NORTHWESTERN UNIVERSITY

Analytics for Airline Revenue Management and Irregular Operations

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Industrial Engineering and Management Sciences

By

Bill Pun (Chan Seng Pun)

EVANSTON, ILLINOIS

June 2013

To my dearest parents

Pun Wai and Chan Hang Chon

and my lovely sisters

Pun Lei Chi and Pun Ieng Ieng

for their endless patience and unconditioned love.

ABSTRACT

Analytics for Airline Revenue Management and Irregular Operations

Bill Pun (Chan Seng Pun)

In this dissertation, we study problems in both revenue management (RM) and irregular operations recovery. The first chapter is devoted to RM in airline passenger. We consider a problem that allocates seats to fare classes, and captures capacity nesting and customer upsell. While capacity nesting allows airlines to sell seats allocated to low-yield classes to high-yield passengers, customer upsell allows low-yield passengers to purchase seats reserved for high-yield classes. We adopt an approximate dynamic programming algorithm to iteratively approximate the complicated objective function with piecewise linear functions. We observe that the resulting allocation policy outperforms a popular bid-price policy up to 35% when demand and upsell probability are high.

The second chapter is about RM in air cargo. We study the underlying capacity allocation problem in the mid-term capacity allocation process, in which shippers bid flight capacity on multiple flights to receive discounted shipping rates and guaranteed space. The model minimizes demand covariance between the bids and future volatile free-sales demand subject to a revenue lower bound and all necessary allocation requirements. Due to its complexity, we decompose the problem by a flight partition and a set of demand clusters. We show using simulation that our partitioning algorithm is robust, and the resulting allocation increases revenue by 2% if the revenue lower bound is high and demand covariance is captured.

In the final chapter, we study a fully integrated recovery problem that recovers disrupted flight schedules by iteratively and simultaneously recovering resources (aircraft, crews, and passengers). The problem is solved by Benders decomposition, where the master problem is an extended fleet assignment problem, and the subproblems are the resource recovery problems. Several decomposition and algorithmic strategies are developed to reduce the total running time. We show that our solution can outperform a partially integrated solution used in practice by as much as 8%, which accounts for one million dollars in saving per disruption.

ACKNOWLEDGEMENTS

I would like to thank my dissertation advisor Diego Klabjan for his guidance and patience in supervising this dissertation, which would not have been possible without his assistance.

I would like to express my gratitude to Fikri Karaesmen, Seyed Iravani, and Irina Dolinskaya for joining my dissertation committee and offering generous support over the years.

I am very grateful to have the opportunities to work with revenue management professionals: Sergey Shebalov, Jamison Graff, and Raja Kasilingam. Especially, I am indebted to Dr. Shebalov, who provided me many insightful comments that help improve the quality of my first and last chapters, and invited me to join him at Sabre Holdings to work on revenue management problems. My gratitude is also toward Dr. Graff, who sent me to JDA's clients around the global to learn their cargo systems. I also thank Dr. Kasilingam for sharing his expertise in cargo revenue management while I was in Hong Kong.

I would like to express my gratitude to Steve Golbeck, my fantastic roommate since the day I passed my Ph.D. entrance exam. Although the apartment was out of shape ever since we moved in, without him keeping it minimally functional, my off-campus life could have been a disaster.

I owe sincere and earnest thankfulness to Joanna Wu and Zi Yang, two of my best friends in Chicago who dragged me from my office and shared a lot of happy moments together. I also particularly thank Luis Guimarães and Frank Schneider. Although they only stayed in Chicago for a relatively short period of time, they colored my off-campus life in many different ways.

I am obliged to many of my colleagues, who share their lives with me and patiently listen to my complaints when things did not go smoothly. Especially, I want to thank Geng Yue for all the moments that we passed together and the helps that I received.

Last but not least, I express my greatness gratitude to my lifelong best friends U Sio Chong and Julieta Guerreiro, who grow up with me and have been spiritually supporting me all these years.

CONTENTS

List of Tables

LIST OF FIGURES

List of Algorithms

Chapter 1

ITINERARY-BASED CONTROL WITH NESTING AND UPSELL

In order to accept future high-yield booking requests, airlines protect seats from low-yield passengers. More seats should be reserved when passengers faced with closed fare classes can upsell to open higher fare classes. We address the airline revenue management problem with capacity nesting and customer upsell, and formulate it as a stochastic optimization problem to determine a set of static protection levels for each itinerary. We apply approximate dynamic programming to approximate the objective function by piecewise linear functions, whose slopes (corresponding to marginal revenues) are iteratively updated and produced by a sophisticated heuristic that simultaneously handles both nesting and upsell. The resulting allocation policy is tested on a real airline network and benchmarked against the randomized linear programming bid-price policy under various demand settings. Simulation results suggest that the proposed allocation policy significantly outperforms when incremental demand and upsell probability are high.

Key words: network revenue management, capacity nesting, customer upsell, approximate dynamic programming

1.1 INTRODUCTION

Airline Revenue Management (RM) is about making a decision whether or not a booking request should be accepted for a seat in a given fare class at a particular point in time. If the request is accepted, the

revenue is immediately collected. Otherwise, the airline reserves the seat for a passenger who might book in the near future and pay a higher fare. In short, the goal of RM is to maximize revenue by managing a capacity-constrained flight network. An RM control policy for such a purpose is often constructed based on the primal and/or dual solutions of a resource allocation problem. Constructing a good control policy has been an interesting topic to both practitioners and researchers for decades. Challenges are mainly due to the size of the flight network, the dynamic nature of the airline business, and the stochastic booking behaviors of passengers. These challenges motivate the airline industry to develop efficient and well-performed heuristics.

At the beginning of the decision process, an underlying optimization problem allocates seats to passenger classes before passengers start booking, and it is typically resolved later during the booking process. By allocating seats to each class appropriately, seats that could be sold at higher fares can be protected from low-yield passengers who usually book their tickets months in advance. In practice, instead of using the allocation solution as is, allocated seats are nested over fare classes to set up protection levels, so that airlines can sell empty seats allocated to low-yield classes at a higher fare to high-yield passengers whose classes are fully booked. Furthermore, when it is profitable to do so, airlines adjust the allocation to recapture low-yield passengers at a higher fare by prematurely closing their corresponding low-yield classes. These are capacity nesting and customer upsell, which are two commonly discussed and desired features of the airline RM problem. In this paper, we refer to the allocation solution without nesting as the partitioned allocation policy, and the derived protection levels as the nested allocation policy.

A popular alternative control policy to the partitioned and nested allocation policies mentioned above is bid-price policy, which consists of a threshold price (bid price) for each itinerary. A booking request from a passenger paying a fare above the bid price is accepted given the itinerary is open. In practice, the optimal bid price is approximated by summing the (approximated) marginal revenue of a seat over all flights in the itinerary. Lastly, by assuming a relationship between the passenger population and fare level (price elasticity), the fare can be adjusted to achieve a similar capacity protection effect. For more details, we refer the reader to Williamson (1992) and Talluri and van Ryzin (1998) for bid prices, Bitran and Caldentey (2003) for pricing solutions, de Boer et al. (2002) for numerical experiments, and McGill and Van Ryzin (1999) for a comprehensive review of RM.

Although the bid-price policy has been extensively studied, the traditional seat allocation policy remains popular. It is due to the fact that many existing RM systems are built to handle allocation policies for their

capability to include customer behavior (cancellation, no-show, and upsell) more intuitively and to provide a more granular control over the network. With the ever tightening revenue margin nowadays, capturing customer behavior is vital to the prosperity of the airlines.

In practice, bid-prices are used not as a control policy but as means to prorate itinerary fares and to decompose the flight network during the pre-optimization phase, where the marginal revenue of a seat is approximated by a dual solution of a relaxed seat allocation model. After the fares are prorated, virtual classes are defined at the flight level and mapped to the original fare classes. The protection level for each virtual class is then determined by an efficient leg-based heuristic that captures customer behavior. Our work simplifies existing allocation-based RM systems by eliminating both the use of a proration scheme and the need of defining virtual classes. It also provides an allocation policy that requires no changes in management practice.

In this paper, we model the network RM problem as a stochastic programming problem under the assumption that low-yield passengers book first. The problem is similar to a two-stage stochastic programming problem. It first maximizes revenue by allocating seats to each itinerary subject to flight capacity. Then, given an allocation level and demand realization for each itinerary, sales are maximized by distributing the allocated seats to each fare class while considering nesting and upsell.

The problem is solved by an approximate dynamic programming (ADP) algorithm that we developed to approximate the complicated objective function by piecewise linear functions. The slopes for each piecewise linear function are estimated by a sophisticated heuristic that locally adjusts class-level seat allocation given new demand information. In addition, when upsells can be ignored, we show that our model is the same as the model in Curry (1990), and the nested allocation policy, similar to the partitioned allocation policy, enjoys the asymptotic optimality in Cooper (2002).

Simulation results on a medium airline network using a real-world dataset are discussed. Sensitivity analyses are conducted to evaluate the performance of the nested allocation policy by varying demand magnitudes and upsell probabilities. When both demand and upsell probability are high, we observe that the proposed allocation policy significantly outperforms the RLP bid price policy in Talluri and Van Ryzin (1999).

Our contributions are the following.

1. We provide a new stochastic programming formulation to model the network RM problem which considers both capacity nesting and customer upsell at the itinerary level.

2. We revisit the itinerary-based network RM problem, extend it, expose its algorithmic structure, and solved it with a parallelizable ADP algorithm that approximates the complicated objective function. Our method does not require the use of fare proration or virtual classes, and returns a static allocation policy that can be easily stored and implemented.
3. We devise a sophisticated heuristic that serves as the core of the ADP algorithm. Given the number of seats available to an itinerary, it approximates the set of protection levels and the associated seat margin. We numerically show that it significantly outperforms the choice-based EMSR heuristic by Gallego et al. (2009) when the number of fare classes is high.
4. We justify the use of the nested allocation policy by revealing its asymptotic optimality when both capacity and demand increase, and no upsell is considered.
5. We benchmark our allocation policy against the RLP bid-price policy and provide a comprehensive numerical study to evaluate the performance of the nested allocation policy when demand is scaled and when upsell probabilities cannot be accurately estimated, a common problem in practice.

We outline our paper as follows. Section 1.2 provides a general overview on several well-known seat allocation models. Section 1.3 presents our itinerary-based nesting model with upsell, and Section 1.4 elaborates the approximate dynamic programming algorithm that we apply to solve our problem. Several other heuristics and algorithms are also presented. Section 1.5 discusses asymptotic optimality of the nested allocation policy when no upsell is considered. Section 1.6 reports simulation results, and Section 1.7 concludes the paper.

1.1.1 Literature Review

We briefly discuss previous work closely related to our material. Starting with two passenger classes defined by their fares, Littlewood (1972) derives the optimality condition to determine the optimal protection level for the lowest fare class when its passengers book first. Brumelle and McGill (1993), Curry (1990), and Wollmer (1992) independently generalize the optimality condition to multiple classes. While Wollmer (1992) handles discrete demand, Curry (1990) assumes continuous demand, and Brumelle and McGill (1993) is applicable to both. Furthermore, Curry (1990) proposes a two-step optimization procedure to obtain protection levels at the itinerary level. While we also consider the RM problem at the itinerary level, we formulate the problem as a stochastic programming problem and extend it to capture upsells.

A stochastic approximation algorithm is proposed by van Ryzin and McGill (2000) to approximate

optimal protection levels. Their assumptions are the same as those in Brumelle and McGill (1993) that bookings are independent and arrive in a low-to-high fare order. The algorithm does not rely on an apriori distribution and provably converges to optimality. However, their algorithm is single leg, and hence, does not capture the network effect.

Higle (2007) models the seat allocation problem as two-stage stochastic programming problem. Her model captures demand at the origin-destination level and determines protection levels at the flight level. Our model is similar, but it captures both nesting and upsell at the itinerary level. Also, we do not require classes to be defined at the network level (across itineraries), which is nontrivial and required in her model for her flight-level nesting scheme to work.

Taking the two-stage framework one step further, Chen and Homem-de-Mello (2010) model the network RM problem as a multi-stage stochastic programming problem. However, doing so rises tractability concerns, and they do not capture upsell.

Recent attentions have been given to integrating customer upsell with traditional revenue management models. Fiig et al. (2010) derive a fare adjustment scheme to handle discrete choice models. The scheme was tested by a passenger origin-destination simulator. Gallego et al. (2009) develop several choice-based expected marginal seat revenue (EMSR) algorithms for a problem with multinomial logit (MNL) demand. They show superior performances over both EMSR-w with upsell from Belobaba and Weatherford (1996) and an adapted version of Fiig et al. (2010). We compare our algorithm directly with their path independent choice-based EMSR algorithm and achieve a significantly higher revenue when the number of fare classes increases.

Zhang and Adelman (2009) propose the use of approximate dynamic programming to solve the network RM problem with customer choice. They provide multiple bounding results and a column generation algorithm to handle the MNL choice model with a requirement that product sets are disjoint over customer segments. Their model estimates the set of products to sell instead of a set of seats to protect. Different from us, they assume one passenger per arrival, approximate the optimal DP directly with a math programming problem, and focus on the flight-level bid-price control. Nonetheless, it is non-trivial to extend their approach to capture more than one passenger in between two time periods, and their model cannot be parallelized to exploit nowadays multi-core computing environment.

1.2 Overview of Revenue Management

We define the following sets and parameters:

- T set of time periods (reading days),
- F set of flights,
- I set of itineraries,
- I_f set of itineraries that uses flight f,
- F_i set of flights in itinerary i,
- $C_i = \{1, 2, \ldots, |C_i|\}$ set of fare classes on itinerary i ordered by fares with 1 referring to the full fare class,
- $J = \{(i, c)\}_{c \in C_i, i \in I}$ set of products (itinerary-fare combinations),
- J_f set of products that uses flight f,
- r_j fare of product j,
- D_{jt} random variable that represents the number of booking requests for product j at time t,
- d_{jt} realization of D_{jt},
- K_f number of available seats on flight f,
- Π_j protection level for product j, where a protection level for a product is the total number of seats reserved for all strictly higher classes.

Decision variables are defined as follows:

- y_i number of seats allocated to itinerary i,
- x_{ic} number of seats reserved for class c on itinerary i, and
- z_{ic} number of empty seats extracted from all lower and equal classes to class c on itinerary i.

We often use j and (i, c) as subscripts interchangeably. If demand is aggregated over all time periods, or each class of demand has a designated arrival time period, then the subscript to time t is ignored. Any multidimensional quantity is denoted in bold. A superscript * denotes the optimal objective value of a problem. A subscript of a multi-dimensional quantity refers to the sliced set in the subscripted dimension, e.g. $\mathbf{\Pi}_i = \{\Pi_{i1}, \ldots, \Pi_{i|C_i|}\}$ is the set of protection levels for all classes on itinerary i. Minimum and maximum operations are assumed component-wise, and $(\cdot)^+$ represents $\max\{\cdot, 0\}$. For ease of notation,

we define $\mathbf{\Pi}_i^c = \{\Pi_{i1}, ..., \Pi_{ic}\}$ to be the set of protection levels for fare classes with a fare at least r_{ic}. Additionally, let $\mathbf{K} = \{K_1, ..., K_{|F|}\}$ be the set of flight capacity. We define the set of feasible itinerary-level allocations by

$$\mathcal{I}(\mathbf{K}) = \left\{ \mathbf{y} : \sum_{i \in I_f} y_i \leq K_f \text{ for } f \in F \text{ and } y_i \in \mathbb{N} \text{ for } i \in I \right\},$$

and the set of feasible product-level allocations by

$$\mathcal{J}(\mathbf{K}) = \left\{ \mathbf{x} : \sum_{j \in J_f} x_j \leq K_f \text{ for } f \in F \text{ and } x_j \in \mathbb{N} \text{ for } j \in J \right\}.$$

The summation in $\mathcal{I}(\mathbf{K})$ or $\mathcal{J}(\mathbf{K})$ represents the requirement that the total allocation to itinerary or product cannot exceed the number of seats available on each flight. Additionally, we define the allocation policy mapping function by

$$\mathcal{P}(\mathbf{\Pi}_i, y_i) = \{\mathbf{x}_i : x_{ic} = \min\{y_i, \Pi_{ic}\} - \Pi_{ic-1} \text{ for } c \in C_i\},$$

which maps a set of protection levels to a partitioned (non-nested) allocation given the total number of seats available to an itinerary. It converts a nested allocation policy to a partitioned allocation policy. Similarly, we define the inverse mapping by

$$\mathcal{N}(\mathbf{x}_i) = \left\{ (\mathbf{\Pi}_i, y_i) : \Pi_{ic} = \sum_{c' \leq c} x_{ic'} \text{ for } c \in C_i, y_i = \sum_{c \in C_i} x_{ic} \right\},$$

which takes the class-level allocation and computes the corresponding set of protection levels, and hence, converting a partitioned allocation policy to a nested allocation policy. Note that $\mathcal{P}(\mathbf{\Pi}_i, y_i)$ is surjective, since multiple sets of $(\mathbf{\Pi}_i, y_i)$ can yield the same $\mathbf{x}_i$, and $\mathcal{N}(\mathbf{x}_i)$ is injective with $y_i = \Pi_{i|C_i|}$.

Starting with the dynamic programming (DP) formulation of the RM problem described in Talluri and van Ryzin (1998), we discuss several tractable approximation models to the DP value function in each time period. The DP accurately models the problem if the probability of having more than one arrival between two time periods is negligible, or as a special case, if arrivals between two time periods only belong to the same class (see Robinson (1995)). Mathematically, the optimality equation is

$$v_t(\mathbf{K}) = \mathbb{E}\left[\max_{\substack{\mathbf{x} \in \mathcal{J}(\mathbf{K}) \\ 0 \leq x_j \leq D_{jt}, j \in J}} \sum_{j \in J} r_j x_j + v_{t-1}\left(\left\{ K_f - \sum_{j \in J_f} x_j \right\}_{f \in F} \right) \right] \tag{1.1}$$

with $v_0(\cdot) = 0$. For each time period t, a decision has to be made about the number of bookings to accept. In the end, the DP returns a dynamic control policy to indicate which classes are open for each possible demand scenario over all time periods. Major challenges include the curse of dimensionality (see Powell (2007)) and tremendous storage requirement of the dynamic controls. Promising techniques have been developed to cope with these challenges by approximating the value function in a way that a set of static controls can be efficiently retrieved. Several relevant models are presented in sequel.

