levels (indicates the number of seats protected from being booked by other fare classes). The control parameters are associated with the control method in use. Both the control method and set of control parameters must be designed to maximize the total expected revenue. Together they are usually called the “booking policy”. So by booking policy we mean a control method and a set of control parameters that is used to control the booking process and by booking control we mean the actual implementation of the booking policy by the airlines. Booking policy could be interpreted as a guideline for making the accept/reject decisions. There exist various methods and optimization models to calculate or estimate the optimal booking policy, see for examples Talluri and Van Ryzin (2004).

Airlines usually offer tickets for many origin destination flights. That makes a network of interrelated itineraries where many of them share the same leg. Typically, the problem of seat inventory control of the network of flights is
solved in one of two ways: as independent single leg flights, or through heuristic methods for a network of simultaneous flights. Solving single leg problems allows for (i) the use of powerful techniques like dynamic programming, and (ii) the incorporation of many features like overbooking, cancellation, consumer choice, etc. However, the interaction between flights is lost. For example if the problem is solved as independent legs, one connected flight using two legs might be rejected in favor of accepting a single leg flight with higher revenue per leg, even though it might have been more profitable to the network to accept the connected flight. To overcome this difficulty, one practice is to decompose the price of the multi-leg flight into individual single legs in such a way that it could represent the contribution of that part of the itinerary to the network and then solve the network as independent legs. The process of decomposing the price of the multi leg itinerary is called prorating or indexing. (Talluri and Van Ryzin, 2004) In a network of flights, itineraries are differentiated by their origin, destination and fare. The abbreviation ODF which stands for origin-destination-fare has been used in the literature to represent a typical itinerary from an origin to a destination with a specific fare. In a single leg capacity control problem though, since all the itineraries go from the same origin to the same destination there is no need to represent them with ODF. In fact in a single leg problem, the only factor that differentiates the products from each other is the fare. It is the norm to categorize the products on a single leg flight into various “fare classes”. Each fare class has a fixed fare and an estimated demand. Throughout this work, whenever we are dealing with a single leg capacity control problem, we use the
term fare class and whenever we are dealing with network capacity control we use the term ODF.

The heart of airline RM is capacity control or seat inventory control, which means to control the booking process so that the revenue that is generated during the booking period is maximized. Assuming that ODFs of a network of flights or fare classes on a single leg flight are defined, the price for each ODF or fare class is set, the demand for each one is forecasted and the level of overbooking has also been determined, the goal is to find the optimal booking policy. There exist different methods and optimization models to find or estimate the optimal booking policy. No matter what optimization model is used, the solution is always translated into a booking policy. In fact, the optimization models solve for the optimal booking policy that maximizes the total revenue. The booking policy is then implemented by the airline to control the booking process.

If the demand were deterministic, finding the optimal booking policy would have been an easy job. The stochastic characteristic of demand is what makes the problem challenging. As mentioned before, a booking policy consists of a booking control method and a set of control parameters.

Generally speaking, there are two types of booking control methods: class-based and revenue-based. These two have been suggested for both single-leg and network capacity control. They are explained in sections 1.1 and 1.2.
1.1 REVENUE-BASED BOOKING CONTROL METHOD (BID PRICE CONTROL)

We first explain the revenue-based booking control method for a single leg problem and then discuss the extension of this method to a network of flights. In a revenue-based booking control, the booking process is controlled by using a control parameter called bid-price. This type of booking control is often referred to as bid-price control. Bid-price control on a single leg flight works as follows. Once a request for a fare class is received, the fare of that fare class is compared with the value of a pre-calculated parameter called bid-price. The request is accepted if its fare exceeds or is equal to the value of the bid-price and rejected otherwise. In other words, seats are sold to requests whose fares are greater than or equal to the bid-price on a first come-first-serve order. Different models have been proposed in the literature to find the optimal value of the bid-price. The detailed descriptions of these models are presented in chapter 2. The basic logic in all of them is that a request for a fare class should be accepted only if the immediate revenue that is gained by accepting it exceeds the expected revenue that is gained by rejecting the request and reserving the seat for future requests. Because of the stochastic characteristic of demand, the future revenue is not known with any certainty. The estimated expected future revenue is in fact the “opportunity cost” of the current seat. This value is a function of remaining capacity of the cabin and remaining time until take off. A common practice is to update the optimal value of the bid-price as the remaining capacity and remaining time change throughout the booking process.
In a network of flights, there is a bid price value associated with each of the ODF’s. Since there are some ODF’s on a network of flights that use more than one leg, bid-price control on a network of flights is slightly different than the one on a single leg flight. When a request for a multi-leg itinerary is accepted, multiple seats from the legs that are used by that itinerary are being sold. Therefore in order to estimate the future value of rejecting a multi-leg itinerary, one has to take into account all of the seats that are involved. There is an opportunity cost associated with each of those seats. A common practice is to estimate the future value of a multi-leg ODF by adding the opportunity cost of the seats that are used by the ODF.

1.2 CLASS-BASED BOOKING CONTROL METHODS

In a class-based booking control of a single leg flight, a specific number of seats are allocated to each fare class or to a group of fare classes. The total numbers of demands that are accepted for each fare class or group of fare classes are then controlled according to these allocations. Similarly, in a class-based booking control of a network of flights, a specific number of seats are allocated to an ODF or a group of ODFs and total numbers of demands that are accepted to each ODF or a group of ODFs are then controlled according to these allocations. Class-based booking control method may be partitioned or nested. We first explain the partitioned and nested class-based control for a single leg flight in section 1.2.1. The extension of this type of control to a network of flights will then be discussed in section 1.2.2.
1.2.1 PARTITIONED AND NESTED ALLOCATION OF SEATS ON A SINGLE LEG

Let \( C \) represent the capacity of the cabin of a single leg flight and \( n \) the number of fare classes. Let us define the booking period as the duration of time from when the airlines start to sell the tickets until the flight time. Seat allocation on a single leg is to allocate \( C \) seats to \( n \) different fare classes so that the revenue that is gained during the entire booking period is maximized. Partitioned allocation is when the capacity is partitioned between the fare classes. Under partitioned allocation, the number of seats that are allocated to one fare class cannot be used by other fare classes. Nested allocation on the other hand is when a fare class is allowed to use up the seats that were originally allocated to a lower value fare class.

It has been shown in the literature (Williamson, 1992) that nested allocation has advantage over partitioned allocation. To clarify the advantage of nested allocation over partitioned allocation consider the following example, let \( N_1 \) and \( N_2 \) represent the number of seats that are allocated to fare classes 1 and 2 respectively when fare class 1 has a higher value. Now suppose that the number of requests for fare class 1 turned out to be more than \( N_1 \) and the number of requests for fare class 2 less than \( N_2 \). Since the number of requests for fare class 1 exceeds the number of seats that were originally allocated to it, some of the requests for fare class 1 must be rejected. On the other hand, since there were less requests for fare class 2 than \( N_2 \), some of the seats that were allocated to fare
class 2 will be unsold. These empty seats could have been sold to fare class 1 if it was not for the partitioned allocation. Thus, one possible scenario under partitioned allocation is to reject some of the higher value fare classes and yet have some unsold seats at the flight time. Nested allocation could prevent this problem by making all the seats that are allocated to the lower value fare class available to the higher value fare class.

The optimization models that are used for finding the optimal allocation usually do not incorporate the nesting as a characteristic of the optimal solution. If they do, the optimization model becomes very complicated. Instead, a common recommendation is to find the optimal partitioned allocation and then allow for a nested control policy. For example suppose there are three fare classes where $X_1$, $X_2$ and $X_3$ represent the optimal partitioned allocation. To apply the nesting is to let requests for fare class one (highest revenue) use not only the $X_1$ seats that are allocated to it but also $X_2$ and $X_3$ seats that were allocated to the lower value fare classes. Similarly, requests for fare class 2 can use not only the $X_2$ seats that were originally allocated to it but also $X_3$ seats that were allocated to fare class 3. Under the nesting, a fare class can use up a seat that was originally allocated to another fare class with a lower value. One of our contributions in this area is to incorporate nesting into the optimization procedure. We will discuss it in more details in the next chapters.
1.2.2 PARTITIONED AND NESTED ALLOCATION ON A NETWORK OF FLIGHTS

Partitioned allocation on a network of flights is simply the extension of the single leg version. In fact, the capacity of each leg is partitioned between ODFs which use that leg. Remember that ODF stands for origin-destination-fare and represents a unique itinerary from an origin to a destination with a specific fare. Seats that are allocated to each ODF on a leg can not be used by other ODFs on that leg. Nested allocation of the kind that we saw for single leg flight is not easy to translate into a network. The main reason is that ordering ODF’s on each leg is not as straightforward as they are on a single leg flight. In fact, the price of an ODF is not the only factor that determines its rank on a leg, the contribution of that leg of the ODF to the network is also a factor that has to be taken into account when ranking the ODFs on each leg. This is done through a process called indexing or prorating, in which the fare of the multi-leg ODF is broken into fares on individual legs such that the decomposed fares represent the contribution of that part of the itinerary to the whole network. In fact, each ODF is indexed into individual legs that are used by it and becomes a so called “virtual fare class” on that leg (Talluri and Van Ryzin, 2004) Then all of the fare classes including the virtual ones are sorted on each leg based on their fare. The nested control is then implemented in the same way as for the single leg capacity control. Different methods for this indexing process have been proposed in the literature. They will be briefly discussed later in chapter 2.
1.3 EXISTING NESTED ALLOCATION METHODS

Under the same nested allocation, the booking process can be controlled in two different ways, “standard nesting” or “theft nesting”. These methods are applicable to both single leg as well as a network of flights. In order to apply these control methods to a network of flights, the multi-leg itineraries should first be decomposed into individual legs through an indexing process and then the network will be controlled as independent legs. The single leg might be either the sole leg involved in the capacity control problem or could be part of a network of flights. Standard and theft nesting control methods are explained in sections 1.3.1 and 1.3.2 respectively.

1.3.1 STANDARD NESTING CONTROL

Let $n$ represent the total number of fare classes on a single leg flight. When we say fare classes on a single leg flight we are actually including the virtual fare classes that are indexed from a multi-leg itinerary to the individual legs in a case of a network problem. Therefore, this method could be used for capacity control on a single leg flight or on a network of flights. In the standard nesting control method, the first $n$ fare classes are sorted and ranked based on their fare. Then $n$ buckets are defined. A bucket is a set of fare classes.

Let $B_i$ denote bucket $i$. By definition, $B_i$ includes fare class $i$ and all higher-value fare classes; $B_i = \{1,2,\ldots,i\}$. Booking limit $BL_i$ is defined as the total number of requests that could be accepted for the fare classes of $B_i$. The optimal values of $BL_i$ must be calculated using some optimization model. We do not discuss
these optimization models here. Instead we will focus on how the $BL_i$’s could be implemented and used by the airlines to control the booking process.

In order to implement this nesting control method in practice, airlines should keep track of the remaining capacity of the cabin as well as the total number of demands that have been accepted for each bucket. Let $d_i$ represent the total number of requests that have been accepted so far for all the fare classes of bucket $i$ together. Note that since fare class 1 is a member of all the buckets, when a request for fare class 1 is accepted, $d_i$ of all the buckets must be updated. Similarly, when a request for fare class $k$ is accepted, $d_i$ of buckets $B_1, B_2, \ldots, B_k$ must be updated. Once a request for fare class $k$ is received, it would be accepted only if,

(i) There are enough seats available

(ii) $d_i < BL_i$ for $i = 1, 2, \ldots, k$

The second condition means that none of the buckets that $k$ belongs to have yet reached their limit. Once a request for fare class $k$ is accepted, $d_i$’s are incremented by one for all $i \leq k$, and the process is repeated for each new reservation request.

1.3.2 THEFT NESTING CONTROL

Let us first define “protection level”. The concept of a protection level is an alternative way of implementing a nesting control policy. The protection level of fare class $i$, $PL_i$, is the number of seats that are protected from fare class $i$ and
all lower valued fare classes and reserved for higher value fare classes. In the
theft-nesting-control method, first the fare classes (including the virtual fare
classes for a network problem) are sorted and ranked based on their fare. Let us
assume that some optimization model is used to calculate the optimal values
of $PL_i$s. We do not discuss these optimization methods in this section, instead we
will focus on how the $PL_i$s are intended to be used to control the booking process.

In order to control the booking process by the theft-nesting-control method, the
airline would keep track of the remaining capacity of the cabin. Let $x$ denote the
remaining capacity of the cabin at each time. Once a request for fare class $k$ is
received, it would be accepted only if,

(i) There is enough capacity in the cabin.

(ii) $x > PL_k$

The second condition means that there is at least one seat that is allocated to fare
class $k$ and lower value fare classes. After accepting a request for fare class $k$, the
remaining capacity must be updated and the process is repeated for the next
request.

1.4 PROBLEM DEFINITION

The focus of this study is capacity control on a single leg flight and on a
network of flights. Let $C$ represent the capacity of a single leg flight and $n$ the
number of fare classes. Furthermore, assume that fares of all the $n$ fare classes are
given and fixed and the demand for each one is forecasted. Let the booking
period be the duration of time since the airline starts to sell the tickets until the
flight time. Capacity control on a single leg flight is to control the booking process by selling each of the $C$ available seats such that the total expected revenue that is gained throughout the entire booking period is maximized. Let ODF which stands for origin-destination-fare represent a specific itinerary from origin O to the destination D with a specific fare F. Capacity control on a network of flights is to control the booking process by selling the seats that are used by each ODF such that the total expected revenue that is gained throughout the entire booking period on the network of flights is maximized. The accept/reject decisions are made by use of a booking policy. The optimal booking policy is the one that guarantees making the “right” accept/reject decision at all times. The question is how to find the optimal booking policy.

In this work we present some of the existing booking control methods and the optimization models that are used to calculate or estimate the booking policy for capacity control of both single leg and a network of flights. We will then propose alternative methods and show its advantage over those that are most commonly practiced by presenting the results from numerical experiments.

The main assumptions that we have used in our approach are as follows.

1- We assume that there is no cancellation of a reservation during the booking period.

2- We assume that all the customers who have reserved a seat will show up at the flight time. In other words, there is no such thing as no-shows and therefore there is no need for overbooking a flight.
3- We assume that requests for itineraries/fare classes arrive one at a time and independent of each other. In other words, a demand for an itinerary will request one seat on each of the legs that are used by that itinerary. On a single leg case, a demand for a fare class will request only one seat of the cabin. Batch arrivals are not taken into consideration.

4- We assume that the arrival of the requests is a Homogeneous Poisson process and the probability of arrival of each fare class (on a single leg) or ODF (on a network of flights) in one time unit is estimable. In order to keep up with the Poisson arrival assumption, one time unit has to be small enough so that at most one request can arrive in a time unit.

We should mention here that our approach is capable of relaxing assumptions 1, 2 and 3. Some revisions are needed to cope with the relaxation, which clearly make the proposed methods and models more complicated. More detail on which is out of scope of this study and could be a potential continuation of this work. The homogeneity assumption in 4 can be relaxed with no change in the overall approach.

In the rest of this work, we will first look at some of the existing models and methods in the literature. We will explain some of them in more detail in chapter 2. In chapter 3, we will present some models and technical tools that we have used in our proposed method. We will then present our proposed method for single leg capacity control in chapter 4. We also present some numerical result in
chapter 4 to compare the performance of our proposed method with some of the existing ones. In chapter 5, we discuss how to extend our proposed single elg capacity control method to a network of flights. Some numerical result are also presented in chapter 5 to compare the performance of our proposed method with the existing ones. Finally, in chapter 6 we present a summary and future study.
 CHAPTER 2
EXISTING OPTIMIZATION MODELS FOR FINDING THE OPTIMAL
BOOKING POLICY

In this chapter we look at some of the existing optimization models that have been proposed in the literature for finding the optimal booking policy for capacity control of a single leg or a network of flights. Airline revenue management in the real world deals with a huge network of interrelated flights. Therefore the real problem is capacity control on a network of flights. As discussed in chapter 1, there are mainly two approaches for capacity control of a network of flights. One is to solve the network as a whole and the second approach is to first decompose the prices of the multi leg itineraries into individual legs and then control the network by controlling the process on individual single leg flights. The decomposed price on each leg should be a good estimate of the contribution of that part of the itinerary to the whole network. Because of this second approach, the models and methods for capacity control on a single leg flight are also very important to look at.

The remainder of this chapter is organized as follows. A literature review on airline revenue management is presented in section 2.1. We then present some
of the existing optimization models in section 2.2. We will discuss the first
optimization model that was proposed for single leg capacity control problem
in section 2.2.1. A deterministic linear programming model which is the most
common optimization model that is being used in practice is presented in section
2.2.2. This model could be used for capacity control on a single leg as well as a
network. Finally a dynamic programming model for finding the optimal booking
policy on a single leg flight is presented in section 2.2.3.

2.1 LITERATURE REVIEW

Littlewood (1972) was the first to propose a model for the seat capacity
control problem. That model was developed for a single leg with only two fare
classes. The assumption in his model was that requests arrive in order from low
fare to high fare in two non-overlapping segments during the booking period. His
idea was to compare the marginal revenues in each of the two fare classes and
close the low value fare class once its fare is exceeded by the expected revenue of
selling the same seat to the higher value fare class. Belobaba (1987) extended
Littlewood’s idea to more than two fare classes and proposed a heuristic to find
the nested protection levels. His method is called expected marginal seat revenue
(EMSR) and does not yield optimal booking limits when more than two fare
classes are considered. Later, Curry (1990), Wollmer (1992) and Brumelle and
McGill (1993) proposed different algorithms to find the optimal nested allocation
for multiple fare classes where fare classes book sequentially.
Glover et al. (1982) were the first to propose a network formulation for the airline revenue-management problem. They formulated the problem as a network flow problem. The drawback of their model is the necessity of passengers being path indifferent, which is not a realistic assumption. The integer-programming model underlying the network flow problem is able to distinguish between different paths. The linear relaxation of Glover’s et al. model which is a deterministic linear programming (DLP) model has been proposed as an alternative. However, it replaces the stochastic demand with only one estimate; the expected value; and does not fully capture the stochastic nature of demand. Wollmer (1992) proposed a model, which takes into account the distribution of the demand. A drawback of this model is the large number of decision variables. De Boer et al. (2002) proposed a model, which is similar to Wollmer’s with fewer decision variables but still does not incorporate nesting into the optimization procedure.

A very important aspect of seat inventory control is nesting. As discussed in the introduction, to use nested control in a network of flights, one needs to map the multi-leg itineraries to individual legs through a process called indexing or prorating. Furthermore, the fare classes on each leg have to be ranked. There have been different ideas about how to rank the fare classes in a network of flights. Williamson (1992) suggested ranking fare classes in a network of flights based on the incremental revenue that is generated if an additional seat is made available to each fare class. For the DLP model, she approximated this by using the dual variables of the demand constraint. The idea of nesting has become quite
popular among practitioners. In many cases, a standard method such as linear programming is used to determine good partitioned seat allocations and then a nesting policy based on these allocations is implemented. Some researchers have proposed the incorporation of the nesting procedure into the optimization algorithm, although in that case the optimization techniques become much more involved. Notable examples of the latter are Bertsimas and de Boer (2005) and van Ryzin and Vulcano (2003). In our proposed method for capacity control, we look at the problem from a different perspective and our model is able to incorporate nesting into the optimization procedure.

2.2 SOME OF THE EXISTING OPTIMIZATION MODELS FOR FINDING THE OPTIMAL BOOKING POLICY

Below, we discuss in details some of the optimization models that are used more frequently in practice or the ones that are vital. The purpose of all of the presented models is to calculate or approximate the optimal value of control parameters such as the bid-price value for bid-price control or optimal allocation for class-based control such as optimal set of protection levels for standard nesting control. Note that depending upon the assumptions that are made on the arrival pattern of the requests, a model could be “static” or “dynamic”. These two terms, static and dynamic, are used in the literature and we adopt them here. The early stage models that were developed for a single leg capacity control were all static.
The main assumptions in a static model are as follows. (Talluri and Van Ryzin, 2004):

1- Requests for fare classes arrive in non-overlapping time intervals in order of their value, with the lowest value fare class arriving first and the highest value fare class arriving last.

2- The total demand for all fare classes are independent random variables.

3- The demand for each fare class is independent of availability of other fare classes.

In a static model usually, the total demand for each fare class is assumed to follow a normal distribution with known mean and variance.

In a dynamic model, the assumption on the order of requests is relaxed. In other words, requests for fare classes can come in any order during the booking period. In order to be able to track the arrivals of the requests, the arrival pattern is usually assumed to be Poisson. In these models, the entire booking period is divided into time units where each time unit is small enough that the probability of receiving more than one arrival in one time unit is negligible. The requests can come in any order and not necessary low value to high value. Assumptions 2 and 3 of the static model still hold here.