The stochastic seat allocation model $SP^*(\mathbf{K}) = \max_{\mathbf{x}\in\mathcal{J}(\mathbf{K})} \sum_{j\in J} \mathbb{E}[r_j \min\{x_j, D_j\}]$ is widely known. It aggregates demand for the remaining time periods and aims to maximize the expected revenue by allocating available seats to each product. While this model is intuitive, it assumes a high-to-low fare arrival order (as it allows "cherry-picking" passengers) and yields a partitioned allocation that captures neither nesting nor upsell. Its continuous relaxation is known as the probabilistic nonlinear programming model, and its deterministic version is known as the deterministic linear programming model $DLP^*(\mathbf{K}) = \max_{\mathbf{x}\in\overline{\mathcal{J}}(\mathbf{K})} \sum_{j\in J} r_j \min\{x_j, D_j\}$, where $\overline{\mathcal{J}}(\mathbf{K})$ is the same as $\mathcal{J}(\mathbf{K})$ without the integral allocation requirements (see Talluri and van Ryzin (2004) for more details about $DLP(\mathbf{K})$). To incorporate demand stochasticity while preserving the simplicity of $DLP(\mathbf{K})$, Talluri and Van Ryzin (1999) propose a randomized linear programming model $RLP^*(\mathbf{K}) = \mathbb{E}[\max_{\mathbf{x}\in\overline{\mathcal{J}}(\mathbf{K})} \sum_{j\in J} r_j \min\{x_j, D_j\}]$. In practice, its expectation is numerically approximated using finitely many demand samples. For each of the demand samples, the primal solution is discarded, and only the dual solution is stored. The final policy is a bid-price policy with its bid-prices computed using the average dual solution over all the demand samples. It has been theoretically proven that the RLP bid-prices outperform the *DLP* bid-prices.

To capture nesting over multiple fare classes without upsell, Curry (1990) derives an itinerary-based allocation model, which yields a set of static protection levels for each itinerary. His model can be solved efficiently by a two-stage procedure and is optimal if bookings arrive in a low-to-high fare order. Let ξ be the number of remaining seats given to an itinerary. The model reads

$$IP^*(\mathbf{K}) = \max_{\mathbf{x}\in\mathcal{J}(\mathbf{K})} \left\{ \sum_{i\in I} R_{i|C_i|}(\mathbf{\Pi}_i, y_i) \middle| (\mathbf{\Pi}_i, y_i) \in \mathcal{N}(\mathbf{x}_i) \text{ for } i \in I \right\}$$

and

$$\begin{aligned} R_{ic}(\mathbf{\Pi}_i^{c-1}, \xi) = &\int_0^{\xi-\Pi_{ic-1}} \left[r_{ic} d_{ic} + R_{ic-1}\left(\mathbf{\Pi}_i^{c-2}, \xi - d_{ic}\right) f_{ic}(d_{ic}) \right] dd_{ic} \\ &+ \left(r_{ic}(\xi - \Pi_{ic-1}) + R_{ic-1}(\mathbf{\Pi}_i^{c-2}, \Pi_{ic-1}) \right) \int_{\xi-\Pi_{ic-1}}^{\infty} f_{ic}(d_{ic}) dd_{ic}, \end{aligned} \tag{1.2}$$

where $R_{i0} = 0$, $\Pi_{i0} = 0$, and $f_j(\cdot)$ is the demand density function for product $j = (i, c)$. The revenue function (1.2) is recursive and has a state space of protection levels and remaining empty seats. It collects revenue by accepting booking requests for class c and adjusts the remaining seats before proceeding to class $c - 1$. Note that since booking requests arrive in a low-to-high fare order, time index can be ignored. The discrete version of (1.2) can be found in Wollmer (1992).

All the optimization models discussed above are rather intuitive and well-studied. In the following sections, we discuss how to extend $IP(\mathbf{K})$ to capture customer upsell when demand is multinomial logit. A solution method is then proposed.

1.3 Itinerary-based Allocation Model with Nesting and Upsell

In this section, we develop a network model that captures both capacity nesting and customer upsell based on $IP(\mathbf{K})$. Furthermore, we also propose an equivalent stochastic formulation with a structure that allows us to develop an approximation algorithm.

For capacity nesting, a customer with its corresponding class closed may be given an empty seat from any lower classes. For customer upsell, it is the opposite. A customer may be willing to pay more to obtain an empty seat if its corresponding fare class is closed. The former is a choice of the airline with customers accepting the requests, and the later is a choice of the customer with a probability that is usually assumed multinomial logit (MNL). For the MNL demand model, the upsell probability of product j is determined by its attractiveness $a_j = exp(\beta_j^s s_j + \beta_j^r r_j)$, where β_j^s and β_j^r are the elasticities of the schedule and fare of product j, and s_j is the schedule quality which measures the attractiveness of the flight schedule. The upsell probability from class c to a higher class c' on itinerary i is computed based on the proportion of the attractiveness of the higher class, i.e. $p_{icc'} = a_{ic'}/(\sum_{l=1}^{c} a_{il} + a_{i|C_i|})$, where $a_{i|C_i|}$ is the attractiveness of all options not offered by the host airline. For more information about MNL, we refer the reader to Gallego et al. (2009).

For an itinerary i, let $U_{icc'}$ be a random variable corresponding to the number of upsells from class c to class c', let $u_{icc'}$ be its realization, let $\eta_{ic} = \sum_{c'>c} u_{icc'}$ be the number of accumulated upsells to class c from all lower classes $\{c + 1, \ldots, |C|\}$, let $\mathbf{p}_i(c) = (p_{ic1}, \ldots, p_{ic|C_i|})$ be the vector of upsell probabilities of class c with $p_{icc} = \cdots = p_{ic|C_i|-1} = 0$ and $p_{ic|C_i|}$ being the probability of not buying from the host airline (leaving our reservation system without making any bookings, and potentially buying from other

airlines). This definition is consistent with our definition of $a_{i|C_i|}$. For a given number of rejected bookings n, let $\mathbf{q}_{ic}(n, \mathbf{p}_i(c), c) = (U_{ic1}, \ldots, U_{ic|C_i|})$ be the vector of upsells of class c on itinerary i based on the multinomial probability distribution $\mathcal{B}(n, \mathbf{p}_i(c))$. Note that for each $\mathbf{u}_{ic}$, a realization of $\mathbf{U}_{ic}$, we have $\sum_{c' \in C_i} u_{icc'} = n$ and $\sum_{c' \in C_i} p_{icc'} = 1$, and $u_{ic|C_i|} \leq n$ is the number of customers not buying any products from the host airline. The itinerary-based allocation model with nesting and upsell is

$$U'^*(\mathbf{K}) = \max_{\mathbf{x} \in \mathcal{J}(\mathbf{K})} \left\{ \sum_{i \in I} V_{i|C_i|}(\mathbf{\Pi}_i, y_i, \mathbf{0}) \middle| (\mathbf{\Pi}_i, y_i) \in \mathcal{N}(\mathbf{x}_i) \text{ for } i \in I \right\},$$

and the revenue function is

$$V_{ic}(\mathbf{\Pi}_i^{c-1}, \xi, \boldsymbol{\eta}_i) = \begin{cases} \mathbb{E}\left[r_{ic} \min\{\xi - \Pi_{ic-1}, D_{ic} + \eta_{ic}\} \right. \\ \quad + V_{ic-1}\left(\mathbf{\Pi}_i^{c-2}, \xi - \min\{\xi - \Pi_{ic-1}, D_{ic} + \eta_{ic}\}, \right. \\ \qquad \left. \left. \boldsymbol{\eta}_i + \mathbf{q}_{ic}(D_{ic} - (\xi - \Pi_{ic-1} - \eta_{ic})^+, \mathbf{p}_i(c), c)\right)\right] & \text{if } \xi \geq \Pi_{ic-1}, \\ \mathbb{E}[V_{ic-1}(\mathbf{\Pi}_i^{c-2}, \xi, \boldsymbol{\eta}_i + \mathbf{q}_{ic}(D_{ic}, \mathbf{p}_i(c), c))] & \text{otherwise,} \end{cases} \tag{1.3}$$

where $\mathbf{0}$ is a vector of zeros of size $|C_i|$. The revenue for class c is computed based on the minimum of the available seats and the total demand (demand for class c plus upsells). While the remaining capacity is updated based on the number of accepted bookings $(\min\{\xi - \Pi_{ic-1}, D_{ic} + \eta_{ic}\})$, the total number of upsells is recorded when rejected bookings exist (i.e. $D_{ic} - (\xi - \Pi_{ic-1} - \eta_{ic})^+ > 0$, where the maximum operator indicates that upsells to class c must be first accommodated or discarded before the original demand for class c is considered). The main differences between $IP(\mathbf{K})$ and $U'(\mathbf{K})$ are that the vector of the observed upsells $\boldsymbol{\eta}_i$ is now part of the state space, and η_{ic} is added wherever demand for class c is present.

Ideally, rejected passengers should be recaptured at the moment that they are rejected. However, since value of time is not part of the model, upsold passengers are identical regardless which lower classes they originally belonged, and allocation level is first reduced by the number of upsold passengers, our model correctly accounts for upsells. To see this, suppose we reject 5 class-c' passengers, and 2 of them upsell to a higher class $c < c'$. We can immediately subtract 2 empty seats from class c, or we can subtract those 2 empty seats later after class-c passengers arrived and before accepting class-c passengers. Given the same allocation, these two cases produce the same revenue so long as subtracting empty seats from class c later does not change the revenue that we can collect from the 2 rejected passengers. This also holds with upsells from multiple lower classes, so long as the upsold passengers to class c are identical.

Proposition 1 shows an equivalent formulation of the problem that exposes its recursive structure.

Proposition 1. *Problem $U'(\mathbf{K})$ exhibits the following stochastic formulation:*

$$U^*(\mathbf{K}) = \max_{\mathbf{x}\in\mathcal{J}(\mathbf{K})} \sum_{i\in I} \mathbb{E}\left[Q_i(\mathbf{x}_i, \mathbf{D}_i)\right]. \tag{1.4}$$

The revenue function for each itinerary i is

$$Q_i(\mathbf{x}_i, \mathbf{d}_i) = \max_{\mathbf{z}\ \textit{integer}} \mathbb{E}\left[\sum_{c\in C_i} r_{ic} \min\left\{x_{ic} + z_{ic+1}, d_{ic} + \psi_{ic}\right\}\right] \tag{1.5}$$

subject to

$$z_{ic} = (x_{ic} + z_{ic+1} - d_{ic} - \psi_{ic})^+ \qquad c \in C_i \tag{1.6}$$

$$(U_{ic1}, \ldots, U_{ic|C_i|}) = \mathbf{q}_{ic}\left(d_{ic} - (x_{ic} + z_{ic+1} - \psi_{ic})^+, \mathbf{p}_i(c), c\right) \qquad c \in C_i \tag{1.7}$$

$$\sum_{c'=c}^{|C_i|-1} U_{ic'c} = \psi_{ic} \qquad c \in C_i \tag{1.8}$$

where expectation in (1.4) is taken over demand, and expectation in (1.5) is taken over upsells given a demand sample.

Proof. The proof is completed by mapping the remaining capacity and upsell between two problems, and exploring the recursive structures of (1.6) and (1.7). See Appendix A.1.1 for details. □

Problem $U(\mathbf{K})$ first maximizes the expected revenue by allocating seats to each product. Once the set of allocated seats and a demand sample are given, sales are maximized by utilizing empty seats and recapturing rejected customers from lower classes. Constraints (1.6) and (1.7) are the definitions of z_{ic} and $U_{icc'}$ for $c' \in C$. Note that if the expectation over upsell is ignored, the revenue function $Q_i(\mathbf{x}_i, \mathbf{d}_i)$ is similar to the objective function of $DLP(\mathbf{K})$ but with variable z_{ic+1} to account for the accumulated empty seats from lower classes and ψ_{ic} to handle realized upsells to class c. This alternative formulation of the problem separates the RM decision and sales processes in two steps for an easier and more intuitive interpretation of the RM problem, and more importantly, it has a structure that we can exploit to develop an algorithm which we discuss in the next section.

This alternative formulation is also applicable to $IP(\mathbf{K})$, whose corresponding equivalent formulation can be obtained by assuming continuous demand and dropping φ from the formulation of $U(\mathbf{K})$. Formally, the reformulated problem without upsell is

$$P^*(\mathbf{K}) = \max_{\mathbf{x}\in\mathcal{J}(\mathbf{K})} \sum_{i\in I} \mathbb{E}\left[\mathcal{S}_i(\mathbf{x}_i, \mathbf{D}_i)\right],$$

where the revenue function is

$$\mathcal{S}_i(\mathbf{x}_i, \mathbf{d}_i) = \max_{\mathbf{z} \text{ integer}} \sum_{c \in C_i} r_{ic} \min\{x_{ic} + z_{ic+1}, d_{ic}\}$$

$$z_{ic} = (z_{ic+1} + x_{ic} - d_{ic})^+ \qquad c \in C_i. \tag{1.9}$$

As upsell, the second level of stochasticity (the first level is demand), is dropped, the problem essentially becomes a two-stage stochastic programming problem. At the first stage, seats are allocated to each product subject to flight capacity. At the second stage, bookings are accepted based on the first stage allocation and a demand realization. Note that problem $P(\mathbf{K})$ is also valid for discrete demand, and we will show later in Section 1.5 that its corresponding nested allocation policy is asymptotically optimal when no upsells are allowed, and when its demand and capacity are scaled and normalized linearly with a factor that approaches infinity.

Let us refer to low-to-high fare arrival order as LH, high-to-low fare arrival order as HL, and random fare arrival order as R. Before we proceed further, let us summarize all the models we have introduced thus far in Table 1.1.

Table 1.1: Model Summary

Model	Description	Arrival Order	Integral	Nesting	Upsell
$SP(\mathbf{K})$	Stochastic seat allocation model	HL[a]	✓	✗	✗
$DLP(\mathbf{K})$	$SP(\mathbf{K})$ without integral allocation requirements	HL	✗	✗	✗
$RLP(\mathbf{K})$	$DLP(\mathbf{K})$ with random demand samples	HL	✗	✗	✗
$IP(\mathbf{K})$	Curry (1990)'s itinerary-based allocation model	LH	✓	✓	✗
$U'(\mathbf{K})$	Extended $IP(\mathbf{K})$ with upsell	LH	✓	✓	✓
$U(\mathbf{K})$	Stochastic programming formulation of $U'(\mathbf{K})$	LH	✓	✓	✓
$P(\mathbf{K})$	$U(\mathbf{K})$ without upsell	LH	✓	✓	✗

[a] Since the model "cherry-picks" passengers, the implicit assumption on the arrival order is HL.

The `Integral` column indicates if integral allocation requirement is imposed. The `Nesting` column indicates if the nesting nature of the capacity is being explicitly considered in the model, while `Upsell` indicates if the model captures upsells.

1.4 Solution Methodology

We employ the ADP framework described in Powell et al. (2004) to solve our problem. It is specifically designed for a two-stage stochastic problem with the following properties: the objective function is separable

in the first stage decision, stochastic information can be easily collected, and a subgradient to the objective function can be computed.

The idea is to iteratively approximate the complicated objective function by simple basis functions which can easily encode estimates of the true subgradient. Each iteration consists of two parts. First, we solve the first-stage problem with the basis functions from the previous iteration. Next, given the solution of the first-stage problem together with partially observed stochastic information, we solve the second-stage problem to obtain a new set of slopes to improve and update the slopes of the basis functions. As this procedure iterates and more information is observed, the original objective function can be approximated arbitrarily closely under some mild conditions (Powell et al. (2004) show convergence).

Beside some theoretical guarantees, there are two major practical benefits from applying this ADP framework: 1) the first-stage problem is often easier to solve when the objective function is replaced by some simple basis functions, and 2) as the problem is separable in the first-stage decision, the second-stage problem can often be parallelized. In today's multi-core computing environment, this parallelization feature can reduce our solution time significantly.

This ADP framework especially suits our problem as 1) the objective function (1.4) is separable in the first stage decision upon a slight modification shown below, 2) demand and upsells can be easily simulated, and 3) slopes can be estimated directly based on the recursive structures in (1.6) and (1.7). In our application, we use piecewise linear functions as the basis functions. To have the objective function (1.4) separable in the first-stage decision, we change our decision to allocating seats at the itinerary level. It is done by splitting problem $U(\mathbf{K})$ in two sub-problems and adding an auxiliary variable y_i to represent the number of seats allocated to each itinerary. Formally, the first sub-problem is

$$U^{sp1}(\mathbf{K}) = \max_{\mathbf{y}\in\mathcal{I}(\mathbf{K})} \sum_{i\in I} U_i^{sp2}(y_i),$$

and the second sub-problem is

$$U_i^{sp2}(y_i) = \max_{\mathbf{x} \text{ integer}} \left\{ \mathbb{E}\left[Q_i(\mathbf{x}_i, \mathbf{D}_i)\right] : \sum_{c\in C_i} x_{ic} = y_j \right\}.$$

A side benefit from this modification is that as the class-level allocation is decoupled from the network-level capacity constraints, we can obtain a class-level allocation and accommodate upsells locally and independently for each itinerary.

Let $S_i = \min_{f \in F_i}\{K_f\}$ be an upper bound on the number of seats that can be allocated to itinerary i, and for itinerary i and allocation level $s \in \{0, 1, ...S_i\}$, we define $v_{is} \approx U_i^{sp2}(s) - U_i^{sp2}(s-1)$ to be an estimate on the marginal revenue of allocating one more seat to itinerary i when $s-1$ seats have already been allocated. The ADP algorithm approximates $U_i^{sp2}(y_i)$ by a piecewise linear function $\mathcal{Q}_i(y_i) = \sum_{s=1}^{l} v_{is} + v_{is+1}(y_i - l)$ for a unique integer l satisfying $l \leq y_i < l+1$. Note that the function has at most S_i many breakpoints, and we assume $\mathcal{Q}_i(0) = 0$ (constant offsets can be removed from optimization problems, and thus, are irrelevant). With these piecewise linear functions, the ADP algorithm first solves $U^{sp1}(\mathbf{K})$ and produces an itinerary allocation level. For a given itinerary allocation level $\bar{y}_i$, an upsell heuristic (the core of the ADP algorithm, presented later) is run to estimate the true marginal revenue (sub-gradient) of $U_i^{sp2}(\cdot)$ at $\bar{y}_i$. The margin is, in turn, used to refine the accuracy of $\mathcal{Q}_i(\cdot)$, the approximating function to $U_i^{sp2}(\cdot)$. The complete ADP algorithm is presented in Algorithm 1.

Algorithm 1 ADP Algorithm to approximate $U^{sp1}(\mathbf{K})$

1: Initialize v_{is} for $s = 1, \ldots, S_i$ and $i \in I$.
2: **while** stopping criteria are not met **do**
3: Solve $\bar{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{I}(\mathbf{K})} \sum_{i \in I} \mathcal{Q}_i(y_i)$
4: **for all** $i \in I$ **do**
5: Run the upsell heuristic to obtain the marginal revenue $\hat{v}_i$.
6: $\bar{v}_{i\bar{y}_i} = (1 - \alpha_i) v_{i\bar{y}_i} + \alpha_i \hat{v}_i$.
7: Update stepsize α_i by bias-adjusted Kalman filter stepsize rule (see Powell, 2007, chap. 6).
8: $\mathbf{v}_i = \arg\min_{\boldsymbol{\delta}} \{\sum_{s=1}^{S_i} (\delta_{is} - \bar{v}_{is})^2 | \delta_{is+1} \leq \delta_{is} \text{ for } s = 1, \ldots, S_i\}$.
9: **end for**
10: **end while**
11: Compute the class-level allocation $\bar{\mathbf{x}}$ based on the last $\bar{\mathbf{y}}$ using the upsell heuristic.
12: **return** $\bar{\mathbf{x}}$

Step 1 of the ADP algorithm initializes the marginal revenue (slope) for each possible allocation level over all itineraries. Step 2 stops the algorithm when changes to the slopes are negligible. The while-loop consists of two parts. In the first part, step 3 solves $U^{sp1}(\mathbf{K})$ using piecewise linear functions $\{\mathcal{Q}_i(\cdot)\}$ and returns the optimal allocation level $\bar{y}_i$ for each itinerary $i \in I$. In the second part, steps 5 to 8 updates slops of $\mathcal{Q}_i(\cdot)$. Given $\bar{y}_i$, step 5 computes the marginal revenue $\hat{v}_i$ at $\bar{y}_i$ using the upsell heuristic that we developed to solve $U^{sp2}(\bar{y}_i)$. Step 6 updates the slope using $\hat{v}_i$ from step 5 while step 7 updates α_i, which is the stepsize for updating the slope. Note that $\boldsymbol{\alpha}$ is in fact state-dependent, but the dependency is dropped for ease of exposition. After the slope at $\bar{y}_i$ is updated, $\mathcal{Q}_i(\cdot)$ may not be concave. Step 8 then imposes concavity on $Q_i(\cdot)$ by finding the closest concave piecewise linear function, where the distance is measured

by the standard two-norm. An efficient projection algorithm can be found in Powell et al. (2004). The projection does not only guarantee that $\mathcal{Q}_i(\cdot)$ is concave, but it also allows the approximation model in step 3 to be linearized. The resulting linear programming problem is

$$\max_{\substack{\mathbf{y}\in\mathcal{I}(\mathbf{K})\\ \boldsymbol{\rho}}} \left\{ \sum_{i\in I}\sum_{s=1}^{S_i} v_{is}\rho_{is} : \sum_{s=1}^{S_i} \rho_{is} = y_i \text{ for } 0 \le \rho_{is} \le 1 \text{ for } s = 1,\ldots,S_i, i \in I \right\}.$$

With the well-approximated objective functions and the latest allocation level for each itinerary, steps 11 and 12 compute and return a class-level partitioned allocation, which can be used to construct a nested allocation policy according to $\mathcal{N}(\bar{\mathbf{x}})$.

We now elaborate further the upsell heuristic which estimates the marginal revenue of $U_i^{sp2}(\cdot)$ at $\bar{y}_i$ and produces a partitioned allocation to be returned by the ADP. The heuristic iteratively adjusts the existing partitioned allocation by moving seats from less-profitable classes to more-profitable classes (with the presence of upsells, it is not necessary from lower classes to higher classes). It first generates a set of demand samples, and finds the highest class having a positive allocation. Then, it iteratively reduces the number of allocated seats and reallocates the extracted seats to lower classes that can most profitably utilize the seats. If no profitable lower class is found or no more seat can be extracted, the algorithm repeats by finding the next highest class having a positive allocation. This procedure continues until the highest class having a positive allocation is also the lowest class. Let ζ_{ic}^k be the k^{th} demand sample for class c on itinerary i. The algorithm is described in Algorithm 2.

Algorithm 2 Upsell heuristic to approximate $U_i^{sp2}(y_i)$

Require: y_i.

1: Generate N demand samples $\{\zeta_{ic}^1, \ldots, \zeta_{ic}^N\}$ for $c \in C_i$.
2: Initialize $\mathbf{x}_i$ using EMSR-upsell algorithm.
3: Find $\hat{c}$ the highest class with a positive allocation.
4: **while** $\hat{c}$ is not the lowest class **do**
5: $x_{i\hat{c}} = x_{i\hat{c}} - 1$.
6: Apply the revenue estimation algorithm to compute revenue r and collect upsell information.
7: Find a lower class c' that yields the highest margin $\delta_{c'}$ using the margin estimation algorithm and the upsell information collected in the previous step.
8: $x_{ic'} = x_{ic'} + 1$.
9: **if** $\hat{c} = c'$ or $x_{i\hat{c}} = 0$ **then**
10: Find $\tilde{c}$ the next highest class with a positive allocation.
11: **if** class $\tilde{c}$ exists **then**
12: $\hat{c} = \tilde{c}$
13: **else**
14: **return** $\mathbf{x}_i, r + \delta_{c'}, \delta_{c'}$.
15: **end if**
16: **end if**
17: **end while**

Step 1 of Algorithm 2 generates demand samples for the algorithm to estimate the total revenue and marginal revenue of $U_i^{sp2}(\cdot)$ at y_i. Step 2 initializes the class-level allocation using an EMSR type algorithm modified to capture upsells. Step 3 finds $\hat{c}$ the highest class with a positive allocation. If $\hat{c}$ is not the lowest class, then step 5 subtracts a seat from class $\hat{c}$, and step 6 applies a revenue estimation algorithm to compute the base revenue, which is to be added to the highest revenue margin to yield the total revenue. Simultaneously, the revenue estimation algorithm also returns all information necessary for a margin estimation algorithm to find class c', which yields the highest revenue margin in step 7. After the extracted seat from class $\hat{c}$ is added to class c', step 10 finds from all lower classes the next highest class having a positive allocation. If such a class exists, the heuristic starts the next iteration with that class. Otherwise, step 14 returns the modified allocation, the associated total revenue, and the approximated marginal revenue of $U_i^{sp2}(\cdot)$ at y_i.

For completeness, we also briefly summarize the revenue estimation algorithm (Algorithm 5 in Step 6), the margin estimation algorithm (Algorithm 6 in Step 7), and the EMSR-upsell algorithm (Algorithm 7 in 2) with their details documented in Appendices A.2.1, A.2.2, and A.2.3 respectively.

The revenue estimation algorithm is an implementation-level verbatim copy of $Q_i(\mathbf{x}_i, \mathbf{d}_i)$. It takes the number of seats allocated to each class, the set of generated demand samples, and returns the average revenue

over all demand samples along with all rejection and upsell information necessary for the margin estimation algorithm to efficiently compute the seat margin. It heavily relies on the nested recursive structure of (1.6) and (1.7) in Proposition 1 to compute the revenue and extract the information directly.

The margin estimation algorithm recovers our single-seat allocation decision to provide a what-if margin. It requires the same inputs as those for the revenue estimation algorithm, together with the upsell information, as well as the class c' that one extra seat is adding to (in step 5 of Algorithm 2). It starts with checking if there exists a rejected upsell from any lower classes. If a rejected upsell is found, it returns the class-c' fare. Otherwise, if no upsells to class c' are rejected, for any higher classes $l = 1, \ldots, c'$, the algorithm searches for a rejected booking in class l and its corresponding upsell. If an upsell to a particular class l' in $\{1, \ldots, l-1\}$ is found, the algorithm returns $r_l - r_{l'}$ as if no upsell from class l ever exists because of nesting. If a rejected booking exists but does not result in an upsell, the algorithm returns the class-l fare.

The EMSR-upsell algorithm combines a simple fare-adjusted criterion in Gallego et al. (2009) and the algorithm in Curry (1990) to efficiently approximate a set of protection levels that accounts for upsell. Additional cares are given to update the marginal revenue and determine the set of protection levels. Although the choice-based EMSR algorithm in Gallego et al. (2009) is showed to perform better than this EMSR-upsell algorithm. However, it is chosen because 1). It efficiently provides a slope at any feasible allocation level to initialize our ADP algorithm, and 2). As observed in Gallego et al. (2009), the fare-adjusted criterion numerically yields a set of protection levels that tends to reserve more seats to higher classes. This is desirable as the upsell heuristic (Algorithm 2) iteratively and profitably reallocates seats from higher classes to lower classes.

1.5 Asymptotic Property of Nested Allocation Policy

In this section, we complement the asymptotic optimality analysis of Cooper (2002) for partitioned allocation policies by showing the asymptotic optimality of the optimal nested allocation policy without upsell. Our focus is on problem $P(\mathbf{K})$, and as a side product, we show the bounding property of different approximation models to $v_{|T|}(\mathbf{K})$. Specifically, we show in Lemma 1 that $\overline{P}(\mathbf{K})$, the version of the nested allocation model with deterministic demand, yields the same objective value as $\overline{SP}(\mathbf{K})$, the version of the stochastic seat allocation model $SP(\mathbf{K})$ with deterministic demand. In Lemma 2, we show that the nested allocation policy of $P(\mathbf{K})$ is provably better than the partitioned allocation policy of $SP(\mathbf{K})$ even when arrivals of

different classes are randomly ordered. With these two observations, we can directly apply the results from Cooper (2002) to obtain the asymptotic optimality of the nested allocation policy of $P(\mathbf{K})$.

Lemma 1. *We have* $\overline{SP^*}(\mathbf{K}) = \overline{P^*}(\mathbf{K})$.

Proof. The proof is completed by substituting stochastic demand by its expectation, and checking feasibility of the optimality solution to each problem. See Appendix A.1.2 for details. □

Lemma 1 states that we do not need nesting if demand is deterministic. Let us denote by $\mathcal{R}^{\pi}(\mathbf{D})$ the revenue obtained by implementing the allocation policy constructed based on the optimal solution to problem π given demand sample $\mathbf{D}$. Note that $\mathbb{E}\mathcal{R}^{\pi}(\mathbf{D})$ is different from π^*, which is simply the objective value of problem π with various assumptions on the original RM problem. We can achieve $\mathbb{E}\mathcal{R}^{\pi}(\mathbf{D}) = \pi^*$ only if all the underlying restrictive assumptions are applicable when the allocation policy is implemented and revenue is collected. Such examples include $\mathbb{E}\mathcal{R}^{v_{|T|}(\mathbf{K})}(\mathbf{D})$ and $\mathbb{E}\mathcal{R}^{SP(\mathbf{K})}(\mathbf{D})$ for any arrival order of $\mathbf{D}$, and $\mathbb{E}\mathcal{R}^{P(\mathbf{K})}(\mathbf{D})$ when the arrival order of $\mathbf{D}$ is in fact low-to-high fare.

Lemma 2. *We have* $\mathbb{E}\mathcal{R}^{P(\mathbf{K})}(\mathbf{D}) \geq \mathbb{E}\mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}) \geq \mathbb{E}\mathcal{R}^{\overline{SP}(\mathbf{K})}(\mathbf{D}) \geq \mathbb{E}\mathcal{R}^{DLP}(\mathbf{D})$ *for any arrival order.*

Proof. The proof is based on the observations that the high-to-low fare arrival order yields a higher revenue than the low-to-high fare arrival order; applying the nested allocation policy under the worst arrival order (low-to-high fare) yields a higher revenue than applying the partitioned allocation policy, which is arrival-order independent, and the fact that the partitioned allocation policy from $SP(\mathbf{K})$ is optimal over all partitioned allocation policies under stochastic demand. See Appendix A.1.3 for details. □

Proposition 2. *We have* $DLP^*(\mathbf{K}) \geq \overline{SP^*}(\mathbf{K}) = \overline{P^*}(\mathbf{K}) \geq v_{|T|}(\mathbf{K}) \geq \mathbb{E}\mathcal{R}^{P(\mathbf{K})}(\mathbf{D}) \geq \mathbb{E}\mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}) \geq \mathbb{E}\mathcal{R}^{\overline{SP}(\mathbf{K})}(\mathbf{D}) \geq \mathbb{E}\mathcal{R}^{DLP(\mathbf{K})}(\mathbf{D})$ *for any arrival order.*

Proof. The first inequality is clear, since $DLP(\mathbf{K})$ is the same as $\overline{SP}(\mathbf{K})$ except for the integral allocation requirements. Inequality $\overline{SP^*}(\mathbf{K}) \geq v_{|T|}(\mathbf{K})$ is due to the fact that the number of accepted passengers for each class by using the dynamic policy of $v_{|T|}(\mathbf{K})$ is a feasible solution to $\overline{SP}(\mathbf{K})$, and lastly, inequality $v_{|T|}(\mathbf{K}) \geq \mathbb{E}\mathcal{R}^{P(\mathbf{K})}(\mathbf{D})$ follows from the optimality of $v_{|T|}(\mathbf{K})$. The proof is then completed by applying Lemma 1 and Lemma 2. □

With this bounding results, we can then directly apply Proposition 2 in Cooper (2002) to obtain asymptotic optimality of the nested allocation policy of $P(\mathbf{K})$. We denote by $D^k_{jt} = kD_{jt}$ the k-time scaled

demand for product j at time t, and by $v^k_{|T|}(\mathbf{K})$ the optimal DP with k-time scaled demand. We call a quantity normalized if it is divided by k. We show the asymptotic optimality result next.

Proposition 3. *If the normalized k-time linearly-scaled arrivals converge in distribution to their unscaled means, i.e. $\frac{1}{k}\sum_{t\in T} D^k_{jt} \xrightarrow{\mathcal{D}} \sum_{t\in T}\mathbb{E}D_{jt}$, the nested allocation policy from $P(k\mathbf{K})$ with k-time scaled demand is asymptotically optimal as $k\to\infty$.*

Proof. By Proposition 2, we have

$$\mathbb{E}\mathcal{R}^{DLP(k\mathbf{K})}(\mathbf{D}^k)/v^k_{|T|}(k\mathbf{K}) \le \mathbb{E}\mathcal{R}^{P(k\mathbf{K})}(\mathbf{D}^k)/v^k_{|T|}(k\mathbf{K}) \le 1.$$

From Proposition 2 in Cooper (2002), it follows that $\mathbb{E}\mathcal{R}^{DLP(k\mathbf{K})}(\mathbf{D}^k)/v^k_{|T|}(k\mathbf{K}) \xrightarrow{k\to\infty} 1$, and in turn, we have $\mathbb{E}\mathcal{R}^{P(k\mathbf{K})}(\mathbf{D}^k)/v^k_{|T|}(k\mathbf{K}) \xrightarrow{k\to\infty} 1$ as required. □

Proposition 3 ensures that if both demand and capacity are sufficiently large and classes are well-segmented such that upsells are unlikely, the nested allocation policy of $P(\mathbf{K})$ performs similarly to the optimal DP, and hence, provides a performance guarantee.

1.6 Computational Experiments

In Section 1.4, we have discussed the ADP algorithm and a set of heuristics for solving the network RM problem with nesting and upsell. In this section, we split our discussion in two parts. The first part focuses on the performance of the upsell heuristic (Algorithm 2), and the second part is devoted to the ADP algorithm (Algorithm 1). The reason to analyze the upsell heuristic separately is that the heuristic by itself is the most important part of the ADP algorithm. It is the part that captures both nesting and upsell, and returns a seat margin for the approximating function and a class-level allocation for performance evaluation. The second part demonstrates the performance of the ADP algorithm. Both parts include simulation details and output comparisons.

In the first part, the solution quality of the upsell heuristic is benchmarked against the solution qualities of the algorithm in Wollmer (1992), the choice-based EMSR algorithm in Gallego et al. (2009), and the optimal dynamic programming algorithm with upsell (1.3). The algorithm in Wollmer (1992) is developed directly based on the optimality condition of the revenue function (1.2) assuming discrete demand and no

upsell. It allows us to observe the degree of revenue improvement by incorporating upsell. The choice-based EMSR algorithm in Gallego et al. (2009) is a heuristic developed based on a modified optimality condition of (1.2). It efficiently captures upsells and outperforms most EMSR-type algorithms. It allows us to compare our algorithm to the best algorithm in class (single-leg based and efficient). The optimal dynamic programming algorithm with upsell is used to find the optimal set of static protection levels under the low-to-high fare arrival order assumption and to provide optimality gaps. It is based on evaluating all possible sets of the protection levels using (1.3) over a large set of demand samples. At the end, we also empirically demonstrate the accuracy of the marginal revenue returned by the upsell heuristic and show the effect of the projection operation in step 8 of the ADP algorithm before discussing the solution quality of the ADP at the network level.

In the second part, we run the ADP algorithm and benchmark its allocation policy against the RLP bid-price policy on a real airline network with 136 flights, 309 itineraries, and 31 reading days. In the absence of a scalable method that simultaneously incorporates nesting, upsell, demand stochasticity, and network information, we select RLP as it captures demand stochasticity and network information while providing scalability, reliable performance, and some theoretical guarantees (see Talluri and Van Ryzin (1999) and Topaloglu (2009)). We test our proposed allocation policy over various demand multipliers to measure the effect of the network fill rate and the sensitivity of the allocation policy to inaccurate upsell probabilities.

Figure 1.1 summarizes the architecture of the simulation module that evaluates control policies. It illustrates the flow of the simulation for the first reading day. At the beginning of the simulation, mean demand and flight capacity are queried from the database. While the flight capacity is fed to the optimization module, the mean demand is scaled and randomized. The resulting mean demand is used to generate Poisson demand sample paths for the simulation and optimization engines to compute and evaluate the control policies. The order of the arrivals between two reading days is randomized before any control policy is applied. Note that the simulation does not assume the low-to-high fare arrival order that our model assumes. Hence, it provides a realistic and fair comparison to the RLP bid-price policy which assumes random arrival orders. After the control policy is applied to accept and reject a booking, the corresponding upsell (if exists) is then generated to consume an empty seat from one of the higher classes. In the end, the revenue is computed based on the total number of bookings accepted for each class, and the flight capacity is updated before moving to the next reading day.

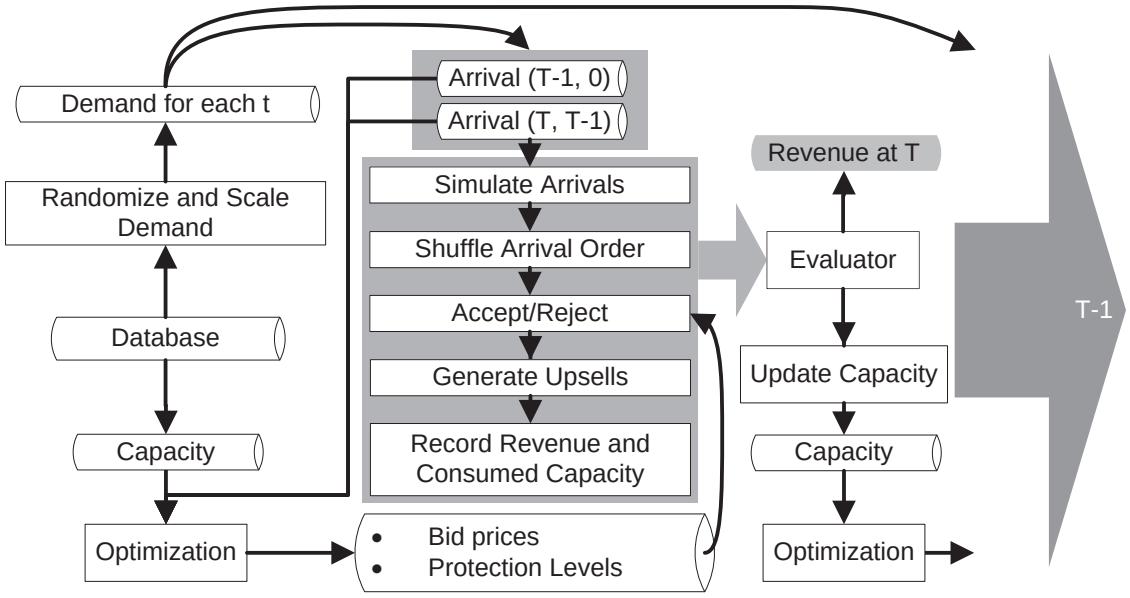

Figure 1.1: Flow chart of simulation at time T, the first reading day.