2.2.1 LITTLEWOOD’S TWO CLASS MODEL

Littlewood’s model was the first model that addressed the seat allocation problem for a single leg flight. Based on the assumptions that are made on the arrival pattern, this model is categorized as a static model. Let $D_i$ and
Let $D_2$ represent total demand for fare classes 1 and 2 respectively. Also let $r_1$ and $r_2$ denote revenues of these two fare classes. Suppose fare class 1 is the higher value fare class or $r_1 > r_2$. In a static model with two fare classes, the entire booking period is divided into two stages where the booking process starts at the beginning of stage 2 and the airplane takes off at the end of stage 1. The requests for fare class 2 (the lower value fare class) arrive during stage 2 and the requests for fare class 1 (the higher value fare class) arrive in stage 1. At some point during stage 2 it might be more profitable to stop selling seats to the lower value fare class (fare class 2) and reserve the remaining seats for the higher value requests that will come later in stage 1. We are interested in calculating the optimal number of seats that should be reserved for fare class 1.

Suppose that there are $x$ seats remaining at some point during stage 2. The question is whether to sell another seat to fare class 2 or stop selling seats to fare class 2 and reserve the remaining seats for fare class 1. Littlewood’s idea was to compare the immediate revenue that is gained if we sell the current seat to fare class 2 with the expected revenue of reserving that seat for fare class 1. The probability that the seat could be sold to a fare class 1 is equal to the probability that demand for fare class 1 exceeds the remaining capacity or $P(D_1 \geq x)$. Therefore, Littlewood’s rule is to keep accepting demands for the lower value fare class as long as the following inequality is satisfied. $r_2 > r_1 \cdot P(D_1 \geq x)$. Note that the right hand side of the inequality is the expected revenue that we can get from the seat if we reject the current request for fare class 2 and reserve the seat for future. As the remaining capacity, $x$, decreases, the probability of selling the seat
to a request for fare class 1, \( P(D_i \geq x) \), increases and so does the right hand side of the inequality. Ultimately, for some value of \( x \), the inequality starts to be violated, which means the immediate revenue of selling the seat to the lower value fare class becomes less than the expected revenue of reserving it for the future. Based on Littlewood’s rule that is when we should stop selling seats to the lower value fare class and reserve the remaining seats for the higher value fare class. Therefore, one can calculate the optimal number of seats that should be protected for fare class 1 by solving the following equation for \( x: r_2 = r_1 \cdot P(D_i \geq x) \).

### 2.2.2 DETERMINISTIC LINEAR PROGRAMMING (DLP) MODEL

The following model was first proposed by Glover et. al. in 1982 for a network of flights. In fact this was the first mathematical model proposed for a network of flights. The model solves for the optimal partitioned allocation that maximizes the total revenue. There are two sets of constraints in this model: demand constraints and capacity constraints. Demand constraints make sure the number of seats that are allocated to each ODF is no more than the demand for that ODF. Capacity constraints limit the total number of allocated seats on each leg to the capacity of the leg. Let \( X_{ODF} \) represent the optimal number of seats that are allocated to an ODF. Also let \( D_{ODF} \) and \( r_{ODF} \) denote the demand and fare of ODF respectively. \( L_{ODF} \) represents the set of legs that are used by ODF. The
The model is as follows:

\[
\text{max } \sum_{ODF} r_{ODF}X_{ODF} \\
\text{st. } \sum_{ODF \in L_{ODF}} X_{ODF} \leq C_i \quad \forall i \quad \text{Capacity Constraints} \\
X_{ODF} \leq D_{ODF} \quad \forall ODF \quad \text{Demand Constraints} \\
X_{ODF} \in \mathbb{N}
\]

Note that \( \mathbb{N} \) represents the set of integer numbers. It is a stochastic integer programming model because demand for each ODF, \( D_{ODF} \), is a random variable and \( X_{ODF} \)'s are integers. A common practice is to replace the random demand with its expected value and relax the integrality assumption. The revised model which is often used in practice is called deterministic linear programming (DLP) model. In DLP model, since the random variable \( D_{ODF} \) is replaced with one estimate only, the stochastic nature of demand is not fully captured. Furthermore, the decision variables of this model are partitioned allocations and nesting is not incorporated into the optimization procedure. Nevertheless, this model is being used in practice quite often. One can use DLP model in two different ways. One way is to use the primal solutions, \( X_i \)'s, allow for nesting and implement it with a nested control of the booking process. The second way which is more common in practice is to use the dual solutions. Dual solutions can be used in two different ways, one way is to use them as bid price values for bid-price control of the booking process for the network as a whole and the second is to use the dual
prices for indexing procedure and then solve the network by solving the individual legs.

On a single leg flight, the origin and destination of all itineraries are the same and the only factors that differ among different fare classes are their fares and demand. The adopted DLP model for a single leg flight is presented below, where $n$ represents the total number of fare classes on the leg and $X_i$ denotes the number of seats that are exclusively allocated to fare class $i$. As can be seen, there is only one capacity constraint corresponding to the only leg in the problem. There are $n$ demand constraints, one for each fare class.

$$\text{max} \quad r_1X_1 + \cdots + r_nX_n$$

$$\text{st.} \quad X_1 + \cdots + X_n \leq C$$

$$X_i \leq E(D_i)$$

### 2.2.3 DYNAMIC PROGRAMMING (DP) MODEL

This model was first proposed by Lee and Hersh in 1993. It was designed to find the optimal booking policy for capacity control of a single leg flight. The arrival of requests is assumed to follow a Poisson pattern, therefore this model is categorized as a dynamic model. The main assumptions in this model are that cancellation, no shows and batch arrivals are not allowed.

The elements of any dynamic programming model are: state, value function and a set of decisions at each state of the system. $\langle t, x \rangle$ represents a typical state of this model where $t$ and $x$ denote the remaining time units until the
flight time and remaining capacity respectively. There are total of $C \times T$ states in this DP model, where $C$ denotes the initial capacity of the plane and $T$ represents the total number of time units in the booking period. Let $V_t(x)$ denote the value function at state $\{t,x\}$. The value function is actually the expected total revenue that will be generate from state $\{t,x\}$ until the end of the process. In the dynamic programming concept, this is called revenue-to-go. At each state of the system, two actions are possible: accept or reject the request. Let $u=0$ and $u=1$ denote the decision of rejecting or accepting the request respectively. Note that the optimal decision at each state of the system will depend on the fare class of the request, which is random. The goal is to find the optimal decision policy at each state of the system, so that the overall expected revenue is maximized. Optimal decision at each state of the system is the one that maximizes the expected revenue-to-go at that state. Revenue-to-go at each state consists of two parts, immediate revenue and the expected future revenue. Future revenue is in fact the optimal revenue-to-go at the next state, followed according to the action that was taken at state $\{t,x\}$. The immediate revenue is equal to the fare of the fare class that was received at state $\{t,x\}$ if the decision was to accept the request and equals zero otherwise. Let $R(t)$ be a random variable which equals to the fare of the fare class that was received at time period $t$. If no request was received at time $t$, the value of $R(t)$ would be zero. Like a typical DP model, Bellman’s equations are used to find the value of $V_t(x)$ for all states of the system. Calculations are done backward starting from the flight time by using the boundary conditions. The Bellman’s equations for this DP model are as follows:
\[ V_t(x) = E \left[ \max_{u \in \{0,1\}} \{ R(t)u + V_{t-1}(x-u) \} \right] \]

\[ = V_{t-1}(x) + E \left[ \max_{u \in \{0,1\}} \{ R(t) - \Delta V_{t-1}(x) \} u \right] \]  

(1.1)

where, \( \Delta V_{t-1}(x) = V_{t-1}(x) - V_{t-1}(x-1) \) is the expected marginal value of a seat at state \( \{t-1,x\} \). The boundary conditions are: \( V_0(x) = 0, \ \forall x \) and \( V_t(0) = 0, \ \forall t \).

The optimal decision at each state is to accept the request only if fare of the request is larger than the associated expected marginal value. This type of control is revenue-based or bid-price control. In fact, the expected marginal values are the optimal values of the bid-price for the specific remaining time and capacity. Since this model calculates the optimal decision at each state of the system, the decision is always optimal for any possible combination of remaining time and capacity. Therefore it could be a perfect model for a single leg flight in a sense that the policy is always optimal, however if we extend this model to a network of flights, the number of states in the resulting dynamic programming model would grow exponentially with the number of legs in the network which causes computational difficulty in practice. This is the main reason that this model is not that popular in practice. Furthermore, discrete time units are used to accommodate the Poisson arrival process. The state space is too large even for application to a single leg.

In theory, given the required assumptions, the dynamic programming approach solves the network modes without indexing, nesting or elaborating booking control policies. Attempts have been made to conquer the state space
dimensionality problem by using approximate calculations such as state-space sampling and meta-modeling. In contrast, we use an approximate dynamic programming model and then develop exact calculations for optimizing the booking control policies on each leg.
CHAPTER 3
INTRODUCTION TO THE PROPOSED METHOD

In chapter 2, we looked at some of the existing optimization models for calculating/estimating the optimal policy for capacity control of a single leg flight. Theoretically, the dynamic programming (DP) model is the most precise model to solve for the optimal policy for single leg capacity control given appropriate assumptions. Unlike deterministic linear programming (DLP) models in which the stochastic demand is replaced with one estimate only, DP fully captures the stochastic nature of demand. The problem with the DP model is that its size grows exponentially with the number of legs when it is extended to a network of flights. Airline revenue management problems in real world generally involve a network of interrelated flights. Therefore, a good model not only must work well on a single leg flight but also must be extendable to a network of flights. Even though solving the DP model on a single leg flight is computationally doable, extending it directly to a network of flights is impractical. Even decomposing prices to solve the network as a collection of single legs is computationally burdensome if the classic DP approach is used on each leg. Instead we develop an approximate DP model using the nested booking control policies in order to make a more practical single leg model for solving the decomposed network.
Using DP to calculate the optimal policy is not popular in practice mainly because of the computational difficulty. DLP on the other hand is very popular. A common practice is to use the dual prices of DLP model as an estimate of the bid price values that control booking for a network of flights.

The goal of this study is to come up with a method that works well on a single leg flight, is extendable to a network of flights with affordable computational effort and is competitive with methods that are currently being practiced. In the rest of this chapter we first present the big picture of our proposed method in section 3.1. The general background and technical tools that are used in our proposed method are then discussed in section 3.2.

3.1 BIG PICTURE

In our proposed method for capacity control of a single leg flight, we use an approximate dynamic programming approach involving large time chunks for each stage to achieve single leg performance that is nearly as good as the original DP method (Lee and Hersh 1993). As the time chunks approach zero, and therefore the number of stages increases, our method converges to the original DP method. The practical dynamic programming approach is quite accurate for relatively large time chunks because we use protection levels (PL) and nested control method during each stage. Unlike many of the existing methods, in our proposed method, nesting is incorporated into the optimization procedure. We determine optimal protection levels by direct search, which is different than the use of DLP suggested in the literature. Furthermore, we do not telescope updates
as is common in the literature. In the telescope approach, the optimal policy is updated at the beginning of each stage for the remaining capacity and time, without taking into consideration the possible updating of the booking policy at the future stages until the flight time. In our proposed method however, since we model the process as a dynamic programming model, the updates of the booking policy at the future stages are taken into consideration. We represent a nested booking control at each stage with a Markov chain which enables us to calculate directly the expected revenue outcome of any allocation of seats.

Like Williamson (1992), we will decompose the prices of the multi leg itineraries into individual legs in order to decompose the network into a series of independent single leg problems. We have developed a method for decomposing the prices that is not found in the literature.

3.2 GENERAL BACKGROUND AND TECHNICAL TOOLS

In order to explain our proposed method, first we need to explain some of the concepts, models and codes that we use. In this section, we discuss in detail all the necessary tools that are used in our proposed method.

The rest of this section is organized as follows. In section 3.2.1 we define a nesting table. In sections 3.2.2 and 3.2.3, R and L booking control methods are presented. In section 3.2.4, the proposed Markov chain that models the booking process under implementation of a specific booking policy is presented. Our proposed method for calculating the expected revenue by using the Markov chain model is then presented in section 3.2.5.
3.2.1 THE NESTING TABLE

What we call a nesting table throughout this work is used to visualize and describe our proposed method. The idea of nested allocation on a single leg can be illustrated by drawing a table with $C$ columns and $n$ rows corresponding to capacity and number of fare classes respectively. Each column represents one seat and can be filled with only one entry in any of the rows, which correspond to the fare classes. Let $PL_i$ denote the number of seats that are protected from fare class $i$. On each row $i$, $PL_i$ of the cells are crossed out starting from the right side of the table. For simplicity we use $PL$, a vector of size $n$, to represent the protection levels. Note that throughout this work the first fare class is always the most valuable one and the rest of the fare classes are sorted in the order of decreasing value. So the lowest value fare class has the largest index. Crossed out cells on each row represent seats that are protected from the corresponding fare class and are reserved for higher value fare classes. Figure 3.1 shows the nesting table for an imaginary airplane with capacity of 5, 3 fare classes and protection level vector of $(0,2,4)$.

![Nesting Table](image)

Figure 3.1: A sample Nesting Table, for an airplane with $C=5$, $n=3$ and $PL=(0,2,4)$
In our study, we have realized that standard and theft nesting control methods can be easily illustrated by using the nesting table. Furthermore, two methods “Fill from the Right (R)” and “Fill from the left (L)” booking control methods are proposed that are equivalent to standard and theft nesting control methods respectively. The equivalency of the R and L to standard and theft nesting models is proved and presented in our paper (Haerian et. al, 2006). R and L methods are easily visualized while standard and theft nesting controls methods are always described in the literature as algorithms for accepting or rejecting requests. Furthermore, we have developed a Markov chain that models the process of filling the airplane under implementation of R or the L control method. The Markov chain is then used to calculate the expected revenue gained during the booking period under implementation of R and L or equivalently standard and theft nesting control methods when requests for reservations are Poisson. Then we can easily compare the revenue that is generated under implementation of each of these nested control methods. The comparison between standard and theft nesting control methods had not been discussed in literature until our paper.

3.2.2 FILL FROM THE RIGHT (R) CONTROL METHOD

Suppose $C$, $n$ and either $PL$ or $X$ are given. As before, they represent capacity of the cabin, number of fare classes, set of protection levels and partitioned allocations. We assume that partitioned allocations are calculated using some optimization model such as the one discussed in section 2.2.2. How the $PL$ or $X$ was calculated does not matter at this point. The focus for now is to calculate the total revenue that is gained by using this allocation, under R method.
and for a given string of requests. Note that requests could be for any of the \( n \) fare classes. If \( X \) was given, \( PL \) could be derived from it (as shown in section 1.2.1). The R method fills up the nesting table from the right as booking requests are accepted. More specifically, it can be described as follows:

- Draw the nesting table for the given \( C, n \) and \( PL \)
- If there is a booking request for class \( i \), start from the right side of the \( i^{th} \) row
  - Fill up the first available seat (the first cell which is not crossed out and its corresponding column has no other entry)
  - If there is no such a seat available, reject the booking request.
- Repeat until there is no more booking request or no seat left.

Example 3.1: Let \( C=8, n=3 \) and \( PL=(0,1,7) \) where \( C, n \) and \( PL \) represent the initial capacity of the airplane, number of fare classes and protection levels respectively. Furthermore suppose that there are 12 requests for fare classes 1, 2 and 3 in the following order: 2, 3, 3, 2, 2, 2, 2, 2, 2, 1, 1, 1. Figure 3.2 shows how the R control method is applied to the corresponding nesting table to control the booking process.
3.2.3 FILL FROM THE LEFT (L) CONTROL METHOD

Suppose $C$, $n$ and either $PL$ or $X$ are given. Again, if a set of partitioned allocations, $X$, was given, the set of protection levels, $PL$, could be calculated from it as described in 1.2.1. As mentioned in the previous section, for now we do not care how the allocations are calculated. We just assume that they are given. Our interest is to calculate the total revenue that is generated by using this allocation, under the L control method (or equivalently theft nesting method) and for a given string of requests. Note that requests could be for any of the $n$ fare classes. The L method fills up the nesting table from the left as booking requests are accepted. It is described in more detail as follows:

- Draw the nesting table for the given $C$, $n$ and $PL$
- If there is a booking request for class $i$, start from the left side of the $i^{th}$ row
  - Fill up the first available seat (the first cell which is not crossed out and its corresponding column has no other entry).
  - If there is no such a seat available, reject the booking request.
• Repeat until there is no more booking request or no seat left.

Figure 3.3 shows how the L control method is applied to the corresponding
nesting table to control the booking process.

```
Requests Listed in Order
From Left to Right

Reject

2       3        3         2         2         2         2         2        1        1        1
```

Figure 3.3: Applying L method to control the booking process for example 3.1

3.2.4 “L” AND “R” MARKOV CHAIN MODELS

One of the advantages of the L and R methods over their equivalent theft
and standard nesting control methods is that they give us a visualization that
enabled us to recognize the possibility for developing a Markov chain which
models the booking process under implementation of a booking policy. This
Markov chain model, in turn, yields expected revenue by direct calculations
which allows us to compare the revenue gained by using standard versus theft
nesting control method. Since the Markov chains that model the booking process
under R or L methods are different, we describe them separately below.
Our proposed method is to search over a set of “eligible” PL’s, develop the corresponding Markov chain for each PL, calculate the expected revenue that is generated under the resulting booking policy and find the optimal one. We have found some rules that greatly decrease the number of eligible protection levels and so reduces the computational difficulty.

3.2.4.1 “L” MARKOV CHAIN MODEL

What we call the L Markov chain model throughout this work is a Markov chain that models the booking process under the L control method for a given set of PL. For any leg with capacity $C$, the proposed Markov chain model has $C+1$ states where there are $C$ seats available at state one, $C$-1 seats at state two and finally no seat at the last state (state $C+1$). In other words, at state $i$, $i-1$ seats are already taken and $C+1-i$ seats are still available. Consider an airplane with capacity of 4 and 3 fare classes. Figure 3.2 shows the L Markov chain that models the booking process under the L control method and for $PL=(0,1,2)$. Since $C=4$, the corresponding L Markov chain has 5 states. The label above each state indicates the state number.

Figure 3.4: L Markov chain model for $C=4$, $n=3$ and $PL=(0,1,2)$
In the L Markov chain model, one time unit is equivalent to one transition from state to state. The only possible transitions are from each state to itself or to the one after that. The first state of the L Markov chain is the original nesting table. All other states are the original nesting table with the filled columns eliminated from it. Therefore, the last state, (state 5 in figure 3.4) which shows the end of the process, is the nesting table with all its columns filled. Since there are no more columns to be filled up at the last state, there is no transition from the last state other than to itself. Therefore, the last state is an absorbing state.

The process always starts from the first state, which is the original nesting table where there are $C$ seats available. Based on the L control method, the first available left column can be filled up with eligible fare classes, which are those whose corresponding rows are not crossed out. In figure 3.4, the first left column of the nesting table of state 1 has no crossed out cells and so it can be filled with a request for any of the three fare classes. Since at any state the filled columns are already eliminated from the nesting table, the first available left column is actually the first left column, which can be filled with eligible fare classes. In that case, the column is filled up and transition to the next state occurs which is the previous table with its filled column eliminated from it. The corresponding transition probability is the sum of all probabilities of eligible fare classes at that state, for the above example this probability is equal to $p_1 + p_2 + p_3$ which is the probability of receiving a request from any of the three fare classes in one time unit. Self-transitions occur when there is either no booking request or no eligible
one. The corresponding transition probability is then \( p_0 \) plus the sum of probability of all non-eligible booking requests at that point.

3.2.4.2 “R” MARKOV CHAIN MODEL

What we call The R Markov chain model is a Markov chain that models the booking process under implementation of the R control method for a given PL. The R Markov chain model is similar to The L Markov chain model with more states. As in the L Markov chain model, the number of columns of the nesting table at each state of the R Markov chain model is equal to the remaining capacity of the airplane at that state. Unlike the L Markov chain model where there is only one state corresponding to each possible remaining capacity, in the R Markov chain model there might be multiple states that correspond to a specific remaining capacity. In other words, there might be multiple states whose corresponding nesting tables have the same number of unfilled columns. Transitions happen from each state to itself or to a state with one less column (one less remaining capacity). As before, the probability of transition from a state to itself equals the probability of no request in one time unit plus the probability of requests for all the fare classes that have no more seats to fill. The first state of the R Markov chain model is the original nesting table. All other states are the nesting table with the filled columns eliminated from it. The last state, which shows the end of the process, is the nesting table with all its columns filled and therefore eliminated. Since there are no more columns to be filled up at the last state, there is no transition from the last state. Therefore, the last state is an
absorbing state. Figure 3.5 shows the R Markov chain model for an airplane with $C=4$, $n=3$ and $PL=(0,2,4)$. There are 12 states in the R Markov chain model. The first state is the original nesting table. States 2, 3 and 4 each have 3 columns and correspond to remaining capacity of 3. States 2, 3 and 4 represent the cases where the first, second and third column from the right of the original nesting table are filled respectively.

![Diagram](image)

Figure 3.5: The R Markov chain for an airplane with $C=4$, $n=3$ and $PL=(0,1,2)$.