Let K^{sim} and K^{opt} be the number of demand sample paths generated for simulation and optimization respectively. For all single-leg simulation experiments, we follow the simulation settings of Gallego et al. (2009) by setting both K^{sim} and K^{opt} to $100,000$ to eliminate the effect of the standard error. Furthermore, all examples in Gallego et al. (2009) as well as some constructed cases are tested. For each itinerary, all algorithms being tested allocate remaining seats, e.g. $\xi - \Pi_{|C_i|-1}$, to the lowest class. For all network simulation experiments, due to the large amount of parameters we test and the long running time involved, we restrict K^{sim} and K^{opt} to 100 and 50 respectively. Although the number of simulation samples for simulation is relatively small, our results lead to insightful conclusions, which are all valid under a 5% significant level.

All simulation experiments have been run on a cluster of 50 servers. Each server has two 3.2 GHz Intel(R) Xeon(TM) CPUs and 6GB of memory, yet only up to 3GB were used due to our 32 bits limitation. IBM ILOG Cplex is used to solve the approximated network problem (step 3 of the ADP algorithm). After the allocation level for each itinerary is determined, the upsell heuristic is parallelized by itinerary over all available processors.

1.6.1 SINGLE LEG COMPARISON

We evaluate the solution quality of the upsell heuristic (Algorithm 2) by comparing it with the solution qualities of the algorithm in Wollmer (1992), the choice-based EMSR algorithm in Gallego et al. (2009), and the optimal dynamic programming with upsell. Let us abbreviate the algorithm to compute protection levels in Wollmer (1992) by `EMSR-w`, the choice-based EMSR algorithm in Gallego et al. (2009) by `EMSR-cb`, the optimal DP with upsell by `DP`, and the upsell heuristic by `EMSR-d`, where `d` refers to the dynamic nature of the algorithm. All examples in Gallego et al. (2009) as well as some constructed examples are tested.

Let α be the multiplier for both the fare and schedule quality, let γ be the margin scaler, let λ be the total demand, and let $|C|$ be the class to represent the option not buying from the host airline. To construct the examples, we set the fare for each class to $r_c = \alpha exp(\gamma(|C| - c + 1)/|C|)$ and schedule quality to $s_c = \alpha exp(\gamma(|C| - c + 1)/(2|C|))$. The fare and schedule quality for the not-buying option are the average fare and average schedule quality over all classes. These functions are selected in a way that the margin increases with the fare class, and the resulting fares closely match the real world data provided. For all constructed examples, the elasticities of fare and schedule are set to be $\beta_j^r = -0.0035$ and $\beta_j^s = 0.005$ respectively, which are the same settings in Gallego et al. (2009). Table 1.2 shows simulation settings for two/three/four/five-class examples, where the first three cases are taken from Gallego et al. (2009), and the fourth case is constructed based on $\alpha = 100$ and $\gamma = 0.6$. The first row refers to the number of classes in each example. The second row shows the elasticities of the schedule and price. The row after is the number of total booking requests. The remaining rows are fares and schedule qualities for different classes. Recall that the last class represents the not-buying option.

Table 1.2: Simulation settings for two/three/four/five-class examples

$\|C\|$	2		3		4		5	
β_j^r/β_j^s	-0.005	0.005	-0.0035	0.005	-0.0035	0.005	-0.0035	0.005
λ	26.67		25		50		20	
Class	fare	schedule	fare	schedule	fare	schedule	fare	schedule
1	1000	1200	1000	200	1000	400	2008.55	448.17
2	800	1000	800	200	900	300	1102.32	332.01
3	900	1100	500	200	600	300	604.96	245.96
4			1100	500	500	300	332.01	182.21
5					1000	600	182.21	134.99
6							846.01	268.67

Optimality gaps are summarized in Figure 1.2. For 2 and 3 classes, both EMSR-d and EMSR-cb perform similarly to DP with optimality gaps smaller than 1%. The solution quality of EMSR-w first deteriorates and is gradually improved as the number of available seats increases. For 4 classes, EMSR-d starts to outperform EMSR-cb when the number of seats available is higher than 16, and its optimality gap is still less than 1% while the optimality gap for EMSR-cb is about 2% in several occasions. For 5 classes, the gaps for both EMSR-w and EMSR-cb are significantly widened. This illustrates the drawback of estimating optimal protection levels based only on simple probability statements that cannot fully describe the dynamic of upsell.

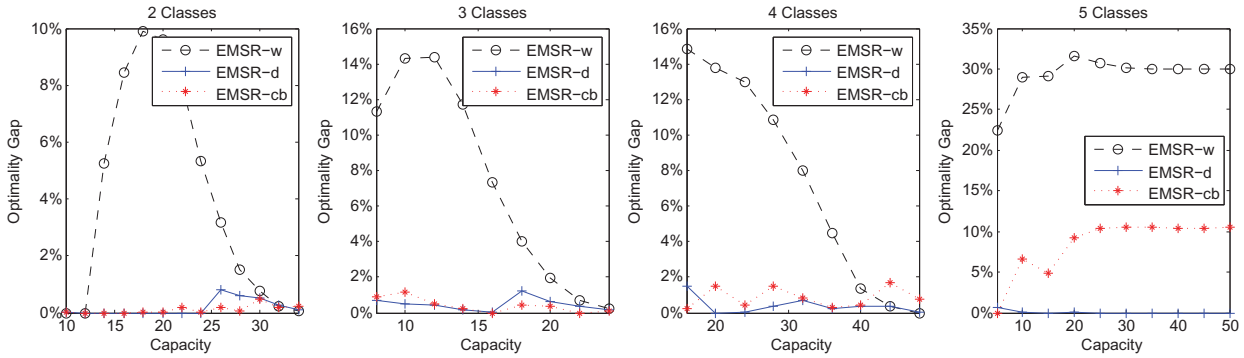

Figure 1.2: Optimality gap comparisons for two/three/four/five classes.

Figure 1.3 shows the running time as a portion of the running time of DP. It is clear that the relative running time of EMSR-d generally decreases as the number of classes increases. However, it sharply increases in the two-class example after the point where capacity and demand level meet. The reason is that EMSR-d initializes the allocation by the EMSR-upsell algorithm (Algorithm 7), which reallocates more seats to higher classes before moving the seats one-by-one to lower classes that are stochastically more profitable. This incurs extra overhead and algorithmic operations, and hence, slows down the algorithm. Furthermore, it is also due to the fact that running DP for the two-class problem is relatively inexpensive and hence, the gap is widened further. However, such an increase of running time gradually diminishes in the number of classes, since the running time of DP exponentially increases.

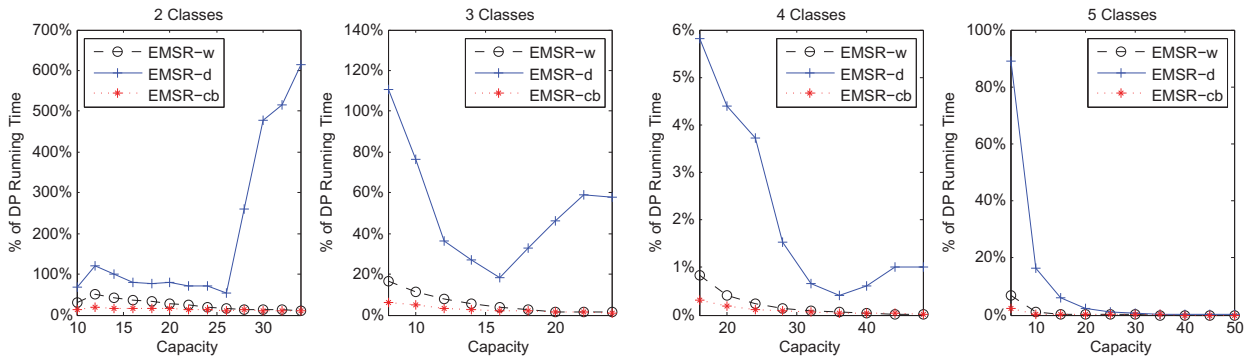

Figure 1.3: Percentage of the running time of DP for two/three/four/five classes from left to right, smaller the better.

In addition to the five cases discussed, extra simulation experiments were conducted for other cases that we constructed based on the method we described in the beginning of Section 1.6. We use EMSR−d instead of DP as the base to compute the optimality gaps, as our primary focus now is on how well our algorithm performs relative to other approximation algorithms. We tested $\alpha \in \{100, 200, 300\}$, $\gamma \in \{2, 2.5\}$, $C \in \{2, 3, \ldots, 10\}$, $\lambda \in \{10, 20, 30, 40, 50\}$, and capacity in $[5, 10, \ldots, 50]$. Average relative optimality gaps based on EMSR−d are given in Table 1.3, and the percentages of running time increase are given in Table 1.4.

In general, EMSR−d outperforms both EMSR−w and EMSR−cb up to 26% and 24% respectively when the number of classes increases, yet the percentage of the running time also increases as the running times for EMSR−w and EMSR−cb are fairly constant regardless the number of fare classes and demand magnitude. Note that when the number of bookings increases (more upsells indirectly), the optimality gap of EMSR−w is larger while the optimality gap of EMSR−cb becomes smaller. It suggests that EMSR−cb indeed captures upsells and outperforms EMSR−w when demand is not too low. Interested reader is referred to Appendix A.1 for the average running time of EMSR−d in second and the average demand factor (demand-to-capacity ratio) for different numbers of classes and total demand.

Table 1.3: Average relative optimality gap based on EMSR-d.

λ	10		20		30		40		50	
$\|C\|$	EMSR-w	EMSR-cb	EMSR-w	EMSR-cb	EMSR-w	EMSR-cb	EMSR-w	EMSR-cb	EMSR-w	EMSR-cb
2.0	1.33%	0.09%	3.90%	0.12%	6.98%	0.17%	9.04%	0.10%	11.79%	0.21%
3.0	22.49%	0.21%	22.15%	0.35%	23.11%	0.80%	23.63%	1.16%	23.63%	1.44%
4.0	21.43%	3.62%	22.42%	3.98%	22.62%	4.36%	23.39%	4.53%	24.67%	4.77%
5.0	20.82%	10.62%	22.48%	10.10%	23.53%	9.32%	24.03%	7.97%	24.94%	6.81%
6.0	20.81%	14.51%	22.95%	13.86%	23.68%	12.30%	25.29%	10.87%	25.81%	9.04%
7.0	21.25%	17.45%	22.82%	16.41%	24.07%	14.50%	25.02%	12.25%	26.06%	9.95%
8.0	20.62%	23.52%	22.47%	21.67%	23.95%	19.49%	25.03%	16.79%	25.95%	13.76%
9.0	20.55%	19.43%	22.53%	18.21%	24.18%	16.24%	24.96%	13.74%	26.02%	11.28%
10.0	20.22%	24.19%	22.38%	22.59%	24.18%	20.18%	25.39%	17.44%	26.02%	14.50%

Table 1.4: Average percentage of running time based on EMSR-d

λ	10		20		30		40		50	
$\|C\|$	EMSR-w	EMSR-cb	EMSR-w	EMSR-cb	EMSR-w	EMSR-cb	EMSR-w	EMSR-cb	EMSR-w	EMSR-cb
2.0	3.64%	1.77%	5.50%	1.55%	7.78%	2.24%	9.75%	2.68%	12.02%	3.12%
3.0	8.82%	7.03%	6.88%	4.66%	5.51%	2.87%	6.69%	2.88%	7.09%	3.07%
4.0	5.26%	4.65%	4.09%	2.81%	4.26%	1.98%	4.82%	2.24%	6.36%	2.47%
5.0	3.81%	3.30%	3.31%	2.17%	4.03%	1.81%	4.45%	1.78%	4.74%	1.75%
6.0	3.01%	2.70%	2.93%	1.87%	3.29%	1.51%	4.22%	1.54%	4.66%	1.53%
7.0	2.57%	2.50%	2.63%	1.71%	2.97%	1.38%	3.46%	1.29%	4.42%	1.43%
8.0	2.25%	2.23%	2.47%	1.60%	2.90%	1.35%	3.33%	1.24%	4.31%	1.36%
9.0	2.14%	2.28%	2.33%	1.58%	2.61%	1.27%	2.95%	1.14%	3.49%	1.14%
10.0	2.01%	2.19%	2.27%	1.58%	2.57%	1.29%	2.98%	1.16%	3.35%	1.12%

In order to closely approximate (1.5) with piecewise linear functions, we need to accurately estimate marginal revenue at each itinerary allocation level. Figure 1.4 illustrates how accurate the marginal revenues are estimated by EMSR-d in two examples which are the most representative given the data that we have. For each example, demand for the not-buying option is two times of the total demand, and the attractiveness of a fare class is set to be the magnitude of its corresponding demand. The figure on the left displays marginal revenue curves generated based on DP, EMSR-w, and EMSR-d for a four-class example with demand in {20, 15, 10, 5} and revenue in {100, 250, 500, 800}. It shows that the marginal revenue curve generated based on EMSR-d collides with that of DP, while the margins from EMSR-w are significantly different. The figure on the right similarly shows the marginal revenue curves obtained from an example with fifteen classes selected from a real world data set. Its demand is {2, 1, 24, 6, 10, 6, 15, 27, 2, 12, 8, 9, 3, 4, 23} and the revenues are {19.89, 22.13, 29.49, 29.78, 32.11, 33.78, 44.49, 51.98, 56.34, 62.52, 74.27, 128.85, 135.05, 170.71, 272.26}. We did not include DP as it is no longer tractable. Instead, we focus on how the projection operation from step 8 of the ADP algorithm changes the marginal revenues. The projected marginal revenue

curve is denoted by `pjEMSR-d` in the figure. It is clear that the marginal revenue curve is not monotonic, and hence, the corresponding revenue curve is not concave. After being projected, the marginal revenue curve becomes monotonic while many of the original margins are preserved. In summary, when the number of classes is small, `EMSR-d` accurately estimates the marginal revenues, and the projection operation can be safely applied to expedite the convergence of the ADP algorithm without significantly altering the original revenue margins returned by `EMSR-d`.

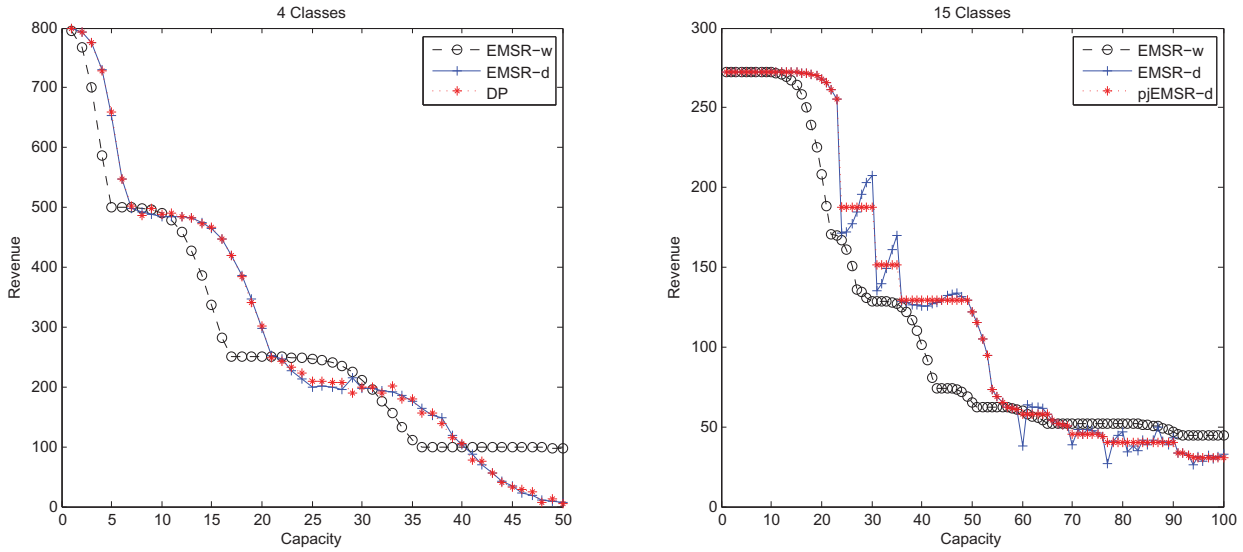

Figure 1.4: The marginal revenue curves for a four-class example (left) and a fifteen-class example (right).

1.6.2 Medium-Size Airline Network

In this section, we evaluate the performance of the protection levels returned by the ADP Algorithm using a medium airline network based on a real world data set. Table 1.5 summarizes the airline network, which has 136 flights, 309 itineraries, 31 reading days, 10.5 fare classes on average for each itinerary, and an average demand ratio[a] of 80%. We first present the simulation settings and implementation details, and conclude this section with simulation results.

Table 1.5: Summary of the medium airline network

No. of flights	136	Min. demand factor	3%
No. of itineraries	309	Avg. demand factor	80%
No. of reading days	31	Max. demand factor	240%
Avg. No. of Classes	10.5		

IMPLEMENTATION DETAILS ON THE ADP ALGORITHM

To run the ADP Algorithm, we need to initialize the marginal revenues properly and set up appropriate stopping criteria to prevent the algorithm from stalling without significantly trading off the solution quality. In our implementation, marginal revenues are initialized based on `EMSR-w`. It is mainly used for its efficiency as the marginal revenue for each possible itinerary allocation level has to be computed. If the upsell heuristic is used instead, running time will be excessively long without improving the solution quality significantly. The reason is that initial marginal revenues, if it is not totally inaccurate, do not significantly affect the final solution, and running several more iterations of the ADP algorithm to improve the solution is considerably less expensive. The ADP Algorithm is stopped if the current revenue is in $[\mu \pm 0.001\sigma]$, where μ and σ are the average revenue and standard deviation computed based on the last 30 revenue points. This stopping criterion guarantees that a large shift in revenue is probabilistically unlikely. To expedite the algorithm, we cease learning (updating marginal revenues) for an itinerary if the difference between its revenue from the last iteration and the revenue from the current iteration are less than 1% of its average revenue over the last 10 iterations.