The process always starts from the first state, which is the original nesting table where $C$ seats are available. Based on the R control method, the first available right column can be filled up with eligible fare classes, which are those whose corresponding rows are not crossed out. The first available right column can be the first, second or any other column of the nesting table depending upon the state of the system and the fare class of the request. That is why in the R
Markov chain model, there is usually more than one state representing the same remaining capacity. The probability of transition from one state to another state of the next stage is the sum of all arrival rates of eligible fare classes at that state. Probability transitions are shown above each arrow in figure 3.5. Transition from state 1 to state 2 could happen only if a request for fare class 1 arrives and the probability of this event is $p_1$, therefore the transition probability from state 1 to state 2 is equal to $p_1$. Similarly, at state 3, if a request for fare class 1 arrives, the first right column will be filled and the process will transition to state 5 at which the first right column is eliminated from the nesting table. Therefore, transition probability from state 2 to state 5 equals $p_1$. A request for fare class 2 or 3 however will fill out the second column from the right of the nesting table at state 3 and the process would move to state 7 from which the filled column is eliminated. The corresponding transition probability is $p_2 + p_3$.

3.2.5 CALCULATING THE EXPECTED REVENUE

In this section we present one way for calculating the expected revenue that is generated during an arbitrary duration of time in the booking period where PL is kept fixed the entire time and the L control method is used to control the booking process. In our study, we have performed numerous numerical experiments in which we searched for the best protection levels and control method together and we observed that R method was rarely the dominant method as far as the expected revenue. Furthermore, when ever the R method generated higher revenue, it was only slightly more than the revenue that was generated.
under L method. Therefore, in our proposed method we only use the L control method. Nevertheless, we have developed a general method of calculating the expected revenue which could be applied to the cases where R or the L control method is used to control the booking process. However because of the simpler nature of the L Markov chain the general method for calculating expected revenue can be specialized and streamlined to reduce computational effort. This further justifies selection of the L method for booking control in the proposed method. Our experiment on the comparison between L and the R control methods as well as our proposed general method for calculation the expected revenue are presented in Haerian et. al (2006). In this section, we present the streamlined method for calculating the expected revenue that is generated in an arbitrary duration of time in the booking period where PL is fixed throughout this time and where L method (theft nesting method) is used to control the booking process.

We use the corresponding L Markov chain model to calculate the expected total revenue.

Let $T$ represent the length of the booking period and $C$ denote the remaining capacity of the airplane at the beginning of it. For any given PL we can develop a L Markov chain which models the booking process under implementation of L booking control. The L Markov chain model has a nice and simple structure that allows us to calculate the conditional expected revenue that is generated during the booking period conditioned that on knowing the remaining capacity at the end of the booking period. Furthermore, the probability of any possible remaining capacity at the end of the booking period can be derived from
the $P^T$ matrix where $P$ is the probability transition matrix of the L Markov chain and $P^T$ shows the state of the system at the end of the booking period.

The L Markov chain model has $C+1$ states. The number of unfilled columns of the nesting table at each state indicates the remaining capacity of the airplane. At the first state of the L Markov chain model all columns are unfilled which corresponds to the remaining capacity of $C$. The second state of the L Markov chain has $C-1$ unfilled columns which means the remaining capacity is $C-1$ or in other words means that one request has been accepted. In the L control method, the columns of the nesting table are filled one by one from the left. Therefore if we know the remaining capacity at the end of the booking period we would know exactly which columns of the nesting table are filled. Furthermore since $PL$ is known, so is the expected revenue that is gained from each column. The expected revenue of each column is in fact a weighted average over the fares of open fare classes. Note that by open fare classes we mean the ones whose corresponding cell on the column is not crossed out. In figure 3.6 we present a sample nesting table and the expected revenue of each of its columns.
Figure 3.6: Expected revenue gained from each of the columns of the nesting table for $C=4$, $n=3$ and $PL=(0,2,3)$

In this picture, $R_i$ denotes the expected revenue that is gained from the $i$-th column of the nesting table and $p_i$ represents the probability of arrival of a request for fare class $i$ in one time unit and $r_i$ denote the fare of the $i$-th fare class. The expected revenue that can be gained from the first column of the nesting table can then be calculated as follows. $R_1 = \frac{r_1 p_1 + r_2 p_2 + r_3 p_3}{p_1 + p_2 + p_3}$ which is in fact the weighted average over the fares of fare classes 1, 2 and 3. Similarly, we can calculate the expected revenue of the second left column of the original nesting table. Let $R_2$ denote this value. The second left column of the original nesting table is open to fare classes 1 and 2. So $R_2$ is the weighted average over fare classes 1 and 2 or $R_2 = \frac{r_1 p_1 + r_2 p_2}{p_1 + p_2}$. The third and fourth columns are only open to fare
class 1 and therefore $R_3 = R_4 = r_1$. Figure 3.7 shows the L Markov chain corresponding to the nesting table of figure 3.6. The shaded columns at each state show the filled columns.

![Figure 3.7: L Markov chain model with shaded filled columns for $C=4$, $n=3$ and $PL=(0,2,3)$]

At the end of the booking period, the process might be at any of the 5 states of the corresponding L Markov chain. Note that at state 1, no column is filled out and therefore no revenue is gained. At state 2, the first left column is filled out and the expected revenue of $R_1$ is gained. At state 3, the first two left columns are filled out and therefore the expected revenue of $R_1 + R_2$ is gained. Similarly at state 4 and 5 expected revenue of $R_1 + R_2 + R_3$ and $R_1 + R_2 + R_3 + R_4$ are gained respectively. Let $\mathbf{P}$ represent the transition probability matrix of the corresponding L Markov chain. The first row of $\mathbf{P}^T$ represents the state probabilities of the system after $T$ time units. In general, $\{\mathbf{P}^T\}_{(i,j)}$ refers to the element $(i,j)$ of matrix $\mathbf{P}^T$ and shows the probability of starting from state $i$ and ending up at state $j$ after $T$ time units. Note that the booking process always starts at the first state of the Markov chain where all the seats are still available. By using the first row of $\mathbf{P}^T$ and the expected revenues of
the columns we can actually calculate the expected revenue that is gained throughout the entire $T$ time units as follows.

\[
\begin{align*}
& \left\{ \left( p^T \right)_{i,1} \times 0 \right\} \\
& + \left\{ \left( p^T \right)_{i,2} \times R_1 \right\} \\
& + \left\{ \left( p^T \right)_{i,3} \times (R_1 + R_2) \right\} \\
& + \left\{ \left( p^T \right)_{i,4} \times (R_1 + R_2 + R_3) \right\} \\
& + \left\{ \left( p^T \right)_{i,5} \times (R_1 + R_2 + R_3 + R_4) \right\}
\end{align*}
\] (3.1)

3.3 SUMMARY

In this section we presented the necessary background and technical tools that are used in our proposed method of capacity control. We introduced the nesting table and showed how to use it to visualize two of the nesting control methods that exist in the literature: standard and theft nesting control methods. We presented L and the R control methods that are equivalent to theft and standard nesting methods respectively and proposed a Markov chain that models the booking process under implementation of each of the nested control methods.

Furthermore, we presented a fast and efficient method for calculating the expected revenue that is generated during the booking period for a given $C, T, n, r, p$ and $PL$ under the L control method.

In the next chapter we present our proposed method for single leg capacity control. In our proposed method we develop an approximate dynamic programming model with large time chunks. The technical tools and calculator that we presented in this chapter are then used to solve the proposed dynamic programming model. Note that by solving the proposed DP we intended to find
the optimal protection levels at each stage of the proposed dynamic programming model and for any possible remaining capacity.
CHAPTER 4
PROPOSED METHODOLOGY

The goal of capacity control in airline revenue management is to control the booking process so that the revenue that is generated during the booking period is maximized. Generated revenue directly depends on the booking policy as well as demand during the booking period. Demand is stochastic so the optimal booking policy would be the one that “on the average” generates the highest revenue or in other words maximizes the expected total revenue. A booking policy includes a control method, such as standard nesting control or bid price control methods plus a set of control parameters, which correspond to the control method of use. For example if standard or theft nesting control method is used, the control parameters are the protection levels. However, if the bid price control method is used to control the booking process, control parameters are bid price values. There are many different methods, algorithms and heuristics for finding the optimal booking policy. A good method is the one that on the average generates higher revenue compared to other methods with affordable computational effort. These two factors are often in contradiction and there is a trade off between generating high revenues and the computational effort that is
required for calculating/estimating the optimal policy. The goal of this study is to
develop a capacity control method that works well on a single leg flight, (on the
average generates more revenue than existing methods), is extendable to a
network of flights with affordable computational effort and, finally, is competitive
with existing methods for network capacity control.

As mentioned earlier, in most of the existing optimization models that
solve for optimal protection levels, the nesting characteristic of the allocation is
not incorporated into the optimization procedure. In fact the common
recommendation is to use an optimization model which solves for the optimal
partitioned allocation and allow for the nesting when implementing it in practice.
Incorporation of the nesting into the optimization procedure has been addressed in
the literature (notably in Bertsimas and de Boer, 2005, and van Ryzin and
Vulcano, 2003), and some algorithms have been proposed. Because of the
stochastic nature of the problem, these algorithms implement some complex
gradient-based simulation-optimization procedures. Our approach as outlined in
Chapter 3 allows us to address that problem from a different perspective than the
existing methods and, in fact, it does incorporate the nesting property into the
optimization procedure. Another advantage of our proposed method over existing
methods is that since it calculates the expected revenue by direct calculations
from the Markov chain model, it does not involve simulation. However, when we
extend our proposed method to a network of flights, in order to compare the
performance of our proposed method with other existing methods, simulation is
the only option.
In this chapter, we present our proposed method for finding the optimal booking policy on a single leg flight. Our proposed method for capacity control of a single leg flight involves a dynamic programming model and a heuristic to look for the best decision at each stage of the dynamic programming model and for every possible remaining capacity. At each stage of the proposed DP and for any remaining capacity, we use the corresponding Markov chain models (presented in section 3.2.4) to calculate the objective value. Note that in our proposed method the booking period is divided into $S$ stages. The value of $S$ is defined by the user and represents the number of times that he/she wants to update the policy during the booking period. When the value of $S$ equals the number of time units in the booking period, our approximate DP model would be exactly like the original DP (Lee and Hersh, 1993). Our proposed methods for capacity control of a network is to first decompose the prices of the multi leg itineraries into individual legs by using our proposed decomposition method and then apply our proposed single leg capacity control method on individual legs. These steps for the network problem are presented in Chapter 5.

The remainder of this chapter is organized as follows. We will first explain our propose method for single leg capacity control in section 4.1 and then present our numerical experiment on a single leg flight in section 4.2.
4.1 PROPOSED METHOD FOR FINDING THE OPTIMAL POLICY FOR CAPACITY CONTROL OF A SINGLE LEG

As mentioned earlier, the goal in our proposed method of single leg capacity control is to find the best booking policy at the beginning of each stage for any possible remaining capacity. Our nesting table is not only able to model the booking process under a nested control policy but one can also use it to model the booking process under the bid price control policy. In fact when protection levels on all the fare classes are either zero or equal to the remaining capacity, the L control method would be equivalent to the bid price control where the bid price value is equal to the fare of the lowest fare class with protection level of zero. So, one can use our L Markov chain to model the booking process under implementation of bid price control as well as a nested booking control. Therefore by using our proposed method of capacity control one can search for the optimal booking policy at the beginning of each stage, where the booking control method could be bid price control, theft nesting control or standard nesting control. However in our research, we have decided to limit ourselves to theft nesting control (L method), mainly for two reasons. The first reason is that we have performed many numerical experiments in which we searched for the optimal PL for the L method of booking control and then re-optimize for the R method of booking control. As mentioned previously, we found that the L control method generated higher expected revenue almost all the time and when it did not, its performance was only slightly less than the optimal R method. Our study on comparing L and R methods are presented in our paper (Haerian et. al, 2006).
The second reason for our choice is that the L Markov chain model is much simpler than the R Markov chain model, which makes the calculations less intense and the overall method more efficient. For these two reasons, we have decided to keep the L booking method as the primary control method during the booking period and search only for the optimal protection levels.

In our proposed method, we divide the entire booking period into $S$ stages and update the optimal policy at the beginning of each stage. The value of $S$ is defined by the user. Of course, a small value for $S$ would mean less computational effort. As we increase $S$, the computational effort to solve our model grows linearly, but the expected revenue that is generated under our proposed method converges to the one generated under the DP method. Stages may or may not be of the same length. Note that length of one time unit is constant. Therefore, stages may contain different numbers of time units. We also assume that demand arrives according to a homogeneous Poisson process. However, we should point out that assuming that the process is homogeneous is not necessary for our approach, but the assumption does reduce the data requirements. Our proposed method involves a dynamic programming model and a heuristic to search for the best booking policy at each of its states. Let us refer to the proposed dynamic programming model as the PDP model. By using the PDP, we model the booking process of a single leg throughout the entire booking period. The heuristic includes three rules which greatly improve the efficiency of the search for the optimal policy at each state of the PDP model. We will present the PDP model and the rules in sections 4.1.1 and 4.1.2 respectively.
4.1.1 PROPOSED DYNAMIC PROGRAMMING (PDP) MODEL

In any dynamic programming model, the state of the system, value function and decisions at each of the states need to be defined. The goal is to find the optimal decision at each state, the one that optimizes the value function at that state. Depending upon the problem, we either want to maximize or minimize the value function.

Our proposed dynamic programming model, which we will refer to as PDP, models the booking process under the L control method throughout the booking period where the optimal set of protection levels are updated \( S \) times. Let \( s \) denote the stage number and \( c \) the remaining capacity of the airplane at the beginning of a stage. \( s \) could change from 0 to \( S \) and \( c \) from 0 to \( C \). Note that stage zero is the flight time. In stages 1, 2, ..., \( S-1 \) there are \( C+1 \) states, corresponding to the remaining capacity of the airplane at the beginning of the stage. In other words, state \( c \) of stage \( s \) corresponds to the beginning of stage \( s \) where there are \( c \) seats remaining. The booking process starts at the beginning of stage \( S \) and the initial capacity of the airplane is \( C \), therefore there is only one state at stage \( S \) corresponding to the remaining capacity of \( C \). State zero at stage \( s \) corresponds to the remaining capacity of zero at the beginning of stage \( s \). The booking process could end at stage zero (flight time) regardless of the remaining capacity at that time or state zero at any of the stages. In other words, the booking process ends either if we reach the flight time or if the remaining number of seats becomes zero. Note that if we allowed cancellation in our model, the booking
process could have ended only at the flight time. However, since we assume there is no cancellation, the booking process will end once the remaining capacity becomes zero regardless of the stage. The value function at each state of the PDP model is the expected total revenue that will be generated from that state until the end of the process. In dynamic programming, this is called revenue-to-go. The goal is to maximize the expected total revenue over the entire booking period, from the beginning of stage $S$ where there are $C$ seats remaining until one of the states that represent the end of the process. In other words, we want to maximize the revenue-to-go at state $c$ of stage $s$. Let $PL$ stand for a vector containing protection levels on all the fare classes. The decision that needs to be made at each state of PDP is to find out which $PL$ maximizes the value function at that state. By solving the PDP model, we mean to find the optimal decision, $PL$, and optimal objective value at each and every state of the PDP model.

Like a typical dynamic programming model, PDP is solved backward starting from the end of the process. In airline revenue management that means the problem is solved starting from the last stage before the flight time. In other words, first we find the optimal $PL$ at the beginning of stage 1 (when there is one stage remaining until the flight time) and for every possible remaining capacity. Next we find the optimal PL at the beginning of the second stage for every possible remaining capacity and so on. At each stage, we first solve the problem for state 1 with the remaining capacity of 1 then state 2 and so on until state $C$ where the remaining capacity is equal to $C$. Finally the last step is to find the
optimal PL at the beginning of the process (S stages before the flight time) and for remaining capacity of C.

After solving the PDP model, we will have optimal PL for each stage of the model and for any remaining capacity, in other words we know what the optimal PL should be for each possible combination of remaining capacity and remaining number of stages before the flight time. In order to implement this solution in practice, one needs to look up the proper PL for state c of stage s and update the policy accordingly at the beginning of each stage.

In the remainder of this chapter we use the following notation.

\(\mathbb{N}_{s,c} \): The set of all possible nested protection levels at stage s of the PDP model and with c seats available. Later in section 4.1.2, we will present three rules that decrease the number of possible nested protection levels. As we will see then, the final set of possible nested protection levels derived by these rules depends on the stage as well as the remaining capacity.

\(E^*(s,c)\) denotes the value function, or revenue-to-go at stage s and with c seats available in the PDP model, i.e., it is the maximum expected total revenue during stage s and all subsequent stages given c seats available at the beginning of stage s.

\(R_{i,PL}\) denotes the expected revenue that can be gained from the \(i^{th}\) column of the nesting table representing the particular protection level PL, as defined in Chapter 3 without the PL subscript. When \(i = 0\), \(R_{i,PL}\) is defined to be 0 for all PL.

\(T_s\) shows the number of time units in stage s of the PDP model.
\( \mathbf{P}_{PL,c} \) is the one step transition matrix defined in Chapter 3, but with the \( PL \) and \( c \) subscript added to denote that its elements depend on the protection levels and the remaining capacity.

\[ \left\{ \mathbf{P}^T_{PL,c} \right\}_{1,k} = \text{The element in row 1 and column } k \text{ of the transition matrix raised to the power } T_s, \text{ thus it is the probability that the process ends up at the } k\text{-th state of the } L \text{ Markov chain at the end of stage } s \text{ or in other words there will be } c + 1 - k \text{ seats available at the end of stage } s. \]

As discussed in section 3.2.5, we can develop the \( L \) Markov chain model for any given remaining capacity and any interval for a given PL. Suppose at the beginning of stage \( s \), the remaining capacity is \( c \) and that PL is implemented during stage \( s \). \( \mathbf{P}^T_{PL,k} \) is the probability that at the end of stage \( s \) (after \( T_s \) time units), we end up at state \( k \) of the corresponding \( L \) Markov chain. State \( k \) of the \( L \) Markov chain is where \( k-1 \) requests are accepted during \( T_s \) time units and so the remaining capacity at the end of the interval is \( c-k+1 \). The expected revenue that is generated during stage \( s \) in this case is in fact the expected revenue that is generated from \( k-1 \) left columns of the corresponding nesting table or \( \sum_{i=0}^{k-1} R_{i,PL} \).

The future revenue that will be generated from the end of stage \( s \) until the flight time is \( E^*(s-1,c+1-k) \) which is actually the optimal revenue to go at state \( s-1 \) of the PDP model and with \( c+1-k \) seats available. \( \sum_{i=0}^{k-1} R_{i,PL} + E^*(s-1,c+1-k) \)

would then represent the conditional expected total revenue that is generated from the beginning of stage \( s \) until the flight time where the remaining capacity at the
The beginning of stage is \( c \) and PL is implemented during the stage conditioned that \( k-1 \) request were accepted during stage \( s \). Conditional probabilities are known and therefore we can calculate the expected total revenue at stage \( s \) of the PDP model with \( c \) seats available where PL is implemented during stage \( s \) as follows.

\[
\sum_{k=1}^{c+1} \{ P_{PL,c}^r \}_{1,k} \left[ \sum_{i=0}^{k-1} R_{i,PL} + E^* (s - 1, c + 1 - k) \right] \quad (4.1)
\]

In order to find the optimal revenue to go at stage \( s \) of the PDP model with \( c \) seats available or \( E^* (s, c) \), we then need to search over \( \mathcal{N}_{c,n} \).

Like a typical dynamic programming model, we can use Bellman’s equation to solve the PDP model. Table 4.1 summarizes the notation that is used in the Bellman’s equation.

Let \( \mathcal{N}_{s,c} \) = The set of all possible nested protection levels at stage \( s \) of the PDP model with \( c \) seats available

\( E^* (s, c) \) = The value function, or revenue-to-go at stage \( s \) of the PDP model with \( c \) seats available

\( R_{i,PL} \) = The expected revenue that can be gained from the \( i^{th} \) column of the nesting table representing the particular protection level \( PL \)

\( T_s \) = The number of minute time periods in stage \( s \) of the PDP model.

\( P_{PL,c} \) = The one step transition matrix

\( \{ P_{PL,c}^r \}_{1,k} \) = The element in row 1 and column \( k \) of the transition matrix raised to the power \( T_s \), thus it is the probability that there will be \( c + 1 - k \) seats available at the end of stage \( s \).

**Table 4.1:** Summary of notation

Bellman’s Equation:

\[
E^* (s, c) = \text{Maximum}_{PL \in \mathcal{N}_{s,c}} \sum_{k=1}^{c+1} \{ P_{PL,c}^r \}_{1,k} \left[ \sum_{i=0}^{k-1} R_{i,PL} + E^* (s - 1, c + 1 - k) \right] \quad (4.2)
\]

With boundary conditions:
\[
\begin{align*}
E^\ast(0, c) &= 0 \forall c \\
E^\ast(s, 0) &= 0 \forall s
\end{align*}
\] (4.3)

The total number of states in the PDP model is \(1 + S \times (C + 1)\) and the total number of states in the original dynamic programming model is \(1 + T \times (C + 1)\) where \(T\) denotes total number of time units in the booking period. As mentioned before, in order to be consistent with the Poisson assumption, \(T\) has to be small enough so that the probability of receiving more than one request in one time unit becomes negligible. This means one time unit is very small which would make \(T\) very large. On the other hand, \(S\) represents the total number of times that the booking policy is updated throughout the booking period. Depending upon the value of \(S\), the number of states in the PDP model can be considerably less than the original dynamic programming model. Having fewer states in a dynamic programming model can greatly decrease the computational difficulty. Note that if we set \(S\) equal to the number of time units in the booking period the resulting expected total revenue would be equal to the expected total revenue that is generated under implementation of the booking policy calculated by the original DP model. When \(S\) is smaller than the total number of time units in the booking period, the calculated policy is not always optimal and there may actually be several arrivals before the policy is updated. Note that whenever a time unit is passed or a new arrival comes, the policy is no longer optimal for this new situation. Given that the policy will be updated \(S\) times during the booking period, the number of arrivals between two successive updates could be anywhere from zero to minimum of \(T\) and \(C\). However, as we will show in numerical
experiment in chapter 4.2, in most cases when the policy is updated several times during the booking period, the expected total revenue that is generated under the PDP model is reasonably close to the one generated under the original dynamic programming model.

4.1.2 DECREASING THE SIZE OF THE SET OF PROTECTION LEVELS

Optimal protection levels at stage $s$ of the PDP model with $c$ seats available is found by searching over a set of eligible protection levels which we refer to as $\mathbb{N}_{s,c}$. The total number of possible protection levels can be very large which could make the search very time consuming. In our study, we have developed three rules that greatly decrease the size of the set of eligible protection levels and improve the efficiency of the search. The rules should actually be used within solving the states of the PDP model. The first rule uses the solution of stage $s$ of the PDP model and sets a lower bound for the number of seats that should be protected from each fare class on stage $s+1$. Rule 2 uses $PL^*(s,c)$ to set an upper bound on the number of seats that should be protected from each fare class at stage $s$ for the remaining capacity of $c+1$. Rule 3 is designed to improve the efficiency of the search over $\mathbb{N}_{s,c}$. These three rules are presented in the next three sections.
4.1.2.1 RULE 1: A LOWER BOUND ON NUMBER OF SEATS THAT
SHOULD BE PROTECTED FROM EACH FARE CLASS

This rule uses the solution of stage \( s-1 \) of the PDP model to set a lower
bound on the number of seats that should be protected from each fare class at
stage \( s \). The basic idea is that a seat should not be sold to any fare class during
stage \( s \) if the revenue from that fare class is less than the marginal value of the
seat at stage \( s-1 \).

Since rule 1 requires the optimal solution of the previous stage of the PDP
model to set the lower bounds on any stage, it applies to all stages of the PDP
model except stage 1 which is the first stage of the PDP model and the last stage
before the flight time. That is because the marginal values after flight time are all
zero.

Suppose that we solve stage \( s-1 \) of the PDP model and summarize the
solutions as in figure 4.1, where \( PL^* \) are the optimal protection levels, \( E^* \) is the
optimal value function, and \( M \) are marginal values discussed below.

<table>
<thead>
<tr>
<th>Stage ( s-1 )</th>
<th>( c )</th>
<th>( PL^* )</th>
<th>( E^* )</th>
<th>( M )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>( PL^*(s-1, 1) )</td>
<td>( E^*(s-1, 1) )</td>
<td>( M(s-1, 1) )</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>( PL^*(s-1, 2) )</td>
<td>( E^*(s-1, 2) )</td>
<td>( M(s-1, 2) )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( C )</td>
<td>( C )</td>
<td>( PL^*(s-1, C) )</td>
<td>( E^*(s-1, C) )</td>
<td>( M(s-1, C) )</td>
</tr>
</tbody>
</table>

Table 4.2: Solution of stage \( s-1 \) of the PDP model in a tabular format

Consider stage \( s-1 \) with \( c \) seats available, where there are \( s-1 \) stages
remaining until the flight time and the remaining capacity is \( c \). \( PL^*(s-1,c) \), \( E^*(s-1,c) \) and \( M(s-1,c) \) represent the corresponding optimal protection level, optimal
revenue to go and marginal revenue. Marginal revenue at stage \( s-1 \) with \( c \) remaining capacity is defined as follows, \( M(s-1,c) = E^*(s-1,c) - E^*(s-1,c-1) \) where \( R^*(s-1,0)=0 \). \( M(s-1,c) \) represent the marginal revenue of an additional seat at state \( c \) of stage \( s-1 \).

Rule 1 can be summarized as follows:

\[
\begin{cases}
M(s-1,k) > r(i) \quad \text{and} \quad M(s-1,k+1) < r(i) \\
\end{cases}
\]

if

\[
\begin{cases}
P_L^r(i) \geq k \quad \text{for} \quad c \geq k \\
P_L^r(i) = c \quad \text{for} \quad c < k
\end{cases}
\]

(4.4)

What this means in words is that if the marginal revenue of seat \( k \) of stage \( s-1 \) is larger than fare of fare class \( i \), then at least \( k \) seats should be protected from fare class \( i \) for all remaining capacities at stage \( s \). Furthermore, protecting at least \( k \) seats from a state with remaining capacity of less than \( k \) would mean to protect all of the seats from that fare class. It has been shown in the literature (Lautenbacher and Stidham, 1999) that the marginal revenues are non-increasing with respect to remaining capacity. In other words:

\[
M(s,c) \geq M(s,c+1) \quad \forall s, c.
\]

(4.5)

Therefore, if \( M(s-1,k) \geq r(i) \) then,

\[
M(s-1,1) \geq r(i), M(s-1,2) \geq r(i), ..., M(s-1,k-1) \geq r(i).
\]

According to rule 1, since the 1\(^{st}\), 2\(^{nd}\), ..., \( k \)-th seat are larger than fare of fare class \( i \), we should protect all of those seats from fare class \( i \) and reserve them for future (stage \( s-I \)).

The rationale is intuitive. Suppose \( M(s-1,k) > r(i) \) for some value of \( k \). This means that the future expected revenue that will be generated from the \( k \)-th seat on stage \( s-I \) is larger than the immediate revenue that could be generated if we accept a request for fare class \( i \) during stage \( s \). It is easy to see why we should
protect the $k$-th seat from fare class $i$ on stage $s$ and reserve it for the future (stage $s-1$), simply because it is more valuable to reserve the seat for the future rather than selling it to a request for fare class $i$ or lower value fare classes. Since the marginal revenues are non-increasing functions of remaining capacity, this argument would hold for all remaining capacities of less than $c$ as well. In other words, $M(s-1,1) \geq M(s-1,2) \geq \cdots \geq M(s-1,c-1) > r(i)$. Therefore by the same argument, seats $c-1, c-2, \ldots, 1$ have to be blocked from fare class $i$ on stage $s$ and reserved for the future (stage $s-1$). Because the marginal revenues of those seats are greater than the immediate revenue of selling those seats to fare class $i$. In our study, we have observed that the minimum protections that are calculated by rule 1 get larger as $s$ increases which means as we solve the problem stage by stage, rule 1 helps to reduce the size of the corresponding $N_{s,c}$ more. In fact, for larger values of $s$ (further from the flight time), minimum protection levels on most of the lower value fare classes are often equal to the remaining capacity, which simply means all the remaining seats should be protected from those fare classes. This could make the size of the resulting $N_{s,c}$ as low as 1.

4.1.2.2 RULE 2: AN UPPER BOUND ON THE NUMBER OF SEATS THAT SHOULD BE PROTECTED FROM EACH FARE CLASS

When solutions are developed at any stage $s$, they must be developed for every possible number of available seats, $c$. We do this in the order of increasing $c$ starting with $c = 1$, (since $c$ is the remaining capacity it can also equal 0, but there is nothing to solve at that state). This enables us to use the information from the optimal protection levels at state $c$ to reduce the search space of protection
levels at state $c + 1$. Rule 2 applies to all states of the PDP model. The intuition for this rule is based on the fact that with a fixed amount of time left until take off the probability of flying with some vacant seats increases as the capacity increases, all other things being equal. Such a situation provides pressure to open seats to more fare classes as capacity increases. To illustrate with an extreme case suppose three minutes before take off there are 100 empty seats and one or two reservation requests are expected in those three minutes it is clear that every request would be accepted no matter what the fare class happens to be. Thus, the idea behind this rule is that if the seat represented in the nesting table by column $i$ is open to a fare class with $c$ total seats available, then if $c+1$ seats are available the seat, $i$, should remain open.

Thus, for any stage $s$, rule 2 uses $PL^{*}(s, c)$, the optimal protection level at state $c$, to set an upper bound on number of seats that should be protected from each fare class at state $c+1$. Let table 4.3 represent the optimal solution of stage $s$ of the system in a compressed format.

<table>
<thead>
<tr>
<th>Stage $s$</th>
<th>State (Remaining Capacity)</th>
<th>Optimal Protection Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$PL^*(s,1)$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$PL^*(s,2)$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>$PL^*(s,c)$</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Optimal PL’s on stage $s$ of the PDP model

After solving the first row, state 1 of the PDP model, we can use $PL^*(s,1)$ to set the maximum for some or all protection levels of the problem of the next row, state 2. Rule 2 can be summarized as follows,
Rule 2: For all $i$, if $\{PL^i(s, c)\}_k = k$ and $k < c$, then $\{PL^i(s, c + 1)\}_k < k$,

where $\{PL^i(s, c)\}_k$ denotes the optimal protection level for fare class $i$.

In other words, this means that if it is optimal to leave $N$ seats open to fare class $i$ when the remaining capacity is $c$, then it would be optimal to leave at least $N + 1$ seats open to fare class $i$ when the remaining capacity is $c + 1$, but $N$ must be strictly positive to apply the rule. If $N$ is not strictly positive, it means that there is insufficient capacity to open the fair class, which may also be true for the next several capacities as well. The maximum protection level for the class occurs at the capacity level where the class is first opened.

At stage 1 of the PDP model for which the future revenues are zero, it is possible to prove the validity of the bound as follows. In order to find the optimal protection levels at state 1, we search over $\mathbb{N}_{i,c}$ to find the PL that maximizes the expected total revenue, where the expected total revenue is equal to:

$$
\sum_{k=1}^{c+1} \{p^*_{PL,c}\}_k \sum_{l=0}^{k-1} R_{i,PL}
$$

Suppose that $\{PL^i(1, c)\}_k = k$ where $k < c$. Note that the above sum takes into consideration the expected total number of requests that would arrive during $T_i$ time units of stage 1 and the optimal number of seats that should be protected from each fare class is found accordingly. Let $N_{i,2,\ldots,i-1}$ denote the expected total number of requests for fare classes $1, 2, \ldots, i - 1$ during stage 1. If $N_{i,2,\ldots,i-1}$ is not large enough to fill out the remaining $c$ seats throughout stage 1, then the optimal solution at the beginning of stage 1 would be to leave some of the seats open to
fare classes $i, i+1, \ldots, n$ to prevent the airplane from flying empty. In other words, the optimal number of seats that should be protected from fare classes $i, i+1, \ldots, n$ will be less than the remaining capacity. Now consider the problem with $c+1$ available seats. Again there are $T_i$ time units until the flight time. If $N_{1,2,\ldots,j-1}$ was not large enough to fill out $c$ remaining seats throughout stage 1 then for sure it is not large enough to fill out more than $c$ remaining seats in that time either. Therefore the optimal PL at state $c+1$ would be to protect at most $k$ seats from fare class $i$.

For stages other than stage one, future revenues are no longer zero. This makes the problem much more complicated, since the optimal protection level on fare class $i$ is affected not only by the expected number of requests for higher-valued fare classes during the stage, but also by the future revenues which, in turn, are a function of remaining capacity at the end of the stage. The reason is that in this case there is another factor that affects the optimal protection levels as well. When future revenues are not zero, an optimal protection level would be the one that maximizes the immediate plus future expected revenue. It has been a great disappointment that we have been unable to prove the validity of the upper bound on protection levels, even by induction. However, to test our conjecture we performed numerous numerical experiments, in which rule 2 always worked for all stages other than stage 1. In fact, applying rule 2 to restrict the search space never once caused us to miss the optimal PL. As a result we have a great deal of confidence in the validity of rule 2, but the lack of a proof causes us to
classify our solution method for the PDP as an heuristic when we restrict the search space with rule 2.

4.1.2.3  RULE 3: MULTI DIMENSIONAL LINE SEARCH

After using rules 1, and 2 the resulting $\mathcal{N}_{s,c}$ would include all possible $PL$’s for stage $s$ and remaining capacity $c$ of the PDP model. The next step is to search over this set to find the $PL$ that generates the maximum expected total revenue. The most naïve search is total enumeration, when one tries all the members of $\mathcal{N}_{s,c}$ one by one to find the optimal one. Rule 3 however is designed to make the search over $\mathcal{N}_{s,c}$ efficient. At any stage of the PDP model, for example stage $s$, and for any remaining capacity say $c$, the expected total revenue that is generated under the L control method is unimodal with respect to $(PL(s, c))$, and for all values of $i$. In other words, the expected total revenue is monotonically increasing for $(PL(s, c)) \leq m$ and monotonically decreasing for $(PL(s, c)) \geq m$.

The value of $m$ depends on many factors, like remaining capacity, rate of arrivals and fares to name a few. The unimodality property could be used to efficiently search over $\mathcal{N}_{s,c}$ for the optimal PL. In fact we can perform a multi dimensional line search when each fare class is considered as one dimension. The procedure is as follows: first we set all of the protection levels to the lower bound calculated by rule 1. We then iteratively increment $(PL(s, c))_n$ while all other protection levels are fixed and calculate the corresponding expected total revenue. We shall keep increasing $(PL(s, c))_n$ until the expected total revenue starts to decrease at
which point we stop the search for \( \{PL(s,c)\}_n \) and fix it to the previous value. The next step is to look for the protection level on fare class \( n-1 \) that maximizes the expected total revenue while fare classes \( 1,2,\ldots,n-2 \) are set to their lower values (calculated by rule 1) and the protection level on fare class \( m \) is found as described above. As before, we shall keep incrementing \( \{PL(s,c)\}_{n-1} \) by one while every other protection level is fixed and calculating the resulting expected total revenue. Once we reach at the point where the resulting expected total revenue starts to decrease, we stop the search and set \( \{PL(s,c)\}_{n-1} \) to the previous level. This is repeated for all the fare classes in the order of increasing fares. Once we are done with all the fare classes we iterate this whole procedure from fare class \( n \) again. Note that on the next iteration we start the procedure with the most recent protection levels. This process is repeated for several iterations until the resulting protection levels are no different than the ones found on the last iteration.

In our experiments, the optimal \( PL \) was usually found in at most three iterations, where on the second iteration there were only a few changes to make. Rule 3 helps us to greatly improve the efficiency of the search.

We ran some experiments to investigate how effective each rule is in reducing the size of the set of eligible PL’s. The exact reduction of the set by each rule is different for each specific problem and depends on different parameters such as the number of stages, capacity and etc. On the average, we observed that rule 1 reduces the size of the eligible PL’s by about 87%. We found rule 2 to be the most effective of all three rules.
4.1.2.4 EMPIRICAL STUDY ON EFFECT OF THE THREE RULES ON REDUCING THE SIZE OF THE SET OF ELIGIBLE PROTECTION LEVELS

We did some empirical study to investigate how effective the three rules are in reducing the size of the set of eligible protection levels. If no rule is applied, the size of the set of protection levels is a function of the remaining capacity as well as the number of fare classes. It is difficult to find the closed formula for the size of the set as a function of these two parameters. However it is easy to write a computer programming code to do the calculations. In fact, we can use \( n \) nested loops to count the number of possible protection levels where \( n \) denotes the number of fare classes. On each row of the nesting table, 0, 1, 2, \ldots or \( C \) cells may be crossed out and therefore protected from higher value fare classes. Furthermore, in order to stay consistent with the nesting property, the number of cells that are crossed out on each row can not be less than the number of seats that are crossed out on the previous row. The pseudo code for calculating the size of the set of eligible protection levels for fixed \( C \) and \( n \) is presented below.

\[
\text{for } i_1 = 0 \text{ to } C \\
\quad \text{for } i_2 = i_1 \text{ to } C \\
\quad \quad \ldots \\
\quad \quad \text{for } i_n = i_{n-1} \text{ to } C \\
\quad \quad \quad \text{SIZE} = \text{SIZE} + 1 \\
\quad \text{end} \\
\ldots
\]
Note that count denotes the variable SIZE represents the size of the set of eligible protection levels.

In order to investigate how the size of the set grows with the capacity, we ran some numerical experiments. In our experiment we fixed the number of fare classes to 3 and incremented the capacity from 1 to 100, for each capacity we used calculated the size of the set of eligible protection levels as explained in the previous paragraph. We then changed the number of fare classes and repeated the experiment. We realized that the size grows exponentially with the capacity and this property remains the same if we change the number of fare classes. Figure 4.1 shows $N$ versus capacity where $N$ denotes the size of the set of eligible protection levels.
Figure 4.1: Size of the set of eligible protection levels versus capacity for \( n=3 \)

In another experiment, we fixed the capacity to 20 and changed the number of fare classes. Figure 4.2 shows the graph of \( N \) versus \( n \) where \( N \) denotes the size of the set of eligible protection levels and \( n \) is the number of fare classes. As it can be seen from figure 4.2, \( N \) grows exponentially with \( n \) as well.
Next experiment was to find out how the rules can help to reduce the size of the set of eligible protection levels. For sample 3 of table 4.6 and with 30 stages and a fixed capacity, we calculated the size of the eligible protection levels by applying no rule at all, rule 1 only, rules 1 and 2 and finally all three rules. We then repeated this experiment for different capacities.

Figure 4.3 shows the size of the set of eligible protection levels as a fraction of the original set when applying rule 1 for different capacities. Figure 4.4 shows the size of the set of eligible protection levels as a fraction of the original set when applying rules 1 and 2 for different capacities and finally figure 4.5 shows the same when applying rules 1,2 and 3 together.
Figure 4.3: The size of the set of eligible protection levels as a fraction of the original set when applying rule 1 for different capacities.

Figure 4.4: The size of the set of eligible protection levels as a fraction of the original set when applying rules 1 and 2 for different capacities.
4.1.3 PROPOSED METHOD FOR FINDING THE OPTIMAL PROTECTION LEVELS FOR CAPACITY CONTROL OF A SINGLE LEG FLIGHT UNDER THE L CONTROL METHOD

In this section we summarize the proposed method for finding the optimal protection levels for capacity control of a booking process under the L control method. We actually put together all the pieces we presented in sections 4.1.1 and 4.1.2 and summarize the necessary steps in our proposed method. As before, $c$ and $s$ denote the remaining capacity and remaining number of stages until the flight time.

The steps of our proposed method for capacity control of a single leg flight:

1. Set $S$, number of times that the policy is going to be updated

Figure 4.5: The size of the set of eligible protection levels as a fraction of the original set when applying all three rules
2. Develop the corresponding PDP model for the given $S$ and $C$

3. Solve the PDP model state by state
   
a. Set $s=1$ and $c=1$
   
   b. Use rules 1, 2 to develop the set of eligible protection levels: $\mathbb{S}_{s,c}$
   
   c. Use rule 3 to search over $\mathbb{S}_{s,c}$ for $PL$, which maximize the expected total revenue as calculated by formula 4.1
   
   d. Increase $c$ by one, if $c \leq C$, repeat steps “b” to “c’, else go to “e”
   
   e. Increase $s$ by one, if $s < S$, repeat steps “b” to “d”, else go to “f”
   
   f. If $s=S$, set $c$ equal to $C$ and repeat steps “b” and “c”.

4. Develop the final optimal tables for all the stages and every possible remaining capacity. Sample tables for stages 1, 2, $S-1$ and $S$ are presented below.

<table>
<thead>
<tr>
<th>Stage 1</th>
<th>Stage 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remaining Capacity</td>
<td>Optimal PL</td>
</tr>
<tr>
<td>1</td>
<td>$PL^*_1(1,1)$</td>
</tr>
<tr>
<td>2</td>
<td>$PL^*_1(1,2)$</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>$C$</td>
<td>$PL^*_1(1,C)$</td>
</tr>
</tbody>
</table>

Table 4.4: Optimal protection levels on stage 1 and 2 in a tabular format
Table 4.5: Optimal protection levels on stage $S-1$ and $S$ in a tabular format

In order to implement this solution one needs to set the protection levels to $PL^*(S,C)$ at the beginning of the booking period and then update the $PL$ at the beginning of each stage according to the remaining capacity by looking up the proper value from the corresponding table.