SIMULATION SETTINGS

To evaluate the solution quality of the nested allocation policy under multiple demand scenarios, we randomize and scale the mean demand. To be more specific, the mean demand is randomized by a normal distribution and scaled by a multiplier. The resulting mean serves as the mean to generate Poisson demand sample paths for both simulation and optimization. The standard derivation is selected to be a multiple of the mean. Denoting the demand mean by $\overline{D}_{ict}$, the randomized demand is $D_{ict} \sim Round(\mathcal{N}ormal((1+m_1)\overline{D}_{ict}, (\overline{D}_{ict}/3)^2))$, where $m_1 \in \{-0.4, -0.2, 0, 0.2, 0.4\}$ is the demand multiplier (see Appendix A.2 for the corresponding demand factors), and the divisor 3 is the scaling factor of the standard deviation in order to match the original mean value, i.e. about 99.7% of the random demand falls into the interval of $[\overline{D}_{ict} \pm \overline{D}_{ict}]$. We regenerate if the realized demand is negative.

In reality, upsell probabilities are difficult to estimate due to data censorship. It is important to examine how forecasting error on upsell affects the performance of the nested allocation policy. Toward this end, we use two different sets of upsell probabilities, one for simulation, and one for optimization (when

[a] Demand-to-capacity ratio over all flights.

upsell information is generated in Algorithm 2). In both sets, the upsell probability is computed based on $p^j_{icc'} = m^j_2 \mathbb{E}D_{ic'} / \sum_{l=1}^{c-1} \mathbb{E}D_{il}$ for $c, c' \in C_i$, where $j \in \{simulation, optimization\}$ and m^j_2 is the upsell probability multiplier. If $m^j_2 = 0$, no upsell occurs. If $m^j_2 = 1$, no not-buying option exists. The values of m^j_2 are selected in $[0, 0.1, \ldots, 0.9]$.

We benchmark our allocation policy against the RLP bid-price policy. The number of demand sample paths for simulation is 100 across the entire booking period, the number of demand samples generated per ADP iteration is 50, and 50 demand samples are generated for the RLP.

DISCUSSIONS

We first discuss the case when demand varies and upsell probabilities are estimated accurately, i.e. the upsell probabilities for simulation and optimization are the same. The results are summarized in Figure 1.5. Each series corresponds to a demand multiplier ranging from -0.4 to 0.4 with an increment of 0.2. The figure shows in general that when demand increases, the percentage of revenue improvement increases over all scales of upsell probabilities, and is especially prominent in the central region where the upsell probability multiplier is in $[0.3, 0.7]$. We also observe that the improvement is similar for different demand multipliers. It suggests that the improvement is robust to the magnitude of the demand. The difference can be as much as 5% between the lowest (-0.4) and highest (0.4) demand multipliers. When the upsell probability multiplier increases, the percentage of revenue improvement increases in a convex manner. It suggests that the ability to capture upsell is vital when there are considerable amount of upsells.

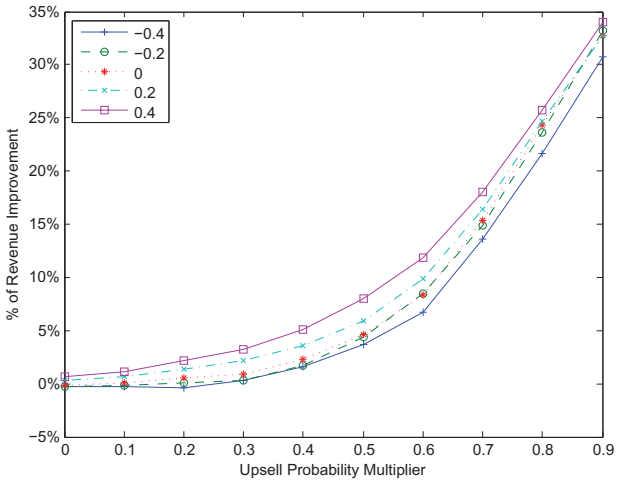

Figure 1.5: Percentage of revenue improvement against upsell probability multiplier when upsell probabilities are forecasted accurately.

Figure 1.6 and 1.7 show the percentage of revenue improvement for all tested combinations of $m_2^{simulation}$ and $m_2^{optimization}$ when $m_1 = 0$, i.e. demand is not scaled. The results are similar for other demand multipliers. See Table A.3 in Appendix A.3 for the exact numerical values. Figure 1.6 shows simulation results with each series corresponding to one value of $m_2^{simulation}$ in $\{0, 0.1, 0.2, 0.3, 0.4\}$. Overall, each improvement curve slowly increases until it is at its peak when $m^{simulation} = m^{optimization}$ and gradually decreases afterward. The largest improvement is about 2.29% when $m_2^{simulation} = m_2^{optimization} = 0.4$. The declining rate is faster when $m_2^{simulation}$ is small. The figure also shows that when $m_2^{optimization} \leq m_2^{simulation}$, revenue improvement is almost guaranteed, except for the case when upsell does not exist, e.g. $m_2^{simulation} = 0$. It signifies that it is better to underestimate the upsell probabilities when applying the ADP algorithm. Figure 1.7 shows simulation results when $m_2^{simulation}$ is in $[0.5, 0.6, 0.7, 0.8, 0.9]$. The situation is the opposite: overestimating upsell probabilities provides better results than RLP, and the revenue improvement can be as high as 33% when there is a 90% chance a rejected customer will upsell, and the upsell probability is estimated accurately. The reason for such an opposite behavior can be contributed to both the cascading effect of the upsell and the suboptimal nature of the upsell heuristic. When $m_2^{simulation} \geq 0.5$, every rejected booking is more likely to upsell than opting for the not-buying option. It results in pushing more low-class demand upward when $m_2^{simulation}$ increases. This phenomenon is particularly obvious when there exists many classes. Since the effect is accumulative starting from the lowest class, having additional seats for higher classes resulting from overestimating the actual upsell ($m_2^{optimiation} \geq m_2^{simulation}$) becomes beneficial. On the heuristic side, although the upsell heuristic captures upsells, it seems to underestimate the number of seats required when there is such an upsell-cascading effect. Another possible explanation is that RLP bid-price policy performs relatively undesirably when there exists a large amount of upsells.

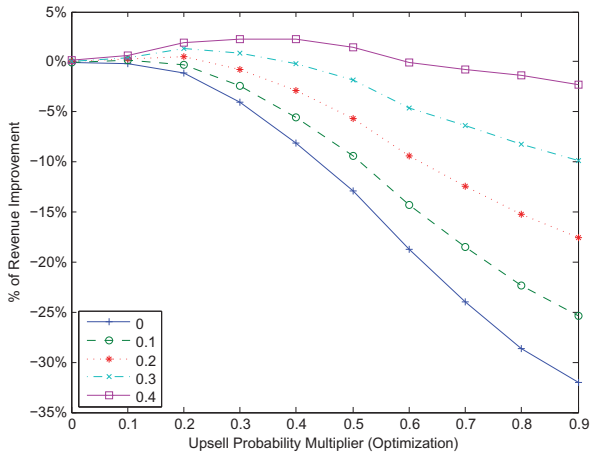

Figure 1.6: Percentage of revenue improvement against $m_2^{optimization}$ when $m_1 = 0$ and $m_2^{simulation} \leq 0.4$

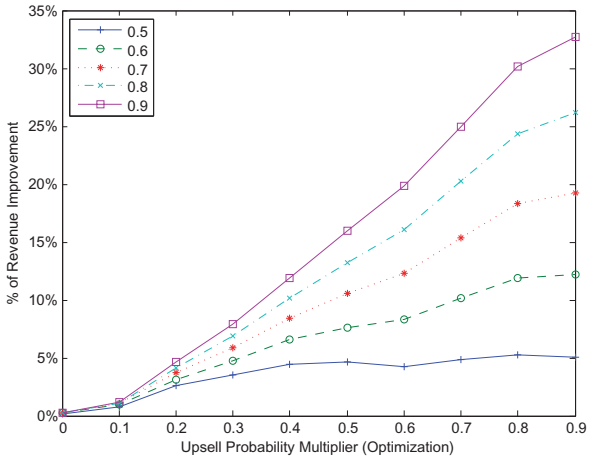

Figure 1.7: Percentage of revenue improvement against $m_3^{optimization}$ when $m_1 = 0$ and $m_3^{simulation} \geq 0.5$

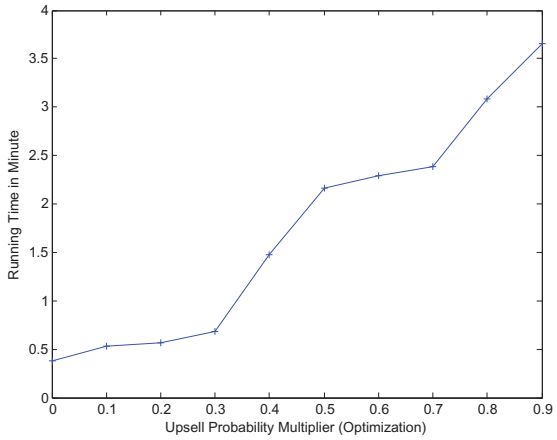

Figure 1.8: Average running time of the ADP algorithm in minute.

Figure 1.8 presents the average running time of the ADP algorithm. It shows that the algorithm takes longer to estimate a set of protection levels when the upsell probability for optimization increases. The running time stretches from less than 30 seconds to about 4 minutes. The reason for such an increase in the running time is that when more upsells are available, the upsell heuristic needs more enumerations to adjust the seat allocations and to compute the required margins for the piecewise linear functions in the ADP algorithm. On the other hand, the running time of the RLP is negligible, and hence, is not reported. Nonetheless, solving over a medium airline network in 4 minute is an acceptable practice.

1.7 CONCLUSION

Despite the prevalence of the bid-price policy, we aim to capture capacity nesting and customer upsell by extending an itinerary-based nesting model proposed by Curry (1990). We show that the itinerary-based nesting model can be formulated as an equivalent stochastic programming problem, which allows us to adapt an ADP framework to efficiently approximate the originally complicated objective function. We also derive an upsell heuristic based on the recursive structure of the problem, and integrate the heuristic into the ADP algorithm to solve the network RM problem over a medium airline network using a real world data set.

From our single-leg simulation experiments, we observe that the upsell heuristic significantly outperforms the algorithm of Wollmer (1992) by as much as 26% and the choice-based algorithm of Gallego et al. (2009) by as much as 24% when the number of classes is large. The projection operation does not significantly alter the seat margin, and thus, can be safely applied to expedite the ADP algorithm without losing accuracy. From our network experiments, we found that the percentage of revenue improvement increases when demand is large and upsell is likely by using our nested allocation policy instead of the RLP bid-price policy. When upsell probabilities can be estimated accurately, the proposed allocation policy rarely does worse than the RLP bid-price policy, and can improve the revenue up to 35% ($\sim \$420,000$) when the upsell probability is high. To be more encouraging, the results are robust to demand magnitude, and hence, similar revenue improvement can be expected from a network that is more or less capacitated. When upsell probabilities cannot be estimated accurately, it is better to underestimate upsell probability when upsell is less likely than opting for the not-buying option. Otherwise, overestimating upsell probability is relatively more beneficial. In the end, we also want to stress the practicality of our algorithm by recalling that it only takes 4 minute to finish running over a medium airline network.

Several interesting questions remain open: 1). Is there a way to estimate the bid prices while capturing upsell based on our ADP algorithm? Currently, the bid prices are itinerary-based and inferior to the RLP bid-prices, as the marginal revenue from sharing capacity on the same flight across multiple itineraries cannot be fully captured. 2). Under what conditions should we switch to the bid-price policy. Note that as our itinerary-based allocation policy is derived mostly at the itinerary level, it *may* not work well when the network is heavily intertwined. 3). Can the ADP algorithm be easily extended to capture other customer behaviors such as cancellation and no-show?

Chapter 2

AIR CARGO ALLOTMENT PLANNING

In the mid-term capacity planning process for air cargo, a cargo carrier reserves capacity up to six months in advance for its clients, who provide regular and frequent shipments over multiple flights. We model the underlying capacity allocation problem as a portfolio optimization problem to allocate cargo space on flights while minimizing the demand covariance between allotments and spot market demand. Due to the complexity of the problem, we develop an efficient partitioning algorithm to decompose the problem into subnetworks and cluster demand. The resulting allocation policy is tested using a real world dataset provided by a solution vendor, and it is benchmarked against a risk-neutral allocation policy used in practice. We observed on average revenue improvement by 2%, which approximately accounts for $\$150,000$ per week for major cargo carriers.

Key words: air cargo, revenue management, stochastic optimization

2.1 INTRODUCTION

Air cargo is an indispensable part of airline business. When an aircraft carries passengers to its destination, its belly is utilized to carry cargo shipments. However, with an expected strong growth of cargo demand since 2010, airlines started to purchase dedicated cargo aircraft (freighters) to exclusively handle cargo shipments. Some carriers even set up an independent cargo division to take advantages from the upcoming demand surge. Nowadays, the cargo business accounts for more than 20% of the total revenue for many major carriers (RITA (2012b)).

Revenue management (RM) in air cargo consists of short and mid-term allocation processes. The short-

term allocation process allocates available flight capacity to volatile spot market demand, and shippers are charged based on the floating market rate. Despite the volatility of the spot market, shippers in the spot market utilize their allocation more effectively as they request capacity only when they are (almost) certain about their shipments. Similarly to the allocation process for passenger RM, the short-term allocation process is run nightly to update the capacity allocation given the remaining capacity and updated demand forecast. The resulting allocation policy is then implemented next day to accept and reject shipment requests. On the other hand, the mid-term allocation process is executed twice per year with a planning horizon of six months. It allocates flight capacity to large and regular shipments that provide a stable revenue stream to the carrier. By promising regular shipments, shippers in return receive a discounted rate and guaranteed space.

The mid-term allocation process is initiated when the carrier releases a new flight schedule. In the process, the carrier sells flight capacity in the form of allotments, which are simply reserved blocks of cargo space, to shippers through various capacity commitments. Those shippers can be freight forwarders, significant clients, and local station managers. Freight forwarders are capacity resellers. They acquire cargo space in advance, and later consolidate spot market demand from their own business to efficiently utilize the acquired cargo space. Significant clients are clients that require a large amount of cargo space over many flights. Although they usually receive a deep discount rate, they account for a significant portion of the total revenue. Local station managers are essentially freight forwarders owned by the carrier. However, they only consider the amount of space that they should acquire on flights that depart from their stations.

Figure 2.1 provides details about the mid-term allocation process in practice. It starts in the first week when the tentative flight schedule for the next six months is released. After receiving the new flight schedule, each shipper/bidder prepares an allotment bid with a bidding price, shipment schedule, and capacity requirements measured in weight, volume, and the number and type of unit load devices (ULD). An ULD is a cargo container or pallet that can be used to containerize shipments, and can be easily loaded onto compatible aircraft. Once the bids are collected, the carrier prepares input for the optimization system including existing allotments to be honored, capacity forecasts, overbooking rates, available itineraries, operating costs, freight rates, and any business rules that are to be complied. The optimization system then converts the requested capacity of each bid to the smallest ULD type to reduce the complexity of the problem, solves a resource allocation problem, and converts the solution back to the originally requested ULD types. It returns an allocation policy that suggests the carrier which bids should be accepted or rejected, which itineraries are used, and how much space should be reserved on each itinerary (depending on the type of the commitment,

the carrier can grant less capacity than requested). However, it neither considers ad-hoc allotment bids, which are bids that come during the planning horizon, nor spot market demand that only appears close to departure. At the end, after fine tuning the allocation for each accepted bid through an iterative negotiation process with its bidder, the carrier constructs allotments by aggregating the allocated weight and volume for each shipper, and uploads the allotments to the booking system for the shippers to book in the future. The entire allocation process takes about seven weeks. Any ad-hoc allotment bids are accepted only if they are profitable, and cargo space is available.

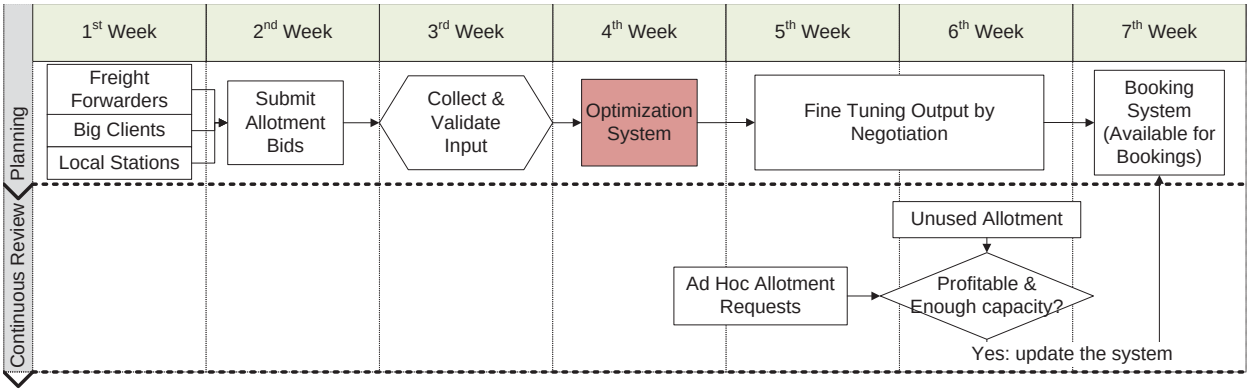

Figure 2.1: The mid-term allocation process.

The two paragraphs just described distinctive allocation processes heavily depend on each other, since flight capacity can be used either as an allotment or reserved for the spot market. When the market rate is high and the allotment utilization is low, the cargo space should be kept for the spot market. Otherwise, allotments can be used to hedge against the volatility of the spot market. Thus, instead of allocating capacity to allotments first and selling the remaining capacity on the spot market later, carriers can utilize their capacity more effectively and economically by considering both allotments and spot market demand simultaneously during the mid-term allocation process.

Although efforts have been made in recent years, RM in cargo has not received comparable attention to RM in passenger, since the cargo business has not been regarded as the main revenue stream for many airlines. In practice, many airlines directly adopt their passenger RM system to manage their cargo business. However, this results in suboptimal capacity controls due to the following major differences between the two RM systems.

- Cargo RM consists of both mid-term and short-term allocation processes while passenger RM has

only the latter.

- Passengers are itinerary-dependent, but cargo shipments are mostly defined at the origin-destination level with a delivery deadline.
- While passenger demand is counted in the number of passengers, cargo demand is counted in the units of weight, volume, and ULDs.
- Similarly, in passenger RM, flight capacity is counted in the number of seats. Cargo capacity is measured in the units of weight, volume, and aircraft positions, which are designated aircraft floor areas with a special equipment to fasten compatible ULDs.
- While passengers can only be show-up or no-show, shippers can partially utilize their allocation without paying any penalties.