4.2 NUMERICAL EXPERIMENTS

In this section we present our numerical experiments on a single leg flight. The experiments were designed to compare the performance of our proposed method (PDP) with two other methods. Let us call the other two methods DLP and DP methods. In the DLP method the dual prices of the deterministic linear programming (DLP) model are used as the bid price values and the booking process is bid price controlled. As mentioned before, in the DLP model (section 2.2.2), the stochastic demand variable is replaced with one estimate only, its expected value, and therefore it does not fully capture the stochastic characteristic of demand. Nevertheless, it is very popular in practice. The other method which we call the DP method uses the original dynamic programming model (section 2.2.3) to find the corresponding optimal policy at the beginning of each and every
time unit throughout the booking process. Theoretically, the DP method is the best method for capacity control on a single leg flight because it fully captures the stochastic nature of demand, the nesting is incorporated into the optimization procedure in a natural way and finally because it is able to calculate the optimal policy for every possible combination of remaining capacity and remaining time until the flight time with taking into consideration the optimal policy that would be used in the future. Both the DLP and DP methods control the booking process by bid price control method. The difference between them is that they use different models and approaches to estimate the bid price values.

In order to compare the performance of the DLP and DP methods with our proposed capacity control method (PDP method), we will compare the expected total revenue that is generated under implementation of each method. The L Markov chain model was used to calculate the expected revenue by direct calculations. Note that if we could not calculate the expected total revenues by direct calculations, an alternative is to simulate the process of booking control under implementation of each method. Fortunately, we have found a way to model the booking process under implementation of DP and DLP methods by using our L Markov chain model. Therefore we can actually yield the expected total revenue by direct calculation rather than simulation. Another advantage of this alternative representation is that by using the L Markov chain model, we can actually calculate the expected load factor under implementation of each of these three methods and be able to yield consistent numerical result.
The remainder of this section is organized as follows. In section 4.2.1, we present how to use the L Markov chain model to represent the booking process under the bid price control method. Our design of experiment for numerical experiments on a single leg flight is presented in section 4.2.2. In 4.2.3, we compare the expected revenue versus rate of arrival under three methods. In 4.2.4, we compare the expected total revenue versus expected load factor for three methods. In 4.2.5, expected total revenue versus number of stags are compared for three different methods and finally in 4.2.6, we show the average over all 12 samples under three different methods.

4.2.1 USING THE NESTING TABLE AND “L MARKOV CHAIN” TO MODEL THE BOOKING PROCESS UNDER IMPLEMENTATION OF BID PRICE CONTROL

In section 3.4.1, we discussed how the L Markov chain can be used to represent the booking process under theft nesting control or equivalently the L control method for a given set of protection levels, PL. Note that our L Markov chain is also capable of modeling the booking process under bid price control for any given bid price value. In fact if the protection levels on each of the fare classes are either zero or equal to the remaining capacity, then applying the L method on the corresponding nesting table would be equivalent to bid price control of the booking process. The bid price value in this case would be less than or equal to the fare of the lowest value fare class whose protection level is
zero, but still greater than the next lowest fare. Figure 4.6 shows an example of such a nesting table.

Figure 4.6: A nesting table with a specific format of protection levels that could be used to model the booking process under bid price control with

\[ r_{i+1} < \text{Bid Price} \leq r_i. \]

Note that none or all of the seats are protected from each of the fare classes. In other words, at any time, some of the fare classes are open and some are closed. Applying the L control method on a nesting table with these specific protection levels indicates that as long as there is enough capacity, any request for the open fare classes will be accepted. On the other hand no request for closed fare classes will be accepted. Note that this concept is the same as bid price control of the booking process where the bid price value is greater than the fare of the first closed fare class (fare class “\(i+1\)” in figure 4.1) and smaller than the fare of the last open fare class (fare class “\(i\)” in figure 4.1). More specifically, let \(BP\) denote the bid price value where \(r(i+1) < BP \leq r(i)\). In other words, the fare of fare classes \(1, 2, \ldots, i\) are greater than \(BP\) and fares of fare classes \(i+1, i+2, \ldots, n\) are smaller than \(BP\). According to bid price control method (presented in section
1.1), any request for fare classes 1, 2, ..., i will be accepted as long as there is enough capacity remaining and any request for fare classes i+1, i+2, ..., n will be rejected. Therefore the L Markov chain that is developed for the nesting table of figure 4.1 represents both the booking process under the L control method with the given protection levels and the booking process under bid price control with BP where \( r(i+1) < BP \leq r(i) \).

Given a bid price value, \( BP \), we can find the equivalent protection levels, \( PL \), and represent the bid price control of the booking process by using the appropriate nesting table and the corresponding L Markov chain model. The advantage of representing the bid price control by using the L Markov chain model is that we can then calculate the expected total revenue by direct calculation rather than using simulation. For a given bid price value \( BP \), the corresponding protection levels could be derived as follows. We should close all the fare classes whose fares are smaller than \( BP \) and leave the rest of the fare classes open. Note that closing a fare class means setting its protection level equal to the remaining capacity and leaving a fare class open in this case is to set its protection level equal to zero. Suppose \( BP \) satisfies the following inequality: 
\[
r(n) < r(n-1) < \cdots < r(k) < BP < r(k-1) < \cdots < r(1),
\]
The resulting protection levels would then be equal to zero for fare classes 1, 2, ..., k-1 and equal to the remaining capacity for fare classes k, k+1, ..., n.

So, now we can use our proposed L Markov chain to model a booking process under bid price control for any given bid price value. Furthermore, we can use DLP or DP model to calculate the bid price values, develop the
corresponding L Markov chain and then calculate the “expected total revenue” under implementation of each method. Note that being able to calculate the expected total revenue by direct calculations from the corresponding Markov chain.

4.2.2 DESIGN OF EXPERIMENT

The goal of the numerical experiments presented in this section is to examine the effect of using the three different booking control methods on the expected total revenue. These three methods for capacity control of a single leg flight are: our proposed method (PDP), DP method and DLP method. Let us refer to our proposed method (presented in section 4.1.3) as PDP throughout this section. As explained in section 4.2.1, DP method is to use the original dynamic programming model (section 2.2.3) to calculate the optimal booking policy and bid price control the booking process on a single leg flight. DLP method is to use DLP model (section 2.2.2) to approximate the optimal booking policy and bid price control the booking process. The booking policies in both cases are bid price values.

The expected total revenue that is generated during the booking period depends not only on the control method of use but also on many other factors such as capacity of the airplane, length of the booking period, fares of the fare classes and rate of arrival of demand for each of the fare classes. In fact, the expected total revenue is a complicated function of all of these factors. No closed form formula is known to indicate how expected revenue is related to each of the
factors alone. Therefore it is impossible to separate the effect of the method of use from other factors. Therefore, in order to examine the effect of the control method as much independent as possible from other factors, we tried to look at the result for a wide range of $C$, $T$, $p$ and $r$’s. As mentioned before, $r$ is a vector of size $n$, which represents the fares of fare classes where $n$ denotes the total number of fare classes. $p$ is also a vector of size $n$ and $p(i)$ denotes the conditional probability that the request is for fare class $i$ given that there is a request in a time unit.

Note that in our numerical experiments on a single leg flight, we assumed that this probability is constant through out the booking period, however our model is capable of relaxing this assumption and changing the probability on different stages of the system. An example of such will be presented in section 5.2 where we present our numerical experiment on a network of flights.

Let $\rho$ represent the probability of arrival of a request of any kind in one time unit. So, $C$, $p$, $r$, $T$ and $\rho$ are the parameters that in addition to the control method, affect the expected total revenue.

One way to examine the effect of different methods (PDP, DLP and DP methods) on the expected total revenue independent of other parameters, one way is to make a very large set that contains many different combinations of the parameters and randomly select samples from the set. An alternative to this method is to intelligently select some samples from the set and run the experiment for those samples only. In this approach, the selected samples must be widely
dispersed over the whole set. In other words, the samples should be selected from different parts of the set.

We designed our experiment according to the second approach. In order to make sure that the samples are good representatives of the whole set, for a fixed capacity, number of fare classes and length of the booking period, we picked 4 different patterns for \( p \) and 3 different patterns for \( r \). Then we made 12 samples by combining these two factors. In order for these 12 samples to be a good representative of the whole set, we made sure that the patterns cover different parts of the set. More specifically, the four patterns of \( p \) are as follows, in the first one \( p(i) \)'s are increasing with \( i \), in the second pattern \( p(i) \)'s are decreasing with \( i \), in the third pattern, \( p(i) \)'s are increasing with \( i \) for the first half of the fare classes and decreasing on the second half, finally the last pattern is where \( p(i) \)'s are decreasing with \( i \) for the first half of the fare classes and increasing for the second half. It is the norm to number the fare classes in order of their fares such that fare class 1 is always the highest value fare class. Therefore, \( r(i) \)'s always decrease with \( i \). The three patterns that we picked for \( r \) are as follows: in the first pattern, \( r(i) \)'s decrease linearly as \( i \) increases, in the second pattern \( r(i) \)'s decrease with \( i \) but their values are closer to each other for the higher value fare classes and farther from each other for the lower value fare classes and finally the third pattern is to set \( r(i) \)'s so that they are closer to each other for the lower value fare classes and farther from each other for the higher value fare classes. The patterns for \( p \) and \( r \) are illustrated in the picture 4.7 and 4.8 respectively.
We then repeated this experiment for many different $C$: capacity, $n$: number of fare classes and $T$: length of the booking period. In all of the experiments, the resulting graphs had the same pattern, our proposed method worked better than DLP method and, as expected, worse than DP, however when updating the policy more than 3 times during the booking period, the performance of our proposed method was at most 0.5% worse than the DP method. We later examine the effect of more frequent updates.

The graphs of sections 4.2.3 to 4.2.6 show the results of applying different methods: PDP, DLP and DP to control the booking process on a single leg flight for 12 different samples. In all of the 12 samples $C=20$, $n=5$ and $T = 1200$. The
fares and conditional probability of arrivals of five fare classes in one time unit
for each of the samples are presented in table 4.6. Note that the probabilities are
conditioned on arrival of a request for any fare class in one time unit.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Pattern</th>
<th>Conditional Probability of Arrival of Fare Classes</th>
<th>Fares of Fare Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.11</td>
<td>0.11</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.11</td>
<td>0.11</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.11</td>
<td>0.11</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.33</td>
<td>0.22</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0.33</td>
<td>0.22</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0.33</td>
<td>0.22</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0.11</td>
<td>0.22</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0.11</td>
<td>0.22</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0.11</td>
<td>0.22</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.33</td>
<td>0.22</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>0.33</td>
<td>0.22</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0.33</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Table 4.6: 12 samples for the numerical result

4.2.3 COMPARING THE EXPECTED TOTAL REVENUE VERSUS RATE OF ARRIVAL UNDER THREE DIFFERENT METHODS

In this section we present some graphs that show the expected revenue that is generated during the booking period and under three different control methods:
PDP, DLP and DP where the policy is updated with the same frequency in the DLP and PDP methods and updated at the beginning of each time unit in the DP method. When the PDP method is used to control the booking process, the optimal protection levels at the beginning of each stage and for every possible remaining capacity are calculated from the PDP model as described in section 4.1.3. As mentioned then, when calculating the optimal protection levels at the beginning of each stage of the system, the optimal protection levels that will be used from the end of the stage until the flight time are also incorporated into the optimization procedure. Same thing is true for DP method. The optimal bid price value at the beginning of each stage is with cooperating optimal policies of future stage unto consideration. However, on DLP method which is the common practice the updates of the optimal protection levels are done using the telescope approach in which the optimal protection levels are calculated at the beginning of each stage according to the remaining capacity and the remaining time until the flight time without taking into consideration the updates of the optimal policy which will happen from the end of the stage until the flight time. The calculated policy is then kept for one stage and at the beginning of the next stage this process is repeated to find the corresponding optimal policy.

Figures 4.9 to 4.14 show the result of applying three different single leg capacity control methods (PDP, DLP and DP) to some of the samples of table 4.6. Graphs of this section show the expected revenue versus N where N denotes the expected number of arrivals during the booking period. In our numerous experiments we realized that when the expected number of requests during the
booking period is less than 80% of the capacity, the resulting expected total revenue is not affected by the method used. This makes sense because when the expected number of requests during the booking period is less than the capacity, in order to prevent from flying empty, any reasonable booking policy would accept all the requests that arrive during the booking period. On the other hand, if the expected total number of requests for higher value fare classes is too large, any reasonable booking control method would protect the seats from lower value fare classes and reserve them for higher value fare classes. In our numerical experiments we learned that when the expected number of requests during the booking period is on the range of 80% to 130% of the initial capacity of the cabin, the effect of applying different control methods on the expected total revenue is the highest. In other words, in this range we can better see how the control method of use can affect the expected total revenue that is generated during the booking period. Therefore in all of the graphs, we show the result in this range only.

In order to stay consistent with the Poisson arrival assumption we made sure that the probability of receiving two or more requests in one time unit is less than 0.00044 which in turn makes the arrival rate to be at most 0.03. We then found the length of the booking period according to the arrival rate and fixed capacity for each experiment. More specifically, for \( C=20 \) we fixed the length of booking period to 1200 so that the resulting arrival rates would make the expected total number of requests during the booking period less than 180% of the initial capacity. In our numerical experiments we first generated 100 random values for
Each value of $N$ corresponds to a specific arrival rate which can be found from this formula: $ho = \frac{N}{C \times T}$ where $\rho$ and $T$ denote the arrival rate and the length of the booking period respectively. For each arrival rate, we then calculated the expected revenue that is generated under implementation of each of the three methods. In order to calculate the expected total revenue, for DP and DLP methods, we first used the corresponding models (DP and DLP models) to calculate the optimal booking policies as often as required (at the beginning of each stage) and then used our L Markov chain to model the booking process under implementation of the corresponding booking policy.

Figure 4.9: Expected revenue under implementation of DLP, PDP and DP methods for sample 1 of table 4.6 where the optimal policy is updated 3 times.
Figure 4.10: Expected revenue under implementation of DLP, PDP and DP methods for sample 1 of table 4.6 where the optimal policy is updated 12 times.

Figure 4.11: Expected revenue under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 6 times.
Figure 4.12: Percentage difference in expected revenue with the DP method under implementation of DLP and PDP methods for sample 3 of table 4.6 where the optimal policy is updated 6 times.

Figure 4.13: Expected revenue under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 60 times.
As it can be seen in figures 4.9, 4.10, 4.11 and 4.13, regardless of $p$, $r$ and the number of times that the policy is updated, the graphs have similar characteristics. First: expected revenue seems to be a concave non-increasing function of $N$ and so of the arrival rate. Second: the DP method generates the highest expected revenue where the DLP method generates the lowest expected revenue. Third: when the policy is updated at least once every 60 time units (more than 3 times in this case), the expected total revenue that is generated under the PDP method is at most 0.5 % worse than the expected revenue that is generated under the DLP method.

This experiment was repeated for many different values of $C$, $n$ and $T$ and for different samples that were created similar to the ones presented in table 4.6.
The three characteristics of the resulting graphs that we discussed in the previous paragraph were observed in all of the experiments. It is easy to see why the expected total revenue is a non-decreasing function of $N$. As $N$ increase, the arrival rate, $\rho$, would also increase which in turn increases the expected total number of requests during the booking period. It is easy to see that under any control policy, larger rate of arrivals could never result in lower expected revenue. Although, control parameters like bid price values and protection levels might change as $\rho$ increases. As $\rho$ increases, the rate of arrival of all of the fare classes including the higher fare classes increases. At some point when the arrival rate gets large enough, the optimal policy changes to a more selective one. In other words, the bid price values and protection levels on the lower value fare classes increase. In all three methods, the objective function of the optimization model is the expected revenue that is generated. Therefore if the optimal booking policy for a specific arrival rate is more selective than the one for a lower arrival rate, it generates larger expected revenue. In summary, regardless of whether or not the policy has improved, the expected total revenue increase with $\rho$.

4.2.4 COMPARING THE EXPECTED TOTAL REVENUE VERSUS EXPECTED LOAD FACTOR UNDER THREE DIFFERENT CONTROL METHODS

The graphs that are presented in this section show the expected total revenue versus expected load factor when PDP, DP and DLP methods are used to control the booking process during the booking period. The load factor is the
fraction of the capacity that will be filled at the flight time. The actual value of the load factor would depend on the number of requests that will be accepted during the booking period. Since we do not use simulation in our experiments, we cannot keep track of the actual number of accepted requests for simulation samples. However, there is a way to calculate the expected total revenue by using the L Markov chain model and the corresponding transition probability matrix. Calculating the expected load factor from the L Markov chain model is presented in section 4.2.4.1.

4.2.4.1 CALCULATING THE EXPECTED LOAD FACTOR BY USING THE L MARKOV CHAIN MODEL

Let $AD_i(x)$ represent the expected number of requests that are accepted from the beginning of stage $i$ until the flight time where $x$ denotes the remaining capacity at the beginning of stage $i$. We are interested in finding the value of $AD_S(C)$ which is the expected number of accepted requests during the entire booking period, from the beginning of stage $S$ until the flight time where the initial remaining capacity is $C$. The expected load factor would then be equal to $AD_S(C)/C$. Note that the optimal protection levels might change at the beginning of each stage. As long as we know the optimal policy on each stage, we can develop the corresponding L Markov chain and use its transition probability matrix to find the expected remaining capacity at the end of the stage. In order to find $AD_S(C)$ we should start the calculation from stage 1.
We first need to use the transition probability matrix of the L Markov chain model of stage 1 to find the expected remaining capacity at the end of stage 1 which is actually the flight time. We do not know what the remaining capacity would be at the beginning of stage 1 until we actually reach that point in time. Therefore we need to find the expected remaining capacity at the flight time for any possible remaining capacity at the beginning of stage 1. The remaining capacity at the beginning of stage 1 might be anywhere from 0 to $C$. So, we need to find $AD_1(0), AD_1(1), \ldots, AD_1(C)$ by using the following formula.

$$
AD_1(x) = \left( \left\{ P_{PL,x}^T \right\}_{1,1} \times 0 \right) + \left( \left\{ P_{PL,x}^T \right\}_{1,2} \times 1 \right) + \cdots + \left( \left\{ P_{PL,x}^T \right\}_{1,x} \times (x-1) \right) + \left( \left\{ P_{PL,x}^T \right\}_{1,x+1} \times x \right)
$$

(4.7)

Next we need to find $AD_2(0), AD_2(1), \ldots, AD_2(C)$, the expected number of accepted requests from the beginning of stage 2 until the flight time. Again, the optimal policy during stage 2 is known and so is the transition probability matrix of the corresponding L Markov chain. Therefore the state of the system at the end of stage 2 can be found. $AD_2(x)$ can be calculated as follows.

$$
AD_2(x) = \left\{ P_{PL,x}^T \right\}_{1,1} \times (0 + AD_1(x)) + \left\{ P_{PL,x}^T \right\}_{1,2} \times (1 + AD_1(x-1)) + \cdots + \left\{ P_{PL,x}^T \right\}_{1,x} \times (x - 1 + AD_1(1)) + \left\{ P_{PL,x}^T \right\}_{1,x+1} \times (x + AD_1(0))
$$

(4.8)
Note that $\{p_{PL, x}^{L} 1_{i, i}\}$ shows the probability that we end up at the $i$-th state of the L Markov chain model at the end of stage 2. $i$-th state of the L Markov chain model is where there are $x-i+1$ seats remaining which in turn indicates that $i-1$ requests were accepted during stage 2. We shall repeat this calculations in a similar manner for stage 3, 4, ..., $S-1$. Since the initial capacity is $C$, at the beginning of stage $S$, we just need to calculate the value of $AD_{S}(C)$ which is the expected total number of requests that are accepted from the beginning of the booking process until the flight time.

4.2.4.2 GRAPHS OF EXPECTED REVENUE VERSUS EXPECTED LOAD FACTOR

The graphs in this section show the expected revenue versus expected load factor under implementation of PDP, DLP and DP methods and for some of the samples of table 4.6. Expected load factors were calculated as explained in section 4.2.4.1. In order to make each of the graphs, we first generated 150 random values for $N$ in the range of 0.8-1.3. For each $N$ we calculated the expected revenue and expected load factor for each of the three methods by using the corresponding L Markov chain model.
Figure 4.15: Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated once.

Figure 4.16: Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 3 times.
Figure 4.17: Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 6 times.

Figure 4.18: Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 12 times.
Figure 4.19: Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 30 times.

Figure 4.20: Expected revenue versus expected load factor under implementation of DLP, PDP and DP methods for sample 3 of table 4.6 where the optimal policy is updated 60 times.
As it can be seen in figures 4.15 to 4.18, some values of the load factors correspond to more than one value of expected revenue. Note that in this experiment, arrival rate is the independent variable and its value is not affected by the policy or method of use. Furthermore, note that load factor is the expected number of accepted requests during the booking period divided by the capacity. The booking policy that is used to control the booking process directly affects the expected number of accepted requests and so the expected load factor. Unlike arrival rate, expected load factor is directly affected by the booking policy of use. On the other hand, the optimal booking policy itself depends in the arrival rate. The larger the arrival rate gets, the more selective the optimal policy would be. Load factor however does not have any effect on the policy and in fact it does not even exist before the policy is defined. Load factor can only be calculated after the policy is determined. What causes the pattern in these graphs can be explained as follows.

Let us look at a specific case here, PDP method in figure 4.10 where the optimal policy is updated only once at the beginning of the booking period. The optimal policy (set of protection levels in PDP method) depends on many factors such as capacity, fares, number of fare classes, length of the booking period and arrival rate. Note that the optimal set of protection levels is a step function of arrival rate. In other words, for some range of arrival rate, the optimal set of protection levels remains the same. Once the arrival rate gets large enough, the protection levels change to a more restricted one, which actually protects more seats from the lower value fare classes. The reason is that as the arrival rate
increases, the expected number of requests for all the fare classes and in particular for higher value fare classes increase therefore the optimal policy would change to a new one in such a way that it gets more selective.

Let $\rho_1$ and $\rho_2$ represent two different arrival rates and let $\rho_1 > \rho_2$. Let $PL_1^*$ and $PL_2^*$ denote optimal protection levels corresponding to $\rho_1$ and $\rho_2$ respectively. Furthermore, suppose that the difference between the two arrival rates is large enough so that these two optimal policies, $PL_1^*$ and $PL_2^*$ are different. Obviously, the optimal policy corresponding to $\rho_1$ is more selective and protects more seats from the lower value fare classes. Although the expected revenue that is generated under $PL_1^*$ is larger than the one generated under $PL_2^*$, since $PL_1^*$ is more selective, the number of requests that are accepted under this policy could be expected to be less than or equal to the number of requests that is accepted under $PL_2^*$. Therefore, these two different set of protection levels may actually result in the same expected load factor and different expected revenues. Policy 1 is designed to control a booking process with larger demand. Policy 2 on the other hand controls the booking process where there are fewer requests, but since policy 1 is more selective than policy 2 and rejects more requests than policy 2, the number of requests that were accepted in each case may end up to be the same. The expected revenue under policy 1 would be larger since the accepted requests are more of the higher value fare classes.

This pattern gradually smoothes away as the optimal policy is updated more often. In fact, as we move from graph 4.15 all the way to graph 4.20 where the policy is updated 60 times, we can see that the graphs get smoother and
smoother and finally when the optimal policy is updated frequent enough, corresponding to each load factor there is only one expected revenue. In fact this effect does not happen when DP method is applied, simply because the best policy is updated at the beginning of each and every time unit. This experiment was repeated for a wide range of $C$, $n$, $r$, $p$ and $T$ and many different samples. The same pattern was observed in all of them.

4.2.5 EXPECTED TOTAL REVENUE VERSUS NUMBER OF STAGES

In this section we present a graph, which shows how increasing number of stages can improve the resulting expected revenue. Number of stages indicates how many times the optimal policy is updated throughout the entire booking period. Obviously, the more we update the optimal policy the larger the generated revenue would be. Under implementation of DP method, the policy is updated at the beginning of each and every time unit. As seen before, the expected revenue that is generated under this method is the largest among the three methods. The trade off is of course the computational difficulty that is involved in the updating process. In this experiment we were interested to see how increasing the number of stages could decrease absolute deviation from expected revenue of DP method. In order to obtain the graph of figure 4.16, we first calculated the mean absolute percent deviation in expected revenue of DLP and PDP methods from expected revenue of DP method for different values of $S$ where $S$ represents the number of stages during the booking period. The average was taken over 150 different arrival rates and 12 samples of table 4.6.
Figure 4.21: Average percentage difference in the expected revenue with DP method where the average is taken over all the 12 samples and different arrival rates.

As it can be seen from figure 4.21, increasing number of stages is more effective when applying PDP method rather than DLP method. In other words, the percentage difference from the ideal expected revenue decreases faster under implementation of PDP method. Also as it can be seen from figure 4.21, when we apply our proposed method (PDP) and update the policy more than 3 times which in this case is after 400 time units the resulting average revenue is within 0.5% of the expected revenue that is generated under DP method. In other words, with much less computational difficulty we can get almost the same result as the DP method.
4.2.6 AVERAGE OVER ALL 12 SAMPLES

In this experiment, we first generated 100 random values for arrival rates on the range as described in section 4.2.2. For each arrival rate, we calculated the expected revenue that was generated under implementation of each of the three methods for each of the 12 samples and for a given number of stages. We then repeated this process for different stage numbers on the range of 1 to 120. Figure 4.22 shows the average revenue that is generated under implementation of each of the three methods versus the number of stages where the average is taken over all 12 samples and different arrival rates.

Figure 4.22: Average revenue under implementation of each of the three methods versus the arrival rate where, the average is taken over all 12 samples and different stage numbers.
As it can be seen on figures 4.22, when we apply our proposed method (PDP) and update the policy more than 3 times which in this case is after 400 time units the resulting average revenue is within %0.5 of the expected revenue that is generated under DP method.

4.3 SUMMARY AND CONCLUSION

In this section we presented our proposed method (PDP method) for capacity control of a single leg flight. In PDP method we use an approximate dynamic programming model with $S$ stages to model the booking process under theft nesting control method where the optimal protection levels are updated $S$ times during the booking process. We then solve this proposed dynamic programming model exactly. This approach is an alternative to the DP method in which the booking process is modeled as an exact dynamic programming model where the policy is updated at the beginning of each and every time unit in the booking period. The DP method involves a great computational effort which makes it impractical to extend to a network of flights. However our PDP method is easily extendable to a network of flights with affordable computational effort. We also presented some numerical experiments in this section which showed PDP method generates higher expected revenue compared to DLP (a very common practice in single leg capacity control). The numerical result also showed that when the policy is updated frequent enough but not as frequent as in the DP method, the resulting expected revenue is within 0.5% of the expected revenue
generated under DP method. In the next section we show how to extend our proposed single leg capacity control to a network of flights.
CHAPTER 5
CAPACITY CONTROL ON A NETWORK OF FLIGHTS

In general, there are two approaches for capacity control of a network of flights. The first approach is to solve the network as a whole and the second approach is to first decompose the prices of the multi-leg itineraries into individual legs and then control the booking process on the network by controlling the booking process on individual legs simultaneously. In other words, a request for a multi leg itinerary is accepted only if it could be accepted on all the legs that are used by the itinerary. For the first approach, the most common practice is to use the dual prices of the capacity constraints of the DLP model and bid-price control the network. In the second approach, different methodologies exist where the main difference between them are: (i) how to decompose the price of the multi-leg itineraries into single legs and (ii) which single-leg capacity control method to use to control the booking process on different legs. The process of decomposing the prices of the multi leg itineraries into single legs is called prorating. Different prorating methods exist in the literature, prorating based on mileage, number of legs and on the ratio of the local
fare levels. Decomposed price on the single leg should actually represent the
collection of that part of the itinerary to the whole network.

Our proposed method for capacity control of a network of flights falls
under the second category. We first decompose the prices of the multi leg
itineraries into individual legs and then control the network by applying our
proposed single leg capacity control (section 4.1) on individual legs of the
network. We have come up with a decomposition method which does not exist in
the literature. In order to compare the performance of our proposed network
capacity control with the other methods that are practiced, we have performed
numerical experiments which we will present in section 5.2.

The remainder of this chapter is organized as follows. In section 5.1 we
discuss the most common prorating method and then present our proposed
prorating method. We will summarize the steps of our proposed method for
network capacity control in section 5.1.2. In section 5.2, we present our
numerical experiments that were done to compare the performance of our
proposed network capacity control with two other methods, where one of them is
the common practice and the other is a combination of common prorating practice
with our proposed single leg capacity control. We will then present a summary of
the chapter in section 5.3.
5.1 SOLVING THE NETWORK BY DECOMPOSING IT INTO INDIVIDUAL SINGLE LEG PROBLEMS

As mentioned before, one approach for solving a network of flights is to first decompose the network problem into multiple single leg problems and then control the booking process by simultaneously controlling the booking process on single legs. In this approach, a request for a multi leg itinerary is accepted only if it could be accepted on all the legs that are used by the itinerary. The network is decomposed to single legs through a process called prorating in which the prices of the multi leg itineraries are decomposed into the single legs. The decomposed price must be a good representative of the contribution of that part of the itinerary to the network. Williamson (1992) investigated different prorating methods such as prorating based on mileage, number of legs and on the ratio of the local fare levels. One of the most common prorating practices is called displacement-adjusted revenue prorating. In the remainder of this section we first present displacement adjusted revenue prorating in section 5.1.1 and then present our proposed prorating method in section 5.1.2.

5.1.1 DISPLACEMENT-ADJUSTED REVENUE PRORATING (DARP)

The most common prorating practice is to use the dual prices of the capacity constraints of DLP model (presented in section 2.2.2) as follows:
Table 5.1: Table of notation for DARP prorating method

\[
\bar{r}_l^{ODF} = r_{ODF} - \sum_{k \in \{L_{ODF} - l\}} d_k
\]

In equation 5.1, \( d_k \) represents the opportunity cost of reducing the capacity of leg \( k \) by one unit, so that \( \sum_{k \in \{L_{ODF} - l\}} d_k \) estimates the opportunity cost of reducing capacities of all legs except leg \( l \) of the \( ODF \) by one unit. This value is then reduced from the fare of the \( ODF \). The reduced original fare of an \( ODF \) called displacement-adjusted revenue in this context is intended to approximate the net benefit of accepting itinerary \( ODF \) on leg \( l \).

5.1.2 PROPOSED PRORATING METHOD: REMAINING CAPACITY PRICING (RCP)

Our proposed prorating method is interconnected with our proposed search method for optimal protection levels on different stages. That is because we need the optimal marginal revenues at each stage to determine the decomposed prices of the multi-leg itineraries at the next stage. Suppose that we want to decompose the fare of the multi-leg itinerary \( ODF \) on its corresponding
legs at the beginning of stage $s$. We use the same notation as defined above except with an additional superscript to indicate stage:

| Let $r_{ODF}^s$ = & The original revenue (fare) for itinerary $ODF$, at stage $s$. |
| --- & --- |
| $\bar{p}_{l,ODF}^s$ = & Decomposed price of $ODF$ allocated to leg $l$, at stage $s$. |
| $L_{ODF}$ = & Set of legs used by $ODF$, which do not change with stage. |
| $LB_{l,ODF}^s$ = & Lower bound on protection levels for $ODF$, at stage $s$ of leg $l$. |
| $E(c_l^i)$ = & Expected remaining capacity of the cabin at stage $s$ of leg $l$. |

**Table 5.2: Table of notation for RCP prorating method**

As discussed in section 4.1.2.2, rule 2 uses marginal revenues at stage $s-1$ and the fare of a fare class at stage $s$ to come up with the lower bound on the protection level of that specific fare class at stage $s$. Similar to what we discussed for a single-leg case in section 4.1.2.2, rule 2 uses $\bar{p}_{l,ODF}^s$ as well as the marginal revenues on leg $l$ at stage $s-1$ to come up with the lower bound on protection level of $ODF$ on stage $s$. By definition $LB_{l,ODF}^s$ is the minimum number of seats that should be protected from the multi-leg itinerary $ODF$ on leg $l$ at stage $s$, but it is currently unknown. $LB_{l,ODF}^s$ is calculated by rule 1 as described in section 4.1.2.1, but the calculation depends on $\bar{p}_{l,ODF}^s$, which in turn depends on how $r_{ODF}$ is decomposed into the corresponding leg.

There are many ways in which $r_{ODF}$ can be partitioned into the legs of set $L_{ODF}$ at stage $s$. Our idea is to decompose $r_{ODF}^s$ in such a way that
is approximately equal, at stage \( s \), for all \( l \in L_{ODF} \). In other words, at stage \( s \), \( r_{ODF} \) should be partitioned into the legs of set \( L_{ODF} \) in such a way that roughly the same proportion of the expected remaining seats in class \( ODF \) have prices on their legs that are greater than the marginal value of a seat as subsequent stages. In this decomposition method, the number of seats that are not protected from ODF (according to rule 1) on each leg will be relative to the expected remaining capacities of the legs at the time. Because this method needs the marginal values of stage \( s-1 \) to decompose the prices at stage \( s \), therefore it is not possible to decompose the prices at stage 1. Our idea is to use the displacement-adjusted revenue method (section 5.1.1) to decompose the prices at stage 1, and our proposed method thereafter.

As we discussed above, in order to calculate the decomposed prices at stage \( s \) by our proposed method, we need to know the expected remaining capacities, \( E(c_i^s) \), of the legs at the beginning of stage \( s \). The \( E(c_i^s) \) of the legs at the beginning of stage \( s \) which is the end of stage \( s+1 \), would depend on the protection levels that were implemented from the beginning of the booking period until the end of stage \( s+1 \). However, since the PDP model is solved backward starting from the first stage, the optimal protection levels on stage \( s+1, s+2, \ldots, S \) would not be known until we solve stage \( s \) first. So, stage \( s \) can not be solved until we know how to decompose the prices at that stage. The prices can not be decomposed until we know the remaining capacities of the legs at the beginning of the stage. Finally these optimal protection levels would not be known until we
actually solve sage \( s \). In summary, we need the solution of stage \( s \) to be able to solve stage \( s \)!

In order to deal with this contradiction, we propose an iterative approach. At the first iteration, we use the initial capacities of the legs at the beginning of each and every stage and apply our proposed decomposition method along with our proposed search method on all the single legs to find the optimal protection levels at each stage on each leg. Then we use this solution of the first iteration to estimate the expected remaining capacity of the legs at the beginning of each stage. We then again apply our proposed decomposition method along with our search method on all the legs by using the estimated remaining capacities of the legs. Since the remaining capacities that are used on the second iteration are different than the ones that were used on the first one (initial capacities) the decomposed prices on each stage would be different and so will be the optimal protection levels at each stage on each leg. We shall repeat this procedure until the optimal solution and decomposed prices converge. In our experiments, the solution usually converged in less than 8 iterations.

5.1.3 PROPOSED METHOD FOR CAPACITY CONTROL OF A NETWORK OF FLIGHTS

In this section we put together every thing that we have discussed so far and summarize our proposed method for capacity control of a network of flights. Our proposed method for capacity control on a network of flights is as follows,
1- Set $S$, number of times that we want to update the policy

2- Use iterative RCP method along with our proposed dynamic programming method (PDP) (section 4.1.3) to find the optimal protection levels at each stage and on each leg of the network.

For each leg of the network, the solution is summarized in tables like table 4.5 and 4.6. In order to implement this solution in practice, one needs to first follow the steps mentioned above to find the optimal table on each leg of the network. Then at the beginning of each stage, look up the proper protection levels on each leg for the corresponding stage and remaining capacity of the leg. Protection levels on the legs should then be set to these optimal values and kept for one stage. The booking process is then controlled by the L control method (or equivalently, theft nesting control method) on all the legs of the network simultaneously. At the beginning of next stage, the optimal protection levels should be updated according to the stage and the remaining capacities of the legs at that time. This shall be repeated until the very last stage which is in fact the stage before the flight time.

5.2 NUMERICAL EXPERIMENTS ON NETWORK

In this section we present our numerical experiments on the network. The goal is to compare the performance and efficiency of three capacity control methods. We are interested in comparing the performance of our proposed capacity control method with two other methods. We refer to our proposed capacity control methods as RCP-PDP and call the other two methods DARP-
PDP and NET-BP. RCP-PDP is our proposed method for capacity control of a network of flights in which we use our iterative RCP prorating method to decompose the prices of the multi leg itineraries into individual single legs and use our proposed single leg capacity control method (PDP) to find the optimal protection levels at the beginning of each stage and on all the legs of the network. We then control the booking process on the network by using these optimal protection levels and by applying the L control method (Theft nesting control method) on individual legs simultaneously. DARP-PDP method uses DARP prorating method (section 5.2.1.2) which is a common prorating method and then uses our proposed search method to find the optimal protection levels at the beginning of each stage for the corresponding remaining capacity on all the legs. The network is controlled by applying these optimal protection levels and by implementing the L control method on individual legs. NET-BP which is a common practice in network capacity control solves the network as a whole and without decomposing it to individual legs. It uses dual prices of the capacity constraints of the DLP model (section 2.2.2) as bid price values and controls the booking process on the network by using bid price control method (section 1.1).

In order to compare the performance of the three methods: RCP-PDP, DARP-PDP and NET-BP, we have simulated the booking process on three networks of different shapes and sizes under implementation of these three control methods. In the simulation process for each of the networks, we generated 5000 random vectors of requests. Each vector was generated according to Poisson arrival assumption and represented a sample string of requests during
the booking period. We then applied three control methods, RCP-PDP, DARP-PDP and NET-BP to control the booking process for each sample of requests. This procedure was then repeated for all 5000 samples and we took the average revenue over all 5000 samples for each of the three methods. By comparing these averages we can then see how well the three control methods work compared to each other.

In the remainder of this section we first look at the three control methods and the simulation process in detail in section 5.2.1. We will then present our numerical experiments in section 5.2.2.

5.2.1 NETWORK METHODS

In this section we look at the three network control methods, RCP-PDP, DARP-PDP and NET-BP in detail. We also discuss the simulation process for each case in detail.

5.2.1.1 RCP-PDP NETWORK CONTROL METHOD

RCP-PDP is our proposed method for network capacity control. As discussed in section 5.1.3, in our proposed method we first use our proposed method of prorating (RCP) to decompose the network problem into multiple single leg problems and then control the booking process on the network by controlling the booking process on individual legs by using our proposed single leg capacity control method (PDP). The abbreviation RCP-PDP indicates that RCP method is used for decomposing the prices of the multi leg itineraries and
PDP method is used to find the optimal protection levels on individual legs. The booking process on each of the legs is controlled by implementing the L control method. In the simulation, booking control is almost independent by leg except that when a leg experiences a multi-leg request all legs are checked for available seats before the request is accepted.

In order to simulate the booking process under implementation of this control method, we first used our proposed method for finding the optimal protection levels on all the legs of the networks, at each stage and for any possible remaining capacity of the legs as described in section 5.1.3. After that we simulated the booking process by using these calculated optimal protection levels. The simulation process was as follows. We first generated 5000 samples of string of requests where each sample represented requests that would arrive during the entire booking period. For each of these samples then, we simulated the booking process from the beginning of stage $S$ until the flight time, where at the beginning of each stage the protection levels were updated according to the stage and the remaining capacities of the legs at that time. At then end of the booking period we calculated the total revenue that was generated during the booking period for that sample. This process was then repeated for all 5000 samples and we took the average and the sample standard deviation.

5.2.1.2 DARP-PDP NETWORK CONTROL METHOD

DARP-PDP method uses displacement adjusted revenue prorating (section 5.1.1) to decompose the prices of the multi leg itineraries into individual single
legs and uses our proposed dynamic programming method to find the optimal protection levels on each leg for the remaining time and capacity at the beginning of each stage. Note that both decomposed prices and the optimal protection levels depend on the remaining capacity of the legs and remaining time until the flight time. More specifically, right hand sides of demand constraints of the DLP model (section 2.2.2) directly depend on the remaining time until take off. Also, the right hand sides of capacity constraints of DLP model directly depend on the remaining capacities of the legs. Therefore any change in the remaining time and the remaining capacities would change the optimal solution of the DLP model and in particular the shadow prices of the capacity constraints. As mentioned in section 5.1.1, shadow prices of the capacity constraints are used to calculate the decomposed prices. Optimal protection levels calculated by our proposed search method would also depend on the remaining time and capacities of the legs as well as the decomposed prices.

In practice, this telescope method of update is used where at the beginning of each stage the optimal booking is calculated for the remaining capacities of the legs at the time and the remaining time until the flight time. This booking policy is then kept fixed for one stage and at the beginning of next stage again, the calculations for finding the optimal booking policy is repeated. In our experiment we actually simulated the telescope approach. More specifically, for each sample request, we first used the DLP model to decompose the prices of the multi leg itineraries into individual legs and then used our proposed dynamic programming method (section 4.1.3) to find the optimal protection levels on each leg. We then
simulated the booking process for one stage where the optimal protection levels were kept fixed throughout the stage. At the beginning of the next stage again, we used DLP model to decompose the prices of the multi leg itineraries into individual legs according to the stage and the remaining capacities of the legs at the time and then used our proposed search method to find the optimal protection levels for each leg. Then the booking process was again simulated for another stage where the protection levels were kept fixed throughout the stage. This was repeated until the end of the booking period. We then calculated the total revenue that was generated during the booking period for each sample. This procedure was then repeated for each of the 5000 samples and we calculated the average revenue over all the 5000 samples to represent the average revenue under implementation of DARP-PDP control method. We also calculated the sample standard deviation.