In fact, cargo RM is more challenging since the mid-term allocation process allocates capacity six months in advance. Let alone the demand stochasticity and the aforementioned differences, the underlying optimization problem is a large multi-dimensional bin packing problem, which is difficult to solve. Thus, our goal is to modify the allocation model so that allocation requirements can be exactly captured, and an allocation policy can be obtained efficiently and of good quality.

There are three major drawbacks in the current optimization system. Firstly, assigning capacity based on the smallest ULD type may not yield a feasible solution when the allocation decision is converted back to the originally requested ULD types, since there are physical limitations on the number of ULDs of a specific type that can be loaded onto an aircraft. If the solution is infeasible, operators either rerun the optimization with additional constraints or manually adjust the solution in the hope that a feasible solution can be obtained. In either case, considerable amount of time and efforts are required to obtain a feasible solution. Secondly, the optimization systems consider allotments first and treat spot market demand secondarily. In fact, as we explained earlier, spot market demand may as well be equally profitable. Thirdly, the process does not capture the variability and correlated nature of allotments and spot market demand. For example, an allotment being frequently underutilized may indicate an upward shift of the spot market demand for the same product. This could happen when there are constant delays in the supply chain of a shipper. In this case, the allotment should be appropriately reduced so that the extracted space could be sold to more profitable shippers or on the spot market.

In this paper, we propose a portfolio optimization problem for the optimization system to resolve the aforementioned drawbacks. Our goal is to capture the necessary allocation requirements while considering

demand correlation between allotments and spot market demand. In addition, we capture ULDs exactly on both ends (bids and aircraft positions). Due to the size of the problem, we decompose the problem by a partitioning algorithm. It efficiently partitions flights and groups profitable demand together, and returns a demand cluster for each set of flights in the resulting flight partition. Then, the portfolio optimization problem is applied to each demand cluster to minimize the demand covariance within the demand cluster subject to a revenue lower bound and capacity upper bounds on the corresponding flights. To evaluate the solution, we benchmark the resulting allocation policy against the risk-neutral allocation policy, which is obtained by solving our problem without demand covariance. Both allocation policies are tested using a simulator that we constructed to evaluate their performances and sensitivities on both the revenue lower bound and demand clusters. While varying the revenue lower bound affects the tradeoff between the risk-neutral revenue and variability, demand swapping across clusters perturbs the covariance matrices considered by the portfolio optimization problem and provides a fair performance comparison with the risk-neutral allocation that is cluster-independent.

We make the following contributions.

1. We provide a risk-averse portfolio optimization model that considers demand variability, correlation between allotments and spot market demand, demand requirements at the container level, and capacity requirements at the aircraft position level.
2. We propose an efficient partitioning algorithm to decompose the optimization problem into many smaller problems by partitioning flights and clustering demand simultaneously. It keeps the portfolio optimization problem tractable by only including highly profitable and connected demand in a cluster whose size is restricted by a threshold.
3. To the best of our knowledge, we are the first to solve a portfolio optimization model over a capacity constrained air cargo flight network, and provide numerical results and sensitivity analyses. Thus, we combine revenue and risk in the same model and algorithm.

The structure of our paper is as follows. Section 2.2 describes the portfolio optimization problem for the mid-term allocation process, and Section 2.3 presents the partitioning algorithm. A simulator used to evaluate the performance of the allocations is provided in Section 2.4. The case study is discussed in Section 2.5. We conclude our paper in Section 2.6.

2.1.1 LITERATURE REVIEW

Although only a few works capture allotments, and none of them handles ULD exactly with load positions, we aim to provide a brief yet concise literature review on materials related to the short-term and mid-term capacity allocation processes, underlying models, and solution algorithms.

Kasilingam (1997) discusses the role of demand forecasting, capacity forecasting, allotment allocation, and overbooking in the overall capacity allocation process. In addition, he provides an overbooking and bucket allocation models. Different from ours, his model allocates capacity to demand buckets, and includes probabilistic chance constraints that can be linearized to model service level requirements. No demand correlation is considered. A similar overview can be found in Slager and Kapteijns (2004), who discuss implementation challenges of a cargo RM system, and several major differences between RM in cargo and passengers.

Hellermann (2004) considers an allotment contract as an option with reservation and exercise fees. His analysis focuses on parameters, structures, and optimality conditions for both the carrier and shipper models. The carrier model maximizes the total return by determining the optimal reservation and exercise fees, which are then fed to the shipper model to obtain an optimal capacity allocation. He also provides an analysis of the integrated model. However, all the models are single-leg and cannot be extended to capture the cargo network.

Amaruchkul et al. (2010) consider a capacity contract, which sets the allocation level, unit price for the capacity used, and unit refund rate for the unused capacity. They express the expected contribution as a utility function for a carrier and a shipper separately. The utility functions are then combined to yield a stochastic optimization model that maximizes the total contribution subject to an incentive compatibility constraint and a shipper contribution lower bound. Their problem is also single-leg, and it is inapplicable in a network setting.

Karaesmen (2001) formulates the simplified version of the spot market allocation problem by a continuous linear programming problem. She assumes a single shipment type, and shows that solutions to a sequence of linear problems converge to an optimal solution of the continuous linear programming problem, so do the bid prices, which can be computed by an approximation algorithm. Her result is theoretically interesting but cannot be extended to capture allotment requirements.

Popescu (2006) investigates a spot market allocation problem with backlogs and positive lead time. She

divides demand into two groups and designs a different model and algorithm for each group. She also considers cases of her model with one and two time lags, and shows that the optimal allocation, if exists, is a deterministic stationary allocation. Her theories and algorithms are developed directly based on the traditional network RM for passengers (see Talluri and van Ryzin (2004)), and no demand correlation and allotments are considered.

Luo et al. (2009) study the spot market allocation problem in weight and volume while accounting spoilage and offload costs. They analyze the model in terms of booking acceptance regions, which can be circular or rectangular. However, their model is single-leg, and the proposed controls are difficult to implement and store in an information system.

Pak and Dekker (2004) present a dynamic programming (DP) problem to allocate capacity for the spot market. The problem is constrained by flight capacity in both weight and volume. They approximate the value function by a standard knapsack problem, which is further approximated by an ordering algorithm that efficiently generates a set of bid-prices. Amaruchkul et al. (2007) propose several heuristics to decompose and approximate the value function of a DP for the spot market. The DP they model is similar to the traditional passenger RM problem with multiple classes (see Curry (1990), Wollmer (1992), and Brumelle and McGill (1993)). They empirically show that their heuristics perform well. Both Pak and Dekker (2004) and Amaruchkul et al. (2007) address the short-term allocation problem using DP to construct a better allocation policy. However, extending their models to capture allotments significantly increase the complexity of the problem.

Chew et al. (2006) also study a single-leg short-term allocation problem. They focus on updating the allocation as departure approaches. The problem decides how much extra space should be allocated and which shipments should be backlogged in order to minimize the expected cost. Relying on convexity of the cost function, they are able to efficiently construct an optimal solution. Their model is also similar to the traditional passenger RM problem with multiple classes. Hence, it inherits drawbacks similar to the DP in Amaruchkul et al. (2007) and it is a single-leg problem.

Levina et al. (2011) investigate the short-term allocation problem at the network level given the remaining capacity not reserved for allotments. They basically capture all the risk components in the spot market, and propose a simulation-based approach to approximate the optimal allocation. Nonetheless, they do not consider the interaction between the allotments and spot market demand, and adding allotments to their existing model appears to be nontrivial.

Levin et al. (2012) model the capacity allocation problem for a particular origin-destination pair at multiple stages. At the first stage, a mixed integer programming problem is solved to optimally select allotment bids. At the second stage, a DP is run to accept and reject spot market booking requests given the remaining flight capacity. Demand is assumed to be independent. At the final stage, a different model is applied to offload undesirable confirmed bookings. By relaxing the weight and volume capacity constraints of the offloading problem, they obtain an upper bound problem to approximate the original problem that is difficult to solve. Furthermore, the upper bound problem naturally returns a set of bid prices for the spot market. Our work is different in that we allow any pair of demand to be dependent, and we do not rely on solving DPs that are often intractable given a large air cargo network. Instead, we model the problem as a portfolio optimization problem to capture all necessary allocation requirements and demand correlation.

In summary, all past works either are single-leg and not easily extendible to a network setting, or assume independent demand at the network level, or handle the spot market and allotment decisions separately. In addition, they neglect important aspects of ULDs and their positioning requirements on aircraft.

2.2 Problem Definition

For the mid-term capacity allocation process, we model the underlying cargo capacity allocation problem as a portfolio optimization problem to accept profitable and stable capacity requests and assign their ULDs to aircraft positions. The problem has integer decision variables and a quadratic objective. It minimizes the demand covariance computed based on historical capacity misutilization (over/under-utilizing) of the allotments, forecasting inaccuracy of the spot market demand, as well as the correlation between these two variabilities, and subject to all capacity allocation requirements and a revenue lower bound.

We define demand at the origin-destination level, and each demand unit is either an allotment bid or a spot market capacity request. While an allotment bid is counted in weight, volume, and the number and types of ULDs, and consumes capacity over multiple flights over time, spot market demand is a single shipment and is counted only in the units of weight and volume. We add an auxiliary ULD corresponding to the spot market (details provided later). Each demand unit is assumed a fixed rate, i.e. price per weight unit, and a set of available itineraries. For a demand unit, it is allowable to accept only a subset of ULD requests. A single demand unit can be routed through several itineraries. In the case that a demand unit only provides a shipment deadline, any itineraries before the deadline can be selected. Capacity of an aircraft is

counted in the number of positions available for a specific position type, and each aircraft has many types of positions available dependent on its configuration. Each ULD may consume multiple positions and vice versa, and there are restrictions on the types and number of ULDs that can be loaded on a given position type. To describe the problem, let us define the following sets:

- $d \in D$ set of demand units, where d can be an allotment bid or a spot market capacity request,
- $i \in I^D(d)$ set of itineraries for demand unit d,
- $i \in I^{DF}(d, f)$ set of itineraries for demand unit d that use flight f,
- $p \in P(f)$ set of position types available on flight f,
- $u \in U^D(d)$ set of ULD types requested by demand unit d,
- $u \in U^P(p)$ set of ULD types that is compatible with position type p,
- $u \in U^F(f) = \cup_{p \in P(f)} U^P(p)$ set of ULD types that can be loaded on flight f,

and decision variables:

- x_{diu} (integer) number of type-u ULDs of demand unit d accepted on itinerary i,
- y_{fpu} (integer) number of type-u ULDs assigned to position type p on flight f.

Additionally, we define the following coefficients:

- d_{du} number of type-u ULDs requested by demand unit d,
- n_{fp} number of type-p positions available on flight f,
- r_{di} unit revenue for carrying each unit of weight of demand unit d on itinerary i,
- w_f available weight on flight f,
- v_f available volume on flight f,
- ρ^w_{du} unit weight for each type-u ULD of demand unit d,
- ρ^v_{du} unit volume for each type-u ULD of demand unit d,
- ρ^P_u number of positions required to accommodate each unit of type-u ULD,
- $\sigma(d, d')$ covariance of demand units d and d' measured in weight,
- τ^w_u tare weight of a type-u ULD,
- τ^v_u tare volume of a type-u ULD.

Furthermore, let φ be a lower bound on the risk neutral revenue (we will explain later how it is derived).

Formally, the capacity allocation problem (CAP) is

$$CAP^* = \min_{\mathbf{x},\mathbf{y} \text{ integer}} \sum_{d\in D}\sum_{d'\in D} \sigma(d,d') \left(\sum_{i\in I^D(d)} \sum_{u\in U^D(d)} r_{di}\rho^w_{du}x_{diu} \right) \left(\sum_{i\in I^D(d')} \sum_{u\in U^D(d')} r_{d'i}\rho^w_{d'u}x_{d'iu} \right) \tag{2.1}$$

subject to

$$\sum_{d\in D}\sum_{i\in I^D(d)}\sum_{u\in U^D(d)} r_{di}\rho^w_{du}x_{diu} \geq \varphi \tag{2.2}$$

$$\sum_{d\in D}\sum_{i\in I^{DF}(d,f)}\sum_{u\in U^D(d)} \rho^w_{du}x_{diu} + \sum_{p\in P(f)}\sum_{u\in U^P(p)} \tau^w_u y_{fpu} \leq w_f \qquad f\in F \tag{2.3}$$

$$\sum_{d\in D}\sum_{i\in I^{DF}(d,f)}\sum_{u\in U^D(d)} \rho^v_{du}x_{diu} + \sum_{p\in P(f)}\sum_{u\in U^P(p)} \tau^v_u y_{fpu} \leq v_f \qquad f\in F \tag{2.4}$$

$$\sum_{d\in D}\sum_{i\in I^{DF}(d,f)} x_{diu} \leq \sum_{p\in P(f)} y_{fpu} \qquad u\in U^F(f), f\in F \tag{2.5}$$

$$\sum_{u\in U^P(p)} \rho^p_u y_{fpu} \leq n_{fp} \qquad p\in P(f), f\in F \tag{2.6}$$

$$\sum_{i\in I^D(d)} x_{diu} \leq d_{du} \qquad u\in U^D(d), d\in D. \tag{2.7}$$

Objective function (2.1) minimizes the covariance between each pair of accepted demand units. Since each unit of weight is not valued the same for each demand unit, the associated unit rate is multiplied. The objective can also be modified to capture covariance in volume or volumetric weight (see DHL (2012)), and doing so will not significantly affect the solution as the density of the demand is fixed. Thus, variation in weight corresponds to variation in volume, and vice versa. Constraint (2.2) imposes the revenue lower bound for a given revenue level φ. Constraints (2.3) and (2.4) are the weight and volume upper bounds on each flight. If an ULD of a specific type is used, its tare weight and volume are added. In layman's terms, the left-hand side sums the weight (volume) of the demand accepted and tare or empty weight (volume) of ULD's used. Constraints (2.5) convert the acceptance decisions x_{diu} to the allocation decisions y_{fpu}. Constraints (2.6) are the upper bounds on the number of positions for each position type on each flight, and constraints (2.7) are the demand upper bounds at the ULD level. Other business requirements such as total tonnage upper and lower bounds at the origin-destination and flight levels are implemented but not presented.

To describe how the covariance is estimated, let us refer to the status of a cargo shipment provided by a shipper but not yet shipped as tendered, and the weight of a tendered shipment as tendered weight. We

define s^w_{dt} to be the tendered weight of demand unit d on historical date t, μ^w_d to be the targeted weight as anticipated by the bid of demand unit d, and $T(d, d')$ to be the set of shipping dates that both demand units d and d' have shipping records. The targeted weight is the weight granted for the allotment, and is the weight forecast for a spot market demand unit. The covariance between demand units d and d' is estimated based on $\sigma(d, d') = (\sum_{t \in T(d,d')}(s^w_{dt} - \mu^w_d)(s^w_{d't} - \mu^w_{d'}))/|T(d, d')|$.

In our implementation, we additionally introduce two new ULD types. The first type corresponds to bulk cargo that cannot be fitted into any ULDs, and the second type is to hold spot market demand for which only the amount of weight and volume are available at the beginning of the mid-term allocation process. As a consequence of the first ULD type, a new position type is added to represent the bulk compartment on the aircraft. The second ULD type requires the density of the ULD to be demand dependent, and hence, we use ρ^w_{du} instead of ρ^w_u for spot demand d.

In summary, CAP provides an optimal allocation policy that minimizes capacity misutilization of allotments and spot market demand variability while maintaining an acceptable revenue level. However, this problem is difficult to solve, since the covariance matrix is large, and an integer solution is required. Although business experience and intuition help reduce the complexity of the problem, we approach the problem in a general manner by imposing a block diagonal structure to the covariance matrix. This is done by partitioning the set of all flights, and each set of flights in the partition implies a demand cluster. A covariance matrix is then computed for each demand cluster, and the CAP is solved for each corresponding subset of flights. Details are provided next.

2.3 Solution Methodology

In this section, we discuss an algorithmic framework that decomposes the portfolio optimization problem into many smaller problems, and each of these smaller problems has its own demand cluster and subset of flights. The framework is illustrated in Figure 2.2. It starts by querying necessary information such as costs, flights, spot market forecast, allotments, and business requirements from a database. A resource allocation problem, the risk neutral problem discussed next, is solved first to yield a preliminary allocation solution. This solution is then used to provide a direction for a partitioning algorithm to generate a flight partition and a set of demand clusters. At the end, for each demand cluster, the covariance matrix is computed using historical bookings and shipping records of the demand units within the cluster, and CAP is applied to each

corresponding set of flights to yield the final allocation that we evaluate though a simulator discussed later in Section 2.4.

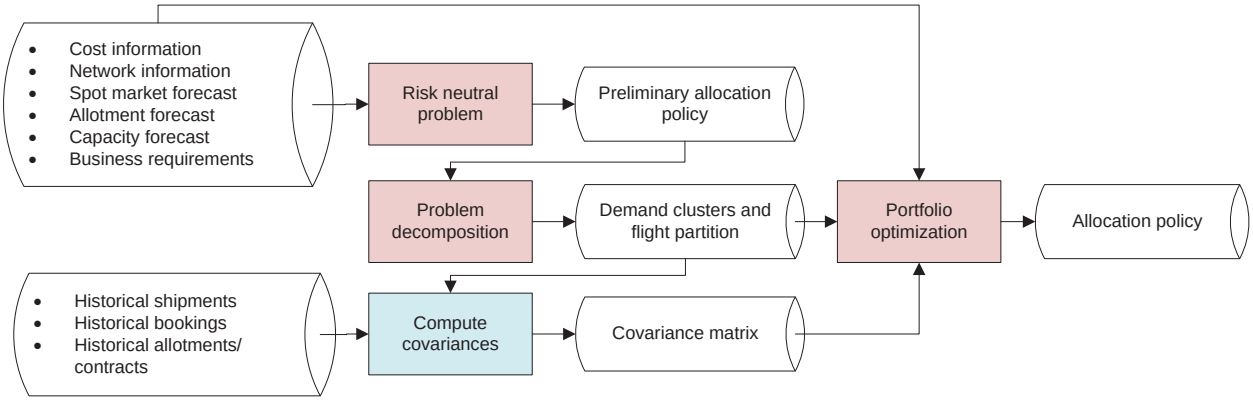

Figure 2.2: Proposed algorithmic framework for problem decomposition.

2.3.1 Risk Neutral Problem

In order to provide a direction to partition flights and cluster demand, the risk neutral problem (RNP) is solved. It is the same problem as the portfolio optimization problem, except that its objective is replaced by the left-hand side of constraint (2.2). The modified problem maximizes the total allocation revenue without accounting for demand covariances. Since RNP is to be solved over a long planning horizon, a rolling horizon approach is implemented. This is accomplished by adding a time dimension to the decision variable y_{fpu} and changing the right hand side coefficients of constraints (2.3), (2.4), and (2.6) to the remaining weight, volume, and number of positions, respectively. The complete model can be found in Appendix B.1. Let τ be the length of the time window. For a given starting time $t = 1, \ldots, T$, where T is the length of the time horizon. Only demand units and resources in $[t, t+\tau]$ and across time $t+\tau$ are considered. A demand is classified as across-time $t+\tau$ if it can be assigned to an itinerary that has a flight that departs before time $t+\tau$ and arrives after time $t+\tau$. Any demand units that have previously been accepted or rejected are excluded from the current time window, and are used to update the capacity upper bounds before the RNP for the current time window is solved. Once the incumbent problem is solved, only the solution at time t is kept before moving the starting time from t to $t+1$. When $t+\tau = T$, all allocation decisions from $T-\tau$ to T are kept.