5.2.1.3 NET-BP CONTROL METHOD

Unlike RCP-PDP and DARP-PDP methods, NET-BP method does not decompose the network into single leg problems and instead solves the network problem as a whole. Like DARP-PDP method, NET-BP also uses a telescope approach in which the optimal booking policy is calculated at the beginning of each stage for the remaining capacities of legs at the time and remaining time until the flight time. NET-BP method uses dual prices of DLP model (section 2.2.2) as bid price values. Note that there is one dual price corresponding to each of the legs of the network. Bid price value corresponding to each multi-leg
itinerary is derived by adding the dual prices of the legs that are used by that itinerary.

In our numerical experiments we simulated the telescoping method of updating as suggested in the literature. We first generated 5000 samples of requests where each sample included a vector of requests and represented requests during the booking period. For each sample, we first used the DLP model to approximate the optimal bid price values for all the itineraries and for the corresponding remaining time and capacity. We then simulated the booking process for one stage where the bid price values were used to control the booking process on the network. At the beginning of the next stage again, we used DLP model to calculate the bid price values for the updated remaining capacities of the legs and remaining time until the flight time. Then the booking process was again simulated for another stage where the booking policy was kept fixed throughout the stage. This was repeated until the end of the booking period. We then calculated the total revenue that was generated during the booking period for each sample. This whole procedure was then repeated for each of the 5000 samples and we calculated the average revenue over all the 5000 samples to represent the average revenue under implementation of NET-BP control method.

5.2.2 NUMERICAL EXPERIMENTS

In this section we present our numerical experiments on network. We applied the three control methods: RCP-PDP, DARP-PDP and NET-BP to two different networks of different shapes and sizes. Our goal in the experiments was
to compare the performance of the three methods. In each case we were interested to compare the average revenue that was generated under implementation of each control method as well as the computational effort that was required to calculate/estimate the optimal booking policy.

Note that in order to calculate the average load factor we first calculated the actual load factor for each of the sample string of requests first and then took the average over all the actual load factors. In order to find the actual load factor for each sample, we used the following formula, \[ \frac{O_1 + O_2 + \cdots + O_L}{C_1 + C_2 + \cdots + C_L} \]. In this formula \( O_i \) denotes the number of occupied seats on leg \( i \) at the end of the booking period, \( C_i \) represents the initial capacity of leg \( i \) and \( L \) denotes the number of legs in the network. The numerator of the fraction is total number of seats that are taken at then end of the booking.

In the remainder of this section, we present our numerical experiments on three networks in three different sections. Our numerical experiment on network 1 and two different experiments on network 2 are presented in sections 5.2.2.1, 5.2.2.2 and 5.2.2.3 respectively.

5.2.2.1 EXPERIMENT 1

The numerical experiments in this section were performed on a network which we will refer to as network 1. Network 1 consists of 4 cities, 3 single legs and 10 ODFs. Figure 5.1 shows network 1, where A, B, C and D represent the three cities. The three single leg flights are: AB, BC and CD which go from A to B, B to C and C to D respectively. There are total of 5 itineraries each with two
different fares which makes total of 10 ODF’s. In figure 5.1, the ODF numbers are labeled above the arrows which show the itineraries. ODF 1 to 6 are single leg flights where ODF 7,8,9 and 10 are two leg flights. ODF 7 and 8 go from A to C thorough B and ODF 9 and 10 go from B to D through C.

![Figure 5.1: Network 1](image)

Table 5.1 shows the capacity of each leg and table 5.2 shows the fares, conditional arrival probabilities and the itineraries of the ODF’s as well as the legs that are used by each ODF. Note that the conditional probability of arrival of an ODF is the probability of arrival of a request for an ODF in one time unit given that there is an arrival of a request for any of the existing ODF’s.

<table>
<thead>
<tr>
<th>Leg</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>10</td>
</tr>
<tr>
<td>BC</td>
<td>15</td>
</tr>
<tr>
<td>CD</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 5.3: Capacity of the legs of network 1
We set the length of a time unit equal to 5 minutes and the length of the booking period equal to 30 days or 8640 time units. In order to stay consistent with the Poisson assumption we always set the arrival rate less than 0.03. Note that by arrival rate we mean probability of receiving a request for any of the ODF’s in one time unit. If rate of arrival is less than 0.03, then the probability of receiving more than one arrival in one time unit would be at most 0.0004411 which is small enough to be negligible.

Figures 5.2, 5.3 and 5.4 show the average revenue that is generated during the entire booking period under implementation of the three different control methods, NET-BP, RCP-PDP and DARP-PDP.

<table>
<thead>
<tr>
<th>ODF</th>
<th>Itinerary</th>
<th>Legs</th>
<th>Fare</th>
<th>Conditional Probability of Arrivals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A-B</td>
<td>AB</td>
<td>100</td>
<td>0.05</td>
</tr>
<tr>
<td>2</td>
<td>A-B</td>
<td>AB</td>
<td>75</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>B-C</td>
<td>BC</td>
<td>220</td>
<td>0.1</td>
</tr>
<tr>
<td>4</td>
<td>B-C</td>
<td>BC</td>
<td>180</td>
<td>0.15</td>
</tr>
<tr>
<td>5</td>
<td>C-D</td>
<td>CD</td>
<td>150</td>
<td>0.15</td>
</tr>
<tr>
<td>6</td>
<td>C-D</td>
<td>CD</td>
<td>210</td>
<td>0.1</td>
</tr>
<tr>
<td>7</td>
<td>A-B-C</td>
<td>AB and BC</td>
<td>250</td>
<td>0.15</td>
</tr>
<tr>
<td>8</td>
<td>A-B-C</td>
<td>AB and BC</td>
<td>310</td>
<td>0.1</td>
</tr>
<tr>
<td>9</td>
<td>B-C-D</td>
<td>BC and CD</td>
<td>350</td>
<td>0.1</td>
</tr>
<tr>
<td>10</td>
<td>B-C-D</td>
<td>BC and CD</td>
<td>420</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 5.4: ODF’s of network 1
Figure 5.2: Average revenue of network 1 during the booking period for arrival rate of 0.01 under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP
Figure 5.3: Average revenue of network 1 during the booking period for arrival rate of 0.015 under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP
Figure 5.4: Average revenue of network 1 during the booking period for arrival rate of 0.02 under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP.

Figure 5.5 shows the average computational effort that is required to calculate the optimal booking policy under implementation of each control method.
Figure 5.5: Average computational effort that is required to calculate the optimal booking policy under implementation of three control methods: NET-BP, RCP-PDP and DARP-BP

Figure 5.6 shows the average revenue versus average load factor for all three control methods.
Figure 5.6: Average revenue of network 1 during the booking period under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP

When the booking policy is updated 30 times which in this case means everyday, the average revenue over all three rate and under implementation of RCP-PDP, is 1.9% higher than the one generated under implementation of DARP-PDP method and 8% higher than the one generated under NET-BP method.

5.2.2.2 EXPERIMENT 2

The numerical experiments in this section were performed on a network which we will refer to as network 2. Network 2 consists of 4 cities, 3 single legs and 12 ODFs. There are two differences between network 2 and 1. One is that there are two more itineraries in network 2, where both of them use all three legs
of the network. Another difference between network 2 and 3 is that the capacity of one of the legs of network 2 is more than the one of network 1. Figure 5.7 shows network 2, where A, B, C and D represent the three cities. The three single leg flights are: AB, BC and CD which go from A to B, B to C and C to D respectively. There is a total of 6 itineraries each with two different fares which makes total of 12 ODF’s. In figure 5.7, the ODF numbers are labeled above the arrows which show the itineraries. ODF 1 to 6 are single leg flights where ODF 7,8,9 and 10 are two leg flights and finally ODF 11 and 12 are three leg itineraries. Note that even though in figure 5.7 the multi leg itineraries are only shown with a direct arrow from origin to the destination, they actually use the legs that are between the origin and destination. Tables 5.3 and 5.4 list all the information regarding the ODF’s of network 2.

Figure 5.7: Network 2
We set the length of a time unit equal to 5 minutes and the length of the booking period equal to 30 days or 8640 time units. In order to stay consistent with the Poisson assumption we always set the arrival rate less than 0.03. Note that by arrival rate we mean probability of receiving a request for any of the ODF’s in one time unit. Figures 5.8 and 5.9 show the average revenue that is
generated during the entire booking period under implementation of the three
different control methods, NET-BP, RCP-PDP and DARP-PDP.

Figure 5.8: Average revenue of network 2 during the booking period for
arrival rate of 0.01 under implementation of three control methods: NET-BP,
RCP-PDP and DARP-PDP
Figure 5.9: Average revenue of network 2 during the booking period for arrival rate of 0.012 under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP.

Figure 5.10 shows the average computational effort that is required to calculate the optimal booking policy under implementation of each control method.
Figure 5.10: Average computational effort that is required to calculate the optimal booking policy under implementation of three control methods: NET-BP, RCP-PDP and DARP-BP on network 2.

Figure 5.11 shows the average revenue versus average load factor for all three control methods.
Figure 5.11: Average revenue of network 2 during the booking period under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP

When the booking policy is updated 30 times which in this case means once a day, the overall average revenue under implementation of RCP-PDP, is 0.9% higher than the one generated under implementation of DARP-PDP method and 5.7% higher than the one generated under NET-BP method. Figure 5.12 shows the average revenue per one seat for the three methods under different stage numbers.
Figure 5.12: Average revenue per seat for network 2 during the booking period under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP

5.2.2.3 EXPERIMENT 3

Experiment 3 is different than the other two experiments in a sense that the probability of receiving a request for an ODF changes throughout the booking period. More specifically, there are three different fares for each itinerary where the cheapest of each ODF is only available 21 days or more before the flight time. In order to achieve this, we divided the booking period into two segments where segment one is the last 21 days before the flight time and segment 2 is 30 to 21 days before the flight time. We performed experiment 3 on network 2 in which there are 4 cities, 3 legs itineraries where there are 3 different fares for each itinerary. Figure 5.13 shows the network and table 5.5 and 5.6 show the ODF’s of
the network along with the fares, probability of arrival of each ODF in one time unit at each of the two segments and the capacity of the legs.

Figure 5.13: Network 2 with more ODF’s for experiment
Table 5.7: ODF’s of network 2 for experiment 3

<table>
<thead>
<tr>
<th>ODF</th>
<th>Itinerary</th>
<th>Legs</th>
<th>Fare</th>
<th>Conditional Probability of Arrivals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Segment 1</td>
</tr>
<tr>
<td>1</td>
<td>A-B</td>
<td>AB</td>
<td>120</td>
<td>0.042</td>
</tr>
<tr>
<td>2</td>
<td>A-B</td>
<td>AB</td>
<td>100</td>
<td>0.083</td>
</tr>
<tr>
<td>3</td>
<td>A-B</td>
<td>AB</td>
<td>75</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>B-C</td>
<td>BC</td>
<td>240</td>
<td>0.083</td>
</tr>
<tr>
<td>5</td>
<td>B-C</td>
<td>BC</td>
<td>220</td>
<td>0.125</td>
</tr>
<tr>
<td>6</td>
<td>B-C</td>
<td>BC</td>
<td>180</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>C-D</td>
<td>CD</td>
<td>230</td>
<td>0.083</td>
</tr>
<tr>
<td>8</td>
<td>C-D</td>
<td>CD</td>
<td>210</td>
<td>0.125</td>
</tr>
<tr>
<td>9</td>
<td>C-D</td>
<td>CD</td>
<td>150</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>A-B-C</td>
<td>AB, BC</td>
<td>340</td>
<td>0.042</td>
</tr>
<tr>
<td>11</td>
<td>A-B-C</td>
<td>AB, BC</td>
<td>300</td>
<td>0.083</td>
</tr>
<tr>
<td>12</td>
<td>A-B-C</td>
<td>AB, BC</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>B-C-D</td>
<td>BC, CD</td>
<td>450</td>
<td>0.042</td>
</tr>
<tr>
<td>14</td>
<td>B-C-D</td>
<td>BC, CD</td>
<td>400</td>
<td>0.083</td>
</tr>
<tr>
<td>15</td>
<td>B-C-D</td>
<td>BC, CD</td>
<td>310</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>A-B-C-D</td>
<td>AB, BC, CD</td>
<td>570</td>
<td>0.083</td>
</tr>
<tr>
<td>17</td>
<td>A-B-C-D</td>
<td>AB, BC, CD</td>
<td>510</td>
<td>0.125</td>
</tr>
<tr>
<td>18</td>
<td>A-B-C-D</td>
<td>AB, BC, CD</td>
<td>490</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.8: Capacities of the legs of network 2 for experiment 3

<table>
<thead>
<tr>
<th>Leg</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>10</td>
</tr>
<tr>
<td>BC</td>
<td>15</td>
</tr>
<tr>
<td>CD</td>
<td>15</td>
</tr>
</tbody>
</table>

As before we set the length of a time unit equal to 5 minutes and the length of the booking period equal to 30 days or 8640 time units. Also, In order
to stay consistent with the Poisson assumption we always set the arrival rate less than 0.03. Again note that by arrival rate we mean probability of receiving a request for any of the ODF’s in one time unit. Figure 5.14 show the average revenue that is generated during the entire booking period under implementation of the three different control methods, NET-BP, RCP-PDP and DARP-PDP.

![Graph showing average revenue vs arrival rate for NET-BP, RCP-PDP, and DARP-PDP with Number of Stages = 30.]

Figure 5.14: Average revenue of network 3 during the booking period under implementation of three control methods: NET-BP, RCP-PDP and DARP-PDP

5.2.2.4 COMPUTATIONAL EFFORT

In order to show how the computational effort in our proposed method relates to the number of legs in the network and the number of fare classes on each leg, we ran some more experiments on networks presented in figures 5.15 to 5.19. more specifically, we liked to see how the computational effort changes as
we increase the number of legs in the network and the number of fare classes on each leg. Figures 5.15 to 5.19 show networks with 2, 4, 6, 8 and 10 legs. In these experiments we set the number of fare classes on each leg equal to 1, 2, 3, … or 6 as needed.

Figure 5.15: Network 3, a 2-leg network

Figure 5.16: Network 4, a 4-leg network

Figure 5.17: Network 5, a 6-leg network
Figures 5.20, shows how the time that takes to calculate the optimal policy changes as the number of legs in the network increases. The number of fare classes in this experiment was 3 on each of the legs. Figure 5.21 shows the computational time versus number of fare classes for network 6 where number of fare classes was changed from 1 to 6.
As it can be seen from figures 5.20 and 5.21, the computational effort that is involved in our proposed capacity control of a network method increases linearly with both number of legs and number of fare classes on each leg of a network.
5.3 SUMMARY

In this chapter we showed how to extend our proposed of capacity control on a single leg flight to a network of flights. We discussed our proposed method for decomposing the prices of the multi leg itineraries into single legs. After decomposing the prices of the multi leg itineraries we then use our proposed dynamic programming method (section 5.1.3) to find the optimal protection levels for each stage of the system on each leg. In our study we also tried a different prorating method in which we decomposed the prices of the multi-leg itineraries into the individual legs based on the initial capacity of the legs rather than expected remaining capacities of the stages. This method was not as good as prorating according to the expected remaining capacities (section 5.1.2) and we did not consider it any further. Also, we tried a new prorating on DARP-PDP method in which we did not normalize the decomposed prices to add up to the price of the multi leg itineraries. The resulted expected revenue was not as good as when we normalized the decomposed prices to add up to the multi-leg fares and be consistent with RCP-PDP method, therefore we did not consider that prorating method any further. In this section we also showed the numerical experiments on two different networks, in which we compared the performance of our proposed network capacity control method with two other methods. Our proposed method for network capacity control is able to generate higher average revenue. Finally, we relaxed the homogeneity assumption on the arrival process in order to model the realistic case of some discount fares being offered only with sufficient advanced purchase. Again RCP-PDP performed the best while DARP-
PDP was second best, as they both test the NET-BP method, which is the most commonly used method in practice today. It is interesting to note that RPC-PDP, while generating the most revenue, did so while filling fewer seats (*i.e.*, lower load factor). A lower load factor has some cost advantages, but also presents greater opportunity for additional revenue from standby passengers, where were not explicitly included in the model. In the next chapter we will discuss the conclusion and future research on this area.
CHAPTER 6
CONCLUSION AND FUTURE STUDY

In this work, we defined an airline revenue management problem and in particular the capacity control problem. We discussed some of the existing methods for capacity control of a single leg as well as a network of flights. We then proposed a method for capacity control on a single leg flight in chapter 4. In our proposed method for capacity control of a single leg flight we use an approximate dynamic programming approach involving large time chunks for each stage to achieve single leg performance that is nearly as good as the original DP method (Lee and Hersh 1993). As the time chunks approach zero, and therefore the number of stages increases, our method converges to the original DP method. Our proposed approximate dynamic programming approach is quite accurate for relatively large time chunks because we use PL and nested control method during each stage. Unlike many of the existing methods, in our proposed method, nesting is incorporated into the optimization procedure. We also calculate the expected total revenue by direct search rather than simulation. Furthermore, we do not telescope updates as is common in the literature. Instead, we find the optimal protection levels while taking into account the updates of the
booking policy in the future. More specifically, since we use a dynamic programming model to find the optimal protection levels at the beginning of each stage of the system, the future optimal policies at the next stages are also incorporated into the optimization procedure. As it can be seen from the numerical result that was presented in chapter 4, our proposed method of single leg capacity control is able to generate higher expected revenue than methods found in common practice. Furthermore, with only several updates of the booking policy throughout the booking period, the resulting expected revenue is nearly as good as the ideal method (section 2.2.3). In chapter 5, we presented how to extend our proposed single leg capacity control method to a network of flights. We did that by first decomposing the network into the single leg problems by using a prorating method and then controlling the booking process on the network by controlling the booking process on individual legs of the network. We also proposed a prorating method which is interconnected with our proposed search method for optimal protection levels on a single leg flight. Our proposed prorating method does not exist in the literature and as we saw in the numerical result that was presented in chapter 5, it does generate higher average revenue than the existing prorating method. However, while still manageable, the computational effort of our approach is considerably more than current methods in use. Much of that effort is due to the iterative procedure we use to establish the prorated prices. Future work could focus on developing other, more efficient, methods for prorating such as DARM-PDP, but perhaps better. There are some possible extensions to this work that we discuss in the next paragraphs.
In order to calculate the expected total revenue that is generated during the booking period on a single leg, we need to raise the transition probability matrix of the corresponding Markov chain, $P$, to the power of $T$ where $T$ represents the length of the booking period. In order to stay consistent with the Poisson arrival assumption, one time unit must be very small which in turn makes the value of $T$ large. When the L control method is implemented to control the booking process on a single leg flight, the size of the corresponding L Markov chain model is $(C+1) \times (C+1)$ where $C$ represents the initial capacity of the airplane. Therefore, calculating $P^T$ might be very time consuming. Because of the special format of the L Markov chain model, we can calculate $P^T$ in a very efficient way. The last row of $P$ has only one non-zero element, that is its last element of the row which is equal to one. Each of the other rows of $P$ has only two non-zero elements: the diagonal element and the one just to its right. Moreover, instead of multiplying $P$ by itself for $T$ times, one can partition it into smaller matrices and perform series of additions and multiplications on scalars to derive $P^T$. This approach would definitely increase the efficiency of the calculations.

In our proposed method for capacity control, we have made some assumptions to make the modeling simpler. One of the main assumptions is that we did not allow for cancellation during the booking period or no shows at the flight time. However these two issues should not be neglected in practice. An extension to this work can be to include cancellation and no shows into the modeling. Note that our Markov chain model is quite capable of adding these two features. In our proposed Markov chain models as it is now, the transitions are
allowed from each state to itself or to the one with one less remaining capacity. However if we allow for cancellation during the booking period, a transition to a state with one more capacity would also be possible. That would make the probability transition matrix of the corresponding Markov chain model a bit more complicated than its current format. There is also the issue that cancellations are not a completely memoryless process so that some approximation is necessary.

Another assumption that we have made throughout this work was that we did not allow for batch arrivals. In other words, we assumed that each request for a fare class or an ODF requests one seat only. Our Markov chain model is capable of relaxing this assumption as well. If we allow for batch arrivals, the process can transit from any state of the Markov chain model to itself or to any other state. That would make the Markov chain model much more complicated than its current format. Moreover, the resulting L Markov chain model will no longer have the nice format that we used in calculating the expected revenue. Finding an efficient way for calculating the expected revenue from this complicated Markov chain model could be a possible extension to our work.

In our propose single leg capacity control method, at the beginning of each sage we search for the optimal protection levels that generate the maximum expected revenue when the L control method is used to control the booking process. A possible extension to this work could be to search for the control method as well as the control parameters. More specifically, we can search for the protection levels and control method both together. According to our numerical experiments on comparing the performance of L and the R control
methods, R method occasionally outperforms L method, but since protection
levels are updated often in our approach, the choice of control method may be
critical only near flight time. Therefore, in order to save some computational
effort, one can search for the control method only at the last couple of stages
before the flight time where in all of the previous stages the search is only over
the protection levels that maximize the revenue that is generated under
implementation of the L control method.
BIBLIOGRAPHY


Littlewood, K., 1972, Forecasting and control of passenger bookings, AGIFORS Symposium Proceedings 12, 95-117.