2.3.2 PROBLEM DECOMPOSITION

To decompose CAP into many smaller problems, we partition flights and cluster demand units simultaneously. Our strategy is to group demand units by their contribution evaluated based on the optimal allocation of RNP. The RNP can assign a demand unit to several different itineraries. If a demand unit is assigned to multiple itineraries, and all these itineraries span several flight sets, this demand unit is assigned only to the empty set. Otherwise, there is at least one itinerary of the demand unit that is contained in a flight set. Among all such itineraries, we select the itinerary with the highest revenue based on the RNP solution and assign the demand unit to the demand cluster corresponding to the underlying flight set of the itinerary. In addition, this demand unit is also assigned to the empty set. Note that a demand unit can be simultaneously assigned to a cluster and the empty set, but it cannot belong to two demand clusters at the same time. By the definition of demand clusters, flights for each demand cluster are linked by itineraries. Each flight set implies a demand cluster, and vice versa. In addition, there is an empty set demand cluster that does not correspond to a flight set. At the end, CAP is solved for each flight set with the covariance matrix computed based on the corresponding demand cluster.

Since our strategy groups demand units by their contribution, it does not guarantee that demand units are heavily correlated within a cluster, or that demand units are independent across clusters. Although an optimal clustering strategy would maximize the number of demand units that satisfy these two properties, it requires each demand pair to be first examined before the covariance can be used for one of the clustering criteria. The resulting long solution time and large storage requirement likely render such a strategy impractical. The advantage of our approach is that if two profitable demand units in the same cluster are heavily correlated, we are ensured that CAP has already accounted for their covariance when an allocation is made. Since high revenue demand units are likely assigned to a cluster, their dependencies are captured by CAP.

To describe the partitioning problem mathematically, we define the following:

- $DI = \{(d, i) : i \in I^D(d), d \in D\}$ set of all demand-itinerary pairs,
- F set of all flights,
- $DI(\overline{F}, \overline{D})$ set of demand-itinerary pairs that are formed by $\overline{D} \subseteq D$, and each demand unit in $\overline{D}$ uses flights exclusively in $\overline{F} \subseteq F$,
- K upper bound on the number of demand units in each cluster,
- $R(\overline{DI})$ total revenue from RNP computed based on a set of demand-itinerary pairs $\overline{DI} \subseteq DI$.

The decision variables of the partitioning problem are

- S total number of demand clusters (flight sets),
- F_s flight set corresponding to cluster $s = 1, \ldots, S$,
- $D_s(F_s)$ set of demand units assigned to cluster s given flight set F_s.

For a given solution, we define $DI(\emptyset) = DI \setminus \cup_{s=1}^{S} DI(F_s, D_s(F_s))$. The partitioning problem reads

$$\max_{F_s, D_s(F_s), s=1,\ldots,S} \sum_{s=1}^{S} R(DI(F_s, D_s(F_s))) - R(DI(\emptyset))$$

subject to

$$|D_s(F_s)| \leq K \qquad s = 1, \ldots, S \tag{2.8}$$

$$D_s(F_s) \cap D_s(F_{s'}) = \emptyset \qquad s \neq s' \text{ and } s, s' = 1, \ldots, S \tag{2.9}$$

$$F_s \cap F_{s'} = \emptyset \qquad s \neq s' \text{ and } s, s' = 1, \ldots, S \tag{2.10}$$

$$\cup_{s=1}^{S} F_s = F. \tag{2.11}$$

The objective is to maximize the total contribution of the demand-itinerary pairs over all clusters minus the contribution of the demand-itinerary pairs in the empty set, which includes both the demand-itinerary pairs that span across multiple flight sets, and the remaining demand-itinerary pairs formed by demand units not in any cluster.

Given solution $\bar{\mathbf{x}}$ to CAP, we define $R(\overline{DI}) = \sum_{(d,i) \in \overline{DI}} r_{di} \sum_{u \in U^D(d)} \rho_{du}^w \bar{x}_{diu}$. Since the total revenue of RNP is fixed, maximizing the total contribution of demand-itinerary pairs is equivalent to minimizing the contribution of the empty set. Thus, without loss of generality, the objective can be rewritten as

$$\max_{S, F_s, D_s(F_s), s=1,\ldots,S} \sum_{s=1}^{S} R(DI(F_s, D_s(F_s))),$$

which simply maximizes the total contribution over all demand-itinerary pairs in the clusters.

This partitioning problem is a large and complex integer programming problem, which is intractable over the entire planning horizon. An efficient flight-based partitioning heuristic is developed to efficiently retrieve a good solution.

The heuristic iteratively includes profitable flights into a flight set in question until the number of demand units in the corresponding cluster exceeds K. It requires a feasible solution $\bar{x}_{diu}$ as input to RNP, set of all

flights, and set of all demand-itinerary pairs. It starts a new flight set with the most profitable flight that has not yet been included to any other flight sets, where profitability of a flight is defined by the total revenue collected from the demand units assigned by RNP to the flight. The algorithm then iteratively adds profitable flights so that more profitable demand-itinerary pairs can be assigned to that flight set. At the end, it returns a flight partition and the corresponding demand clusters for each flight set in the partition. Note that the solution returned by the heuristic is feasible to the partitioning problem, and if $K = |D|$, we simply obtain a trivial partition that includes all flights.

To fully describe the heuristic, we define $G(\overline{F}, \overline{D})$ to be the set of demand units that are not in $\overline{D} \subseteq D$, and each demand unit in the set has been assigned by RNP to at least one itinerary that uses flights exclusively in $\overline{F} \subseteq F$. We also define L_i to be the set of flights in itinerary i. The heuristic is presented in Algorithm 3.

Algorithm 3 Flight-based Partitioning Heuristic

Require: $\bar{\mathbf{x}}$, F, and DI

1: Set $s = 1$
2: Set $\bar{D} = D$, $\bar{F} = F$ and $F_j = \emptyset$ for $j = 1, \ldots, |F|$
3: **while** $\bar{F}$ is not empty **do**
4: Let $f^* = \arg\max_{f \in \bar{F}} \sum_{(d,i) \in DI: f \in L_i, d \in \bar{D}} r_{di} \sum_{u \in U^P(d)} \rho^w_{du} \bar{x}_{diu}$
5: $F_s = F_s \cup \{f^*\}$
6: **repeat**
7: Set $\bar{f} = \arg\max_{f \in \bar{F}} \sum_{(d,i) \in DI(F_s \cup \{f\}), d \in \bar{D}} r_{di} \sum_{u \in U^P(d)} \rho^w_{du} \bar{x}_{diu}$
8: $F_s = F_s \cup \{\bar{f}\}$
9: **until** $|G(F_s, \bar{D})| > K$
10: $D_s(F_s) = G(F_s, \bar{D})$
11: $\bar{D} = \bar{D} \backslash G(F_s, \bar{D})$
12: $\bar{F} = \bar{F} \backslash F_s$
13: $s = s + 1$
14: **end while**
15: **return** F_j and $D_j(F_j)$ for $j = 1, \ldots, s$

Step 4 of Algorithm 3 selects the flight that contributes the most and initializes a new flight set with that flight. In steps 6 - 9, we enlarge the flight set by iteratively including flights that bring highly profitable demand-itinerary pairs. Step 10 updates the demand cluster $D_s(F_s)$ accordingly. At the end, the algorithm returns a flight partition and a demand cluster for each flight set in the partition, and its outcome defines the empty set $DI(\emptyset)$.

2.3.3 PORTFOLIO OPTIMIZATION

Once the flight partition and demand clusters are found, both the capacity of the flights and demand units in the cluster are adjusted accordingly by subtracting the portion of the demand on the demand-itinerary pairs in the empty set. The corresponding decision variables $\{x_{diu}\}_{u \in U^D(d)}$ are excluded before CAP is applied to each flight set to produce an allocation policy. The allocation policies over all flight sets are then combined to form the final allocation policy to be evaluated through a simulator, which we discuss next.

2.4 SIMULATION

We construct a simulator to evaluate the performance of the allocation policy returned by the CAP. The goal is to evaluate the quality of the partition and thus of the fact that only certain covariances are considered. The simulation is divided in two steps (see Figure 2.3).

Recall that the only uncertainty assumed is the difference between the weight in the bid and the actual tendered amount. The first step is the shipment generation process that generates shipment weight samples. We assume that weight is multinomial dependent. For spot market demand, we further assume that the density is constant. Hence, a weight sample implies a volume sample. For each allotment shipment, given a corresponding weight sample (we discuss how this is generated in the next paragraph), the weight of the sample is proportionally distributed over all originally requested ULD types in the bid, where the proportion is determined based on the required weights of the originally requested ULD types. Given the distributed weight for each ULD type, the weight is then converted to the required number of ULDs.

The shipment weight sample generation process ideally requires means (historical) and the covariances over all demand units. However, due to the fact that the covariance matrix is too large to be stored for generating multinomial shipment weight samples, we, instead of using the covariance matrices computed based on the existing clusters, swap demand units between clusters before shipment weight samples are generated. This demand swapping process provides a fair evaluation of the partitioning heuristic while avoiding the huge storage requirements associated with utilizing the entire covariance matrix. Specifically, for each cluster, the simulator first randomly and uniformly selects some specified percentage of demand units. The extracted demand units are then randomly and uniformly redistributed to other clusters. Note that the number of demand units in each cluster can now go above the upper bound on the number of demand

units allowed in the cluster. After demand units are redistributed, a covariance matrix is computed for each new cluster. Shipment weight samples are then generated using a multivariate normal random number generator that takes the new covariance matrices as input. If demand units are not distributed, then sampling would favor our partitioning heuristic and provide a biased assessment. For this reason, we randomly swap demand units to sample "different parts" of the overall covariance matrix.

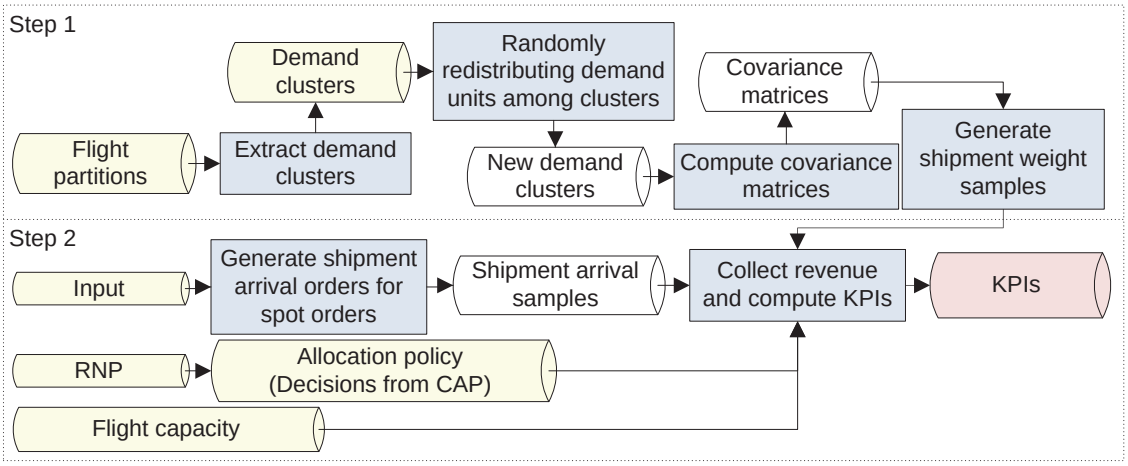

Figure 2.3: Simulator for Policy Evaluation

The second step generates the arrival orders of the shipments once shipment weight samples are obtained. The arrival of a shipment is determined based on the first date that the shipment is available to be shipped. Such a date can be identified by the earliest itinerary that can carry the shipment to its destination. The shipments are then grouped by the dates that they arrive, and if there are multiple shipments on the same date, the arrival orders of the shipments are randomized to generate the arrival samples.

With the given allocation policy and remaining flight capacity, the simulator collects revenue from each shipment sample by assigning its requested weight, volume, and number of ULDs to all available itineraries that are sorted by their operating costs in ascending order. A spot request might not be accepted by an allocation policy, yet in operations, there might be sufficient buffer capacity to accept it, where the buffer capacity is reserved to account for the volatility of the demand. In CAP, the buffer capacity is the capacity not allocated to any demand unit. Such capacity is possible since the objective of CAP is not to maximize the allocation revenue. Our simulation takes this into account by accepting spot demand without an allocation using the buffer capacity. This essentailly injects the first-come-first-serve (FCFS) policy into the simulator, and it is only appplied to demand units without an allocation.

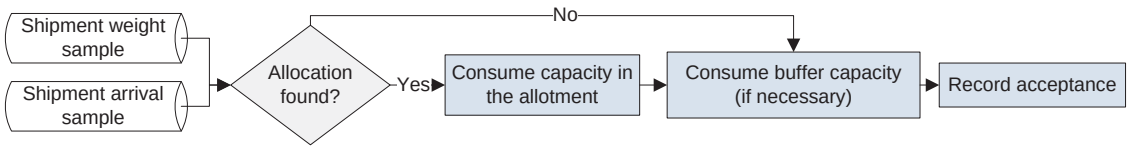

Figure 2.4: Revenue collecting process

Figure 2.4 shows the revenue collection process in detail. The process begins with a shipment and checks if an allocation for the shipment can be found from the existing allocation policy. If an allocation is found, its capacity is then reduced until either the demand is fully accepted or no residual allocation remains. In the latter case or when an allocation is not found, buffer capacity on each available itinerary is utilized to carry the remaining spot demand by randomly choosing compatible positions. This is shown by the upper arrow in Figure 2.4. At the end, the process records the acceptance information to compute the KPIs, and the acceptance information includes the amount of weight and volume, the number of ULDs accepted for each ULD type, and the number of positions consumed on each carrying flight.

The necessary KPIs used to measure the performance on the allocation policy are the total revenue, overtendered demand, and underutilized capacity. The revenue is computed based on the total shipped weight of the demand multiplied by the underlying rate, and it is captured by the left-hand side of constraint (2.2) instead of the risk-driven objective function of CAP. The overtendered demand is the positive difference of the shipped demand and the weight specified by the bid of an allotment or expected weight of spot demand. It measures how much of the overtendered demand is accepted and shipped not through the allocation policy. The underutilized capacity is the positive difference of the allocation produced by the policy and the utilized capacity. It measures if the allocation is sufficiently utilized. Although the best policy should maximize the total revenue while minimizing both the overtendered demand and underutilized capacity, in practice when underutilized capacity increases, overtendered demand tends to decreases, and vice versa. We discuss this observation further in the next section.

2.5 Computational Study

We test our proposed allocation policy using real-world data provided by a major cargo RM solution vendor. It has two weeks of cargo data with over $200,000$ spot market demand forecasts, 100 allotments, $350,000$ itineraries, and $28,000$ flights. We benchmark our solution against the allocation policy used in practice,

which is obtained by solving the risk-neutral problem (RNP, see Section 2.3.1) without considering any demand covariance. We want to see if the allocation policy obtained by discounting the risk-neutral revenue to minimize the allocation risk can perform better than a risk-neutral allocation policy that does not consider the risk component.

We solve the RNP in the rolling horizon manner with the rolling horizon window set to be 7 days (the problem is too hard to be solved in one attempt over 14 days). For the CAP, the revenue lower bound φ in constraint (2.2) is defined to be αz, where α is the revenue discount factor, and z is obtained by solving the RNP without the rolling horizon.

An algorithm by Higham (2002) is implemented to find the nearest semi-positive definite covariance matrix, where the distance is measured by the two-norm. This is to handle covariance matrices with negative eigenvalues, which could happen due to rounding errors. In our experiments, this algorithm is run only for a few clusters, whose corresponding covariance matrices have almost negligible negative eigenvalues.

We present the computational study in three parts. In the first two parts, we benchmarked the CAP allocation policy against the RNP allocation policy. The first part demonstrates the performance of the CAP allocation policy when it is used in practice, and its performance is evaluated by feeding the simulator presented in Section 2.4 directly with historical shipments streams. Neither shipment weight and arrival samples are generated, nor clusters are perturbed. To evaluate the effects of changing the revenue lower bound, we vary the revenue discount factor α from 85% to 100% with an increment of 2.5%. Note that each value of α corresponds to a different allocation solution, and when α is 100%, the underlying solution is the RNP allocation policy that provides a baseline for comparisons. This range of α is selected as it is unlikely that a carrier is willing to trade more than 15% off its potentially achievable revenue (if all shipments are tendered as expected) to account for the risk.

The second part studies both the performance of the CAP allocation policy with a broader range of α and the sensitivity of the allocation policy to the revenue lower bound and demand clusters. We run the simulator to generate $100,000$ demand-swapping samples. For each of these samples, the simulator generates $100,000$ shipment weight and arrival samples. These large numbers of samples ensure that the effect of standard errors is minimized and the results are at least 95% statistically significant. We test α from 55% to 100% with an increment of 5%, and again, $\alpha = 100\%$ corresponds to the RNP allocation policy. Furthermore, we test the sensitivity of the CAP allocation policy on the demand clusters produced by the partitioning heuristic described in Section 2.3. It is done by randomly swapping demand units across different clusters before the

clusters are used to generate shipment weight samples. Recall from Section 2.4 that demand swapping is a strategy to avoid the use of the entire covariance matrix to generate shipment weight samples. We vary β, the distribution factor that controls the percentage of demand units to be randomly extracted from each cluster and uniformly distributed to other clusters, from 0% to 90% with an increment of 10%.

The last part reports the behavior of CAP when the revenue lower bound varies. Specially, we show how both the objective value and the risk-neutral revenue computed using (2.2) change when α decreases from 100%. Furthermore, computational times required for each process in Figure 2.2 are also reported.

All experiments are conducted on a server with a 64-bit Window 2003 server operating system. Its CPU is Intel Xeon(R) with four 2.67 GHz cores, and has 12 GB of RAM. The data is stored in an Oracle 11g database. The implementation was coded in Java, and ILOG Cplex 12.3 is used for optimization.

2.5.1 Historical Shipment Results

Table 2.1 summarizes the results when historical shipment records are directly applied to the simulator using the CAP allocation policies that correspond to various values of α. The revenue, overtendered demand, and underutilized capacity are direct output of the simulator, and all percentages are computed using the output of the RNP allocation policy.

Table 2.1: Performance summary when historical shipment records are directly applied.