APPENDIX A

SUPPORTING INFORMATION FOR CHAPTER 3

Supporting information for chapter 3 includes the MATLAB programming codes that were used for the numerical experiments.
% OUR METHOD -- Searching over the feasible set of PL's where
min, max and
unimodality rules are used to decrease the feasible set !

% Solves the problem for one stage -- suppose stage i

% INPUTS: p_class, r_class, T, Future(E), Minimum PLs

% OUTPUTS: Optimal_Revenue, Optimal_PL, Marginal, Expected
         Accepted Demand
% for capacities 1 to C

function [M, optimal_rev, optimal_pl, marginal] =
    lili_CMC(p_class, r_class, T, E, MIN_IN)

% ---- Setting the Parameters -------
capacity = length(E);
optimal_rev = zeros(capacity,1);
Expected_AD = zeros(capacity,1);
n = length(p_class);

% ------------------------------------

   PL_max = [ones(1,n)];   % at the beginning of each stage,
   PL_max is initialized

   for C=1:capacity
     % C
     fprintf('--------
     MAX = PL_max;
     MIN = MIN_IN(C,:);
     % MAX is updated at the end of each
     value of C

     % iteration 1 ----------------------
     pl = MIN;
     E_C = E(1:C);
     [M_TEMP,rev_to_go] =
        Left_Main_Exact(p_class, r_class, T, pl, E_C);
     max_rev_so_far = rev_to_go;
     optimal_pl(C,:) = pl;
     M{C} = M_TEMP;
     for ii= n:-1:1
       ub = MAX(ii);
       if ii~=n
         ub = min(ub,pl(ii+1));
       end
       while pl(ii)<ub
         pl(ii) = pl(ii) + 1;
Left_Main_Exact(p_class, r_class, T, pl, E_C);
    if rev_to_go > max_rev_so_far
        max_rev_so_far = rev_to_go;
        optimal_pl(C,:) = pl;
        M{C} = M_TEMP;
    else
        pl(ii) = pl(ii) - 1;
        break
    end % if
end % while

%---------- Next iterations --------------------------
DELTA = 100;
iteration = 1;
while DELTA ~= 0
    iteration = iteration + 1;
    DELTA = 0;
    for ii=n:-1:1
        % --------- Decreasing PL -----------
        DECREASED = 0;
        lb = max(0, MIN(ii));
        if ii~==1
            lb = max(lb, pl(ii-1));
        end
        while pl(ii)>lb
            pl(ii) = pl(ii) - 1;
            [M_TEMP, rev_to_go] = 
                Left_Main_Exact(p_class, r_class, T, pl, E_C);
            if rev_to_go > max_rev_so_far
                DELTA = rev_to_go - max_rev_so_far;
                max_rev_so_far = rev_to_go;
                optimal_pl(C,:) = pl;
                DECREASED = 1;
                M{C} = M_TEMP;
            else
                pl(ii) = pl(ii)+1;
                break
            end % if
        end %while
        % --------- Increasing PL -----------
        ub = MAX(ii);
        if ii~==n
            ub = min(ub, pl(ii+1));
        end
        while DECREASED==0 & pl(ii)<ub
            pl(ii) = pl(ii) + 1;
            [M_TEMP, rev_to_go] = 
                Left_Main_Exact(p_class, r_class, T, pl, E_C);
            if rev_to_go > max_rev_so_far
                DELTA = -max_rev_so_far + rev_to_go;
                max_rev_so_far = rev_to_go;
                optimal_pl(C,:) = pl;
                M{C} = M_TEMP;
            else
                pl(ii) = pl(ii)-1;
            end % if
        end %while
break
end % if
end % while
% -----------------------------------------------
end % for ii
end % while DELTA
%-------------------------------------------------------------
% check the optimal PL here for setting the maximum from now on

for j=1:n % revised on May 10th. because now fare class one might be blocked too!
if optimal_pl(C,j)< C
    PL_max(j) = optimal_pl(C,j);
else
    PL_max(j) = C+1;
end
% -----------------------------------------------------
optimal_rev(C,1) = max_rev_so_far;
end % best policy is found for all capacities of stage "stg"

marginal = optimal_rev - [0 ; optimal_rev(1:capacity-1)];

% BASE CODE ! The Heart of the method !
% Calculates (immediate+future revenue = rev_to_go) for given pl and C and future.

% INPUT: p_class, r_class, T, pl (a specific one not necessary the optimal), future(E)
% OUTPUT-1: Expected Accepted Demand at the beginning of T time units
% OUTPUT-2: rev_to_go

function[M_TEMP, rev_to_go] = Left_Main_Exact(p_class,r_class,T,pl,E)

%--------------------------------------

n = length(pl);
C = length(E);
% ------------------------------------

format long g

% LOF(i) indicates the number of columns with lowest fare open of i
% to get the pl from LOF we have to start from LOF(n) and back to LOF(1)
% LOF is an alternative vector to pl which has the same information

LOF = [pl(2:n) , C] - pl;
i=1;

for ind=n:-1:1
    if LOF(ind)~=0
        NT(i) = ind;
        K(i) = LOF(ind);
        i = i+1;
    end
end
if sum(LOF)<C
    NT(i) = 0;
    K(i) = C-sum(LOF);
end

row = 1;
for i=1:length(NT)
    V_NT(i) = 0;
    if NT(i)~=0
        top = r_class(1:NT(i))*p_class(1:NT(i))';
        butt = sum(p_class(1:NT(i)));
        V_NT(i) = top / butt;
    end
    for x=1:K(i)
        V(row) = V_NT(i);
        P(row,row+1) = sum(p_class(1:NT(i)));
        P(row,row) = 1-P(row,row+1);
        row = row + 1;
    end
end

P = [P ; [zeros(1,C) , 1]]; % Left transition probability matrix

E = [0 ; E];
V = [0 , V];

for i=1:C+1
    k = C-i+2;
    R(i) = sum(V(1:i))+E(k);
    AD(i) = i-1;
end

M = P^T;
rev_to_go = M(1,:)'*
M_TEMP = M(1,:);
Supporting information for chapter 5 includes the MATLAB computer programming codes that were used for numerical experiments of chapter 5.
% Modified on Feb/15/07 according to the explanations on the note book

function [Network_r_class,flag] = NEW_OURS_DECOMPOSER(M,r_class,A)

[n,m] = size(A);
Network_r_class = zeros(n,m);

% flag shows the result of the decomposition,
% flag=1 -> Exception1 , flag=2 -> Exception2 , flag=3 -> regular
% decomosition , flag=4 -> none of the above, DLP was used to decompose !

flag = zeros(1,m); %shows the result of the decomposition

for i=1:n
    Network_r_class(i,find(A(i,:))) = r_class(find(A(i,:)));
end

for j=1:m % for each itinerary

    % ------- Decomposing 2-leg itineraries -----------------------
    if sum(A(:,j)) == 2 % 2 leg itineraries
        %------- Decomposition for 2 legs ------------------------
        r = r_class(j);
        Aj = A(:,j);
        [M1,M2] = M{find(Aj)};
        C1 = length(M1);
        C2 = length(M2);
        if M1(1)+M2(1)<r % Exception 1 -- Block nothing on any
            DELTA = r-(M1(1)+M2(1));
            p1 = M1(1)+0.5*DELTA;
            p2 = r-p1;
            Network_r_class(find(Aj),j) = [p1 ; p2];
            flag(j) = 1;
        else
            if M1(C1)+M2(C2)>r % Exception 2 -- Block all on both
                DELTA = M1(1)+M2(1)-r;
                p1 = M1(C1)-0.5*DELTA;
                p2 = r-p1;
                Network_r_class(find(Aj),j) = [p1 ; p2];
                flag(j) = 2;
            else
                MIN_DIFF = 10000;
                for ii=C1:-1:2
                    for jj=C2:-1:2
                        sum1 = M1(ii) + M2(jj);
                        sum2 = M1(ii-1) + M2(jj-1);
                        if r>=sum1 & r<=sum2 & M1(ii)<M1(ii-1) &
                            M2(jj)<M2(jj-1)
                            DIFF = ((ii-1)/(jj-1)) - (C1/C2);
                            if abs(DIFF) < MIN_DIFF
                                MIN_DIFF = abs(DIFF);
                                m1 = ii;
                        end
                    end
                end

    end
end

end
m2 = jj;
end % if r
end %jj
end %ii
% ---- decomposition ----
%[m1-1,m2-1,MIN_DIFF]
LH1 = max(M1(m1),r-M2(m2-1));
RH1 = min(M1(m1-1),r-M2(m2));
p1 = LH1 + 0.5*(RH1-LH1);
p2 = r - p1;
Network_r_class(find(Aj),j) = [p1 ; p2];
flag(j) = 3;
end % if Exception 2
end % if Exception 1
if flag(j)==0 %no price was decomposed yet !
    Network_r_class =
    DLP_DECOMPOSER(T_tot,rule,p_class,r_class,C,A);
    flag(j)=4;
end
end % if 2-leg itinerary
%----------------------------------------------------------
% -------- Decomposing 3-leg itineraries ! ----------------
if sum(A(:,j)) == 3 % 3 leg itineraries
%--------- CMC Decomposition for 2 legs -------
r = r_class(j);
Aj = A(:,j);
[M1,M2,M3] = M{find(Aj)};
C1 = length(M1);
C2 = length(M2);
C3 = length(M3);
if M1(1)+M2(1)+M3(1)<r % Exception 1
    DELTA = r-(M1(1)+M2(1)+M3(1));
p1 = M1(1)+(DELTA/3);
p2 = M2(1)+(DELTA/3);
p3 = r-p1-p2;
    Network_r_class(find(Aj),j) = [p1 ; p2 ; p3];
    flag(j) = 1;
else
    if M1(C1)+M2(C2)+M3(C3)>r % THINK !! Exception 2 --
        Block all on both
        DELTA = M1(1)+M2(1)+M3(1)-r;
p1 = M1(1)-(DELTA/3);
p2 = M2(1)-(DELTA/3);
p3 = r-p1-p2;
        Network_r_class(find(Aj),j) = [p1 ; p2 ; p3];
        flag(j) = 2;
    else
        MIN_DIFF = 10000;
        for ii=C1:-1:2
            for jj=C2:-1:2
                for kk=C3:-1:2
                    lb1 = M1(ii);
                    ub1 = M1(ii-1);
lb2 = M2(jj);
ub2 = M2(jj-1);
lb3 = M3(kk);
ub3 = M3(kk-1);
lb12 = lb1+lb2;
ub12 = ub1+ub2;
ub13 = ub1+ub3;
ub23 = ub2+ub3;
lb123 = lb1+lb2+lb3;
ub123 = ub1+ub2+ub3;
if r>=lb123 & r<=ub123 & lb12<ub12 & lb13<ub13 & lb23<ub23 & lb1<ub1 & lb2<ub2 & lb3<ub3
DIFF = abs(((ii-1)/(jj-1))-(C1/C2))+abs(((ii-1)/(kk-1))-(C1/C3))+abs(((jj-1)/(kk-1))-(C2/C3));
if DIFF < MIN_DIFF
  MIN_DIFF = DIFF;
  m1 = ii;
  m2 = jj;
  m3 = kk;
end
end % if r
end % for kk
end %ii
% ---- decomposition ----  THINK !!!!!!!!!!!!!!!
%[m1-1,m2-1,m3-1,MIN_DIFF]

lb1 = M1(m1);
ub1 = M1(m1-1);
lb2 = M2(m2);
ub2 = M2(m2-1);
lb3 = M3(m3);
ub3 = M3(m3-1);

LH1=max(lb1,r-(ub2+ub3));
RH1=min(ub1,r-(lb2+lb3));

LH2=max(lb2,r-(ub1+ub3));
RH2=min(ub2,r-(lb1+lb3));

p1 = (LH1+RH1)/2;
p2 = (LH2+RH2)/2;
p3 = r-p1-p2;

Network_r_class(find(Aj),j) = [p1 ; p2 ; p3];
flag(j) = 3;
end % if Exception 2
end % if Exception 1
if flag(j)==0 %no price was decomposed yet!
  Network_r_class = DLP_DECOMPOSER(T_tot,rule,p_class,r_class,C,A);
  flag(j)=4;
end
function [SIM AVG, AVG LF, time] = DCMP_OURS_CONV(p_class,r_class,S,C,A,DEMAND)

% ------------ Setting the variables -------------
rule = 'DO SCALE'; % in this case rule is always to
scale !
%--------------------------------------------------

start_time = cputime;
%------- setting the parameters ----------

[sample_size,T_tot] = size(DEMAND);
[n,m] = size(A); % n = number of legs m = number of
itineraries
stg_num = S;

T_stg = T_tot / stg_num;

%--------------------------------------------------
% initializing C_bar

for stg=1:stg_num
    C_bar(:,stg) = C;
end

for itr=1:10
    % display ------------------------------------------
    % ITERATION = itr
    for stg=1:stg_num
        if stg == 1
            for i=1:n
                E{i} = zeros(C(i),1);
            end
        end
end

end % if 3-leg itinerary
end %
end
% on first stage we use DLP_DECOMPOSER!
LEG_r_class =
DLP_DECOMPOSER(T_stg,rule,p_class,r_class,C_bar(:,stg),A);
else
[LEG_r_class,flag] =
CMC_DECOMPOSER(C_bar(:,stg),M,r_class,A);
end % if stg = 1

% Sorting the decomposed prices on each leg
for i=1:n % for each leg
index = find(A(i,:));
leg(i) = struct('itn',[],'r_class',[],'p_class',[]);
leg(i).r_class = LEG_r_class(i,index);
[leg(i).r_class , sort_ind] =
sort(leg(i).r_class,'descend');
leg(i).itn = index(sort_ind);
ITN{stg , i} = leg(i).itn;
leg(i).p_class = p_class(leg(i).itn);
end % for i=1

% Finding the PL* using our method
for i=1:n    % for all legs -- i is the leg index
p = leg(i).p_class;
r = leg(i).r_class;
if stg==1
MIN = zeros(C(i),length(p));
else
MIN = MIN_MAKER(r,M{i},C(i));
end % if stg=1
[SYSTEM_STATE{stg,i},optimal_rev,optimal_pl,marginal] =
lili_CMC_NEW(p,r,T_stg,E{i},MIN);
% SYS_STATE = SYSTEM_STATE{stg,i}
E{i} = optimal_rev;
M{i,1} = marginal;
OPT_PL_SET{stg,i} = optimal_pl;
end % for i=1
% C_bar(:,stg)
% LEG_r_class'
end % stg

INDEX = index_finder(m,ITN);
% finding the expected capacity

for i=1:n
x=C(i);
C_bar(i,stg_num) = C(i);
for k=1:C(i)+1
x_ref(k,1) = C(i) - k + 1;
end
for stg=stg_num-1:-1:1
if x==0
C_bar(i,stg) = 0;
x = 0;
else
x=SYSTEM_STATE{stg,i}{x}*x_ref((C(i)-x+1):(C(i)+1));
    x=ceil(x);
    C_bar(i,stg)=x;
end
end
end

end % C_bar
end % itr

time = cputime -start_time;

%%------------------------------ SIMULATION PROCESS --------------
for iteration=1:sample_size
    AD = zeros(1,m);
    x = C;
    d_ind = 1;
    for stg=stg_num:-1:1
        Stage_Capacity = x;
        for t = T_stg:-1:1
            d = DEMAND(iteration,d_ind);
            d_ind = d_ind + 1;
            if d~0
                BEFORE = x';
                accept = 1;  % assume that is accepted
                for i=1:n
                    if INDEX{stg}(d,i)~=0
                        if x(i) ~= 0
                            accept = accept*(x(i)>OPT_PL_SET{stg,i}(Stage_Capacity(i),INDEX{stg}(d,i)));
                        end
                    end
                end
            else
                accept = -2;  % this means that there was no capacity left
            end
        end
        if accept == 1
            AD(d) = AD(d) + 1;
            x(find(INDEX{stg}(d,:)) ) = x(find(INDEX{stg}(d,:))) - 1;
        end
    end
end

% fprintf('--------------')
% d
% BEFORE
% accept
% AD
% AFTER = x'
function [SIM_AVG, AVG_LF, time] = DCMP_DLP_SIM(p_class,original_r_class,S,C,A,DEMAND,rule)

% DLP-OURS - Remaining capacity version --> based on remaining capacity
% Revised on April/8/07 to be consistent with the definitions in summer 06

% --------------------- NOTATION --------------------------------
% OPT_PL_SET{stage,leg} -- ITN{i,leg} -- E{leg} -- M{leg}-- INDEX{stg}
% ---------------------------------------------------------------

% --------- Setting the varibales ---------------
% rule = 'DO SCALE'; % rule 1-A
% rule = 'NO SCALE'; % rule 1-B
%
% --------- setting the parameters ---------------

[sample_size,T_tot] = size(DEMAND);
[n,m] = size(A); % n = number of legs  m = number of itineraries
stg_num = S;
T_stg = T_tot / stg_num;

%-------------------------------

time = 0;

for iteration=1:sample_size

display ----------------------
% iteration

%display ----------------------

end
end
end
end
end

REV(iteration) = AD*r_class';
LF(iteration) = (sum(C)-sum(x))/sum(C); %according to CMC-Harper idea -- sum(Oi)/sum(Ci)- Oi=occupied on leg i
end

SIM_AVG = [mean(REV) , sqrt(var(REV) / sample_size)];
AVG_LF = mean(LF);

fprintf('--------------')
r_class = original_r_class;
demand_ind = 0;
AD = zeros(1,m);
x = C;

for stg=stg_num:-1:1

    %display ************
    %STG = stg

    start_time = cputime;

    REM_TIME = T_stg*stg;
    Stage_Capacity = x;  % initializing the remaining capacity at the beginning of each stage

    for i=1:n
        if x(i) == 0 % no more seat left on that leg
            means fare of that class would eb zero from now on
            Itns_on_Leg_i = find(A(i,:));
            r_class(Itns_on_Leg_i) = zeros(1,length(Itns_on_Leg_i));
        else
            E{i} = zeros(Stage_Capacity(i),1);
        end
    end

    %LEG_r_class =
    DLP_DECOMPOSER(REM_TIME,rule,p_class,r_class,C,A);  % relative to the original capacity

    LEG_r_class =
    DLP_DECOMPOSER(REM_TIME,rule,p_class,r_class,Stage_Capacity,A);

    %LEG_r_class'
    % Sorting the decomposed prices on each leg
    for i=1:n  % for each leg
        index = find(A(i,:));
        leg(i) =
        struct('itn',[],'r_class',[],'p_class',[]);
        leg(i).r_class = LEG_r_class(i,index);
        [leg(i).r_class , sort_ind] =
        sort(leg(i).r_class,'descend');
        leg(i).itn = index(sort_ind);
        ITN{1,i} = leg(i).itn;
        leg(i).p_class = p_class(leg(i).itn);
    end  % for i=1

    % Finding the PL* using our method
    for i=1:n  % for all legs -- i is the leg index
        if Stage_Capacity(i) ~= 0
            p = leg(i).p_class;
            r = leg(i).r_class;
        end
    end

end
MIN = zeros(Stage_Capacity(i),length(p));
[DUMM,optimal_rev,optimal_pl,marginal] =
  lili_CMC_NEW(p,r,REM_TIME,E{i},MIN);
  OPT_PL_SET{1,i} = optimal_pl;
end
end  % for i=1

% Revised on Feb/23 -- there is one INDEX for each
stage
% Defining the proper index for each demand on each
leg on each stage
INDEX = index_finder(m,ITN);
time = time + cputime - start_time;

%--------------------------- SIMULATION PROCESS -----------------------

for t = T_stg:-1:1
  demand_ind = demand_ind + 1;
d = DEMAND(iteration,demand_ind);
  if d~=0
    accept = 1;  % assume that is accepted
    for i=1:n
      if INDEX{1}(d,i)~=0
        if x(i) ~= 0
          accept = accept*(x(i) >
            OPT_PL_SET{1,i}(Stage_Capacity(i),INDEX{1}(d,i)));
        else
          accept = -2;  % this means that there
          was no capacity left
        end
      end
    end
  end
  if accept == 1
    AD(d) = AD(d) + 1;
    x(find(INDEX{1}(d,:))) =
      x(find(INDEX{1}(d,:))) - 1;
  end
end  % for t

% AD_1 = sum(AD([1,2,7,8,11,12]))
% AD_2 = sum(AD([3,4,7,8,9,10]))
% AD_3 = sum(AD([5,6,9,10,11,12]))
% x
end  % stg

% AD

REV(iteration) = AD*original_r_class';
LF(iteration) = (sum(C)-sum(x))/sum(C);  %according to CMC-
Harper idea -- sum(Oi)/sum(Ci)- Oi=occupied on leg i
end  % iteration

163
SIM_AVG = \{\text{mean(REV)} , \sqrt{\text{var(REV)} / \text{sample\_size}}\};
AVG_LF = \text{mean(LF)};

time = \text{time} / \text{sample\_size};