Revenue Discount Factor	97.5%	95%	92.5%	90%	87.5%	85%
Revenue (%)	0.99	1.72	1.84	1.92	1.70	1.50
Overtendered Weight (%)	5.21	8.10	8.75	10.00	10.25	11.82
Overtendered Volume (%)	7.52	10.61	12.94	13.31	14.30	15.19
Underutilized Weight (%)	-1.84	-5.44	-7.45	-13.06	-15.76	-17.37
Underutilized Volume (%)	-0.68	-4.43	-8.94	-13.49	-14.76	-16.10

We observe that revenue can be improved close to 2% when α is set to be 90%. It means that it is worthwhile to reserve some capacity for risk buffering, and the improvement gradually decreases in both ends. Decreasing toward larger α is due to the fact that the allocation policy gradually becomes the RNP allocation policy, which leaves no capacity unallocated. Decreasing toward smaller α is due to the heavier usage of the first-come-first-serve policy (FCFS), i.e. without any control, as more capacity is set assigned for risk buffering.

Overtendered weight and underutilized weight go in the opposite direction as α increases. When α in-

creases, allocation get reduced, and consequently, overtendered weight increases while underutilized weight decreases. The trends are similar for both weight and volume.

2.5.2 Simulation Results

We now present our simulation results and sensitivity analysis. Figure 2.5 shows the percentage of the revenue change over RNP by using the CAP allocation policy for each selected value of α and β presented in the beginning of this section.

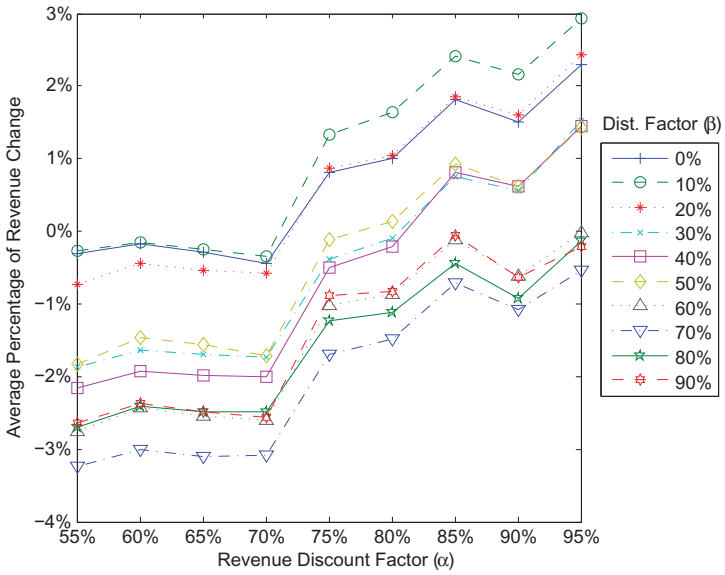

Figure 2.5: Average percentage of revenue change over RNP

In general, the revenue improvement gradually increases when α increases and β decreases. The trend of improvement is similar over all values of β. Several observations are worth mentioning.

1. The revenue change is not monotonic in α. Monotonicity is usually observed in traditional portfolio optimization applications, in which no resources are shared at the network level except for a simple budget constraint. Within the cargo network, due to integrality, and its flight capacity constraints, monotonicity could not be expected.
2. Similarly, the revenue improvement is not monotonic in β. In fact, the best revenue improvement is observed when β is set to be 10%, and the worst revenue improvement is obtained when β is set to be 70%. These contribute to the fact that the partitioning algorithm (Algorithm 3) is a heuristic, which does not cluster demand units by their correlation but by their risk-neutral revenue contribution.

Nonetheless, our results show that our partitioning heuristic works well and is relatively robust to β, since even when the percentage of demand units being swapped is around 50%, and so long as α is at least 85%, the revenue improvement is about 1%.

3. When α is less than 75%, our allocation policy is outperformed by the risk-neutral allocation policy regardless of the value of β. The reason is that too much revenue is traded to reduce the total covariance, i.e., for anticipated uncertainty. This is also due to the fact that CAP naturally discourages allocation, and the unallocated capacity is consumed in the FCFS manner, i.e. without any control, and hence, the risk-neutral allocation policy ultimately prevails.
4. When α is no less than 85% and β is no more than 20%, we observe a revenue improvement about 2% on average by reserving a small amount of capacity and using it as a risk buffer, and as high as 3% of revenue improvement can be achieved, which could account for approximately $\$225,000$ per week for major carriers.

We also measure the overtendered weight and underutilized capacity. Figure 2.6 shows the percentage of overtended weight change over RNP. In general, the percentage of overtended weight increases when less capacity is reserved. This is due to the fact that more demand units are pushed to be accepted via FCFS. When β increases, the percentage of overtendered weight increases, as the allocated capacity may not be well utilized due to demand swapping. Similarly to the average percentage of revenue change, the percentage of the overtendered weight increases the least when β is 10%. Specifically, about 11% of demand units are accepted on average via FCFS when α is no less than 85% and β is no more than 20%.

On the other hand, Figure 2.7 shows the percentage of underutilized weight change (a negative number corresponds to a reduction of underutilized capacity). In general, the percentage of undertended weight increases when α increases as more capacity is reserved for risk buffering. When more capacity is reserved, the allocation becomes less, and hence, is more likely to be underutilized. Similarly, when β increases, the percentage of underutilized weight increases. The reason is that the allocated capacity is not well utilized due to demand swapping. Specifically, when α is no less than 85% and β is no more than 20%, the percentage of underutilized capacity can be reduced by more than 15%.

For both overtendered weight and underutilized capacity, the trends are similar among different values of β, and the same conclusions can be drawn when the overtendered demand and underutilized capacity are measured in volume.

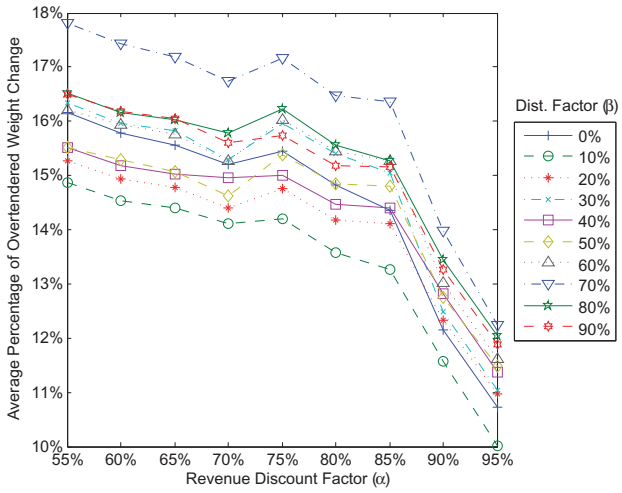

Figure 2.6: Average percentage change of overtendered weight

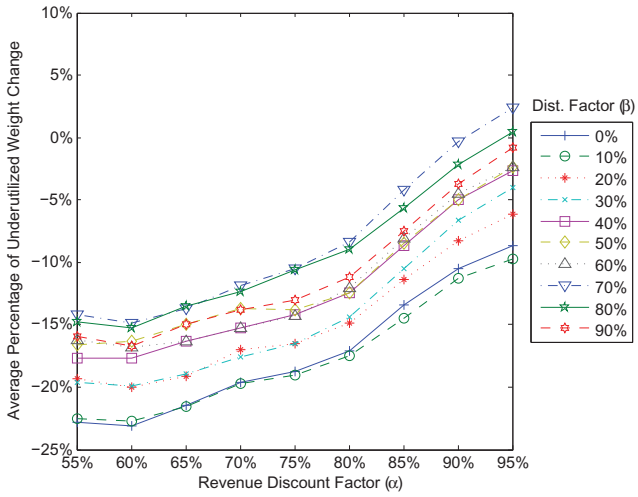

Figure 2.7: Average percentage change of underutilized weight

2.5.3 CAP Optimization Results

Lastly, we present results on how the objective of CAP and its total revenue, computed based on the left-hand side of constraint (2.2), change when different revenue lower bounds are set. Figure 2.8 illustrates the percentage of revenue reduced for different revenue discount factors, where the percentage is computed using the revenue obtained when α is set to be 100%. It shows that the revenue reduced decreases in a concave manner when α increases. This implies that as the revenue discount factor decreases, the allocation does not have to change as much in exchange for a lower total covariance, and similar revenue levels can be kept albeit decreasing slowly. This is due to the fact that the allocation policy, instead of experiencing substantial changes, is simply reduced to further minimize the objective of CAP when α gradually decreases. On the other hand, Figure 2.9 shows the percentage of the total covariance reduced when α varies. The covariance reduction curve behaves similarly to the revenue reduction curve that it is concave and decreasing in α. This is again resulting from the fact that reducing revenue lower bound further does not significantly changes the underlying allocation policy, and hence, the total covariance is not reduced as much.

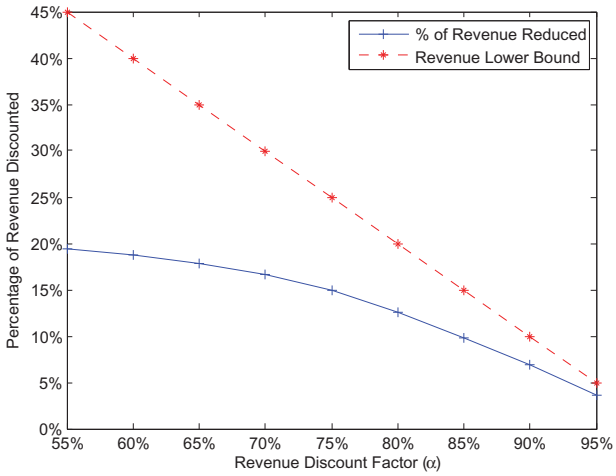

Figure 2.8: Percentage of revenue reduced

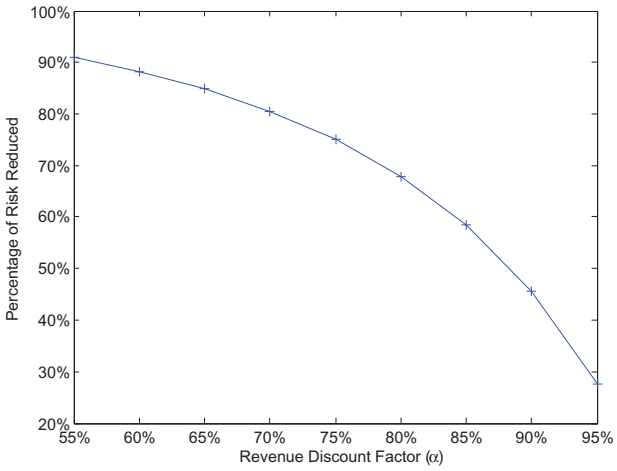

Figure 2.9: Percentage of covariance reduced

Finally, the running time is about 10 minutes for the risk neutral problem, 20 minutes for the flight partitioning algorithm, and 20 minutes for the CAP over all clusters. Note that the running time of CAP is only twice of the running time for the risk neutral problem. The reason is that multi-thread computing is applied to each demand cluster. However, no parallelization scheme is available to the risk neutral problem due to the solution dependency on the rolling horizon framework.

In summary, the running time for a six month period should be about 13 hours which shows that our proposed algorithm framework is practical.

2.6 Conclusion

In conclusion, we have developed a framework to solve a difficult optimization problem that considers the interaction between allotments and spot market demand, accepts demand at the ULD level, and allocates flight capacity at the aircraft position level. Through conducting a set of comprehensive simulation experiments using a real world dataset provided by a major solution vendor, we demonstrate the practicality of our optimization framework, and show that revenue can be improved by 2% when interaction between demand can be captured, which can be translated to a substantial saving of $\$150,000$ per week for major cargo carriers.

During this study, we have identified several future directions to extend this research. One interesting direction is to derive a way to obtain a bid-price at the aircraft position level. Another direction is to extend

the problem to capture the stochasticity nature of the demand, and derive an efficient algorithm to solve it. Lastly, improving the partitioning algorithm may provide a more robust solution to various parameters we have tested.

Chapter 3

AIRLINE INTEGRATED RECOVERY

Airline operations recovery has to be invoked frequently in daily operations, it can be costly and can be triggered by any factor that prevents resources (aircraft, crews, and passengers) from flying as planned. The goal is to efficiently resume regular operations while minimizing the recovery cost. However, due to the size and complexity of the recovery operations, the schedule for each resource is recovered separately and sequentially in practice. We study a fully integrated recovery problem and solve it by Benders decomposition. Specifically, the master problem delays and cancels flights, assigns an equipment to each open flight, and selects maintenance schedules for aircraft to commit. The first Benders subproblem routes each aircraft to fulfill the selected maintenance schedules, while the second subproblem assigns each cockpit crew and flight attendant to a roster that satisfies all work rules. Both subproblems provide Benders cuts to the master problem to iteratively improve the recovery solution. Multiple modeling and algorithmic strategies are proposed to efficiently solve the aircraft subproblem and generate feasible rosters for the crew subproblem. We tested our integrated recovery algorithm on a real world data set over multiple large disruption instances, and compared it with an existing partially integrated solution used in practice. The revenue improvement can be up to 8%, which accounts for one million dollars in saving per disruption.

Key words: airline recovery, multi-commodity flow, Benders decomposition.

3.1 INTRODUCTION

Irregular operations are inevitable for airlines with thousands of daily flights, and cost the airlines considerably to resume regular operations. According to RITA (2012a), only 84% of flights in 2012 were on time. With a very tight revenue margin, airlines actively seek advance methods to further reduce the recovery cost. Irregular operations lead to flight delays and cancellations, and may be caused by adverse weather conditions, an irregular aircraft maintenance, terminal gate unavailability, or even crew sick leave. A delay on a flight may delay all of its connecting flights that share the same resources (aircraft, crew members, or passengers). Consequently, the residual effect is propagated, and the affected zone is enlarged. To further exacerbate the situation, a connecting flight needs to be canceled when its inbound flight has been delayed sufficiently long, and the connection time for a resource becomes illegally short.

We call any resource *disrupted* if its original schedule can no longer be executed, and *recovered* if the disrupted resource is assigned with an alternative schedule that resumes its regular operations before the end of the recovery time window. In addition, a *leg* is either a flight or a delayed flight copy with a different arrival time and departure time, a *maintenance event* is a scheduled maintenance for an aircraft based on the planned schedule, a *maintenance schedule* is a possible alternative time and location for a maintenance event (a maintenance event can have several alternative maintenance schedules). Since the recovery time window is short, we assume that an aircraft has at most one maintenance event in the time window. On the crew side, a *crew member* is either a cockpit crew member or a flight attendant, a *crew* is the set of crew members needed to operate an equipment assigned to a flight, and a *segment* in a roster of a crew is either a leg or non-flying event (e.g. day-off, sick leave, training, etc).

When flight schedules are disrupted, the operations control center (OCC) of an airline strives to resume regular operations in a relatively short time while minimizing the recovery cost. A fully integrated recovery problem is composed of a schedule recovery problem and three resource recovery problems with linking constraints. The schedule recovery problem considers curfew compliance, slot requirements, and gate availability. It reassigns fleets or equipments of different capacity to undisrupted and delayed flights to accommodate disrupted passengers. It can also cancel flights if by doing so, other resources can be successfully recovered. Once the schedule recovery problem is solved, the aircraft recovery problem reroutes disrupted aircraft of the same equipment to satisfy mandatory maintenance subject to maintenance capacity constraints. Given a set of aircraft routes, the crew recovery problem then reassigns each disrupted crew

member to a different roster while satisfying all work rules. Reserve and stand-by crews may also be utilized. Lastly, the passenger recovery problem assigns disrupted passengers to different itineraries so that they can be carried to their final destinations without much delay. Although all the recovery problems can be integrated and solved at once, an OCC operator solves them sequentially due to the complexity and combinatorial nature of the integrated recovery problem.

Since the crash of Colgan Air in 2009, by the end of 2011 the Federal Aviation Administration finalized rule changes about pilot fatigue to improve the working conditions of pilots on duty. The cost for each airline to accommodate the new work rules is estimated to be about $\$300$ million over the next decade. This heavy burden is in addition to a fine of $\$27,500$ per passenger for a tarmac delay over three hours. Thus, an integrated recovery module that further reduces the recovery cost is instrumental in improving the revenue margin of the airlines.

The problem that we address is to recover disrupted flight schedules in the least costly manner given a recovery time window. A flight schedule is considered recovered if the underlying resources for all flights to depart in the time windows are available by the time of (delayed) departure. Our work follows Lettovsky (1997), who proposes a model for the fully integrated recovery problem and the use of Benders decomposition due to the block diagonal structure of his model. In his decomposition framework, the schedule recovery problem is treated as the master problem while the aircraft, crew, and passenger recovery problems are the subproblems and solved sequentially. The linkage is by the equipment assignment to flights. Once the master problem is solved, the aircraft and crew recovery problems can be run in parallel for each equipment. If a subproblem is infeasible, the master problem is solved again with a new feasibility cut generated. Otherwise, an optimality cut is added to the master problem, and the next subproblem is considered. The procedure is repeated until the reduction in cost is minimal, or a time limit is reached. At the end, a special branching rule is proposed to obtain an integer solution. Lettovsky (1997) exhibits the model and a conceptual algorithm, but no implementation and computational studies are provided. We are confident that out-of-the-box Benders decomposition with path-based subproblem formulations does not scale.

Recently, Peterson et al. (2012) has published a similar effort on the fully integrated recovery problem. However, they model the schedule and resource recovery problems using strings proposed by Barnhart et al. (1998) and consider only the passenger recovery costs. As the aircraft and crew recovery problems are solved after the passenger recovery problem, their Benders algorithm is geared towards passenger recovery.

In this paper, we propose a model for the fully integrated recovery problem along with the underly-

ing solution methods. Although we rely on the same Benders decomposition as in Lettovsky (1997), we model the recovery problems differently in order to have scalability. Specifically, we modify the schedule recovery problem so that it, in addition to assigning equipments and selecting flight copies, also determines an alternative maintenance schedule for each aircraft maintenance event. Given the equipment assignment and selected maintenance schedules, we split the aircraft recovery problem into two multi-commodity flow problems. While the first flow problem determines the flow of the aircraft having similar maintenance requirements, the second flow problem constructs new aircraft routes given the flow. Furthermore, we develop innovative strategies and algorithms to reduce the running time for crew recovery, which is often regarded as the bottleneck of the recovery operations. At the end, we present numerical results for multiple large disruption instances based on a real-world dataset provided by a solution vendor.

Our contributions are the following.

1. We model the aircraft recovery problem as a multi-commodity flow problem, where the commodities are defined by the disruption characteristics of the aircraft. This allows more complex and larger problems to be solved as compared to the computationally intensive column generation type technique in Peterson et al. (2012).
2. Our aircraft recovery problem is split into two subproblems. While the first subproblem determines the flow of the aircraft having similar maintenance requirements, the second subproblem assigns aircraft based on the flow provided by the first subproblem. It resolves the need for constructing feasible aircraft routes *apriori* and allows large aircraft disruption instances to be handled with ease.
3. Our crew recovery problem captures both cockpit crews and flight attendants together with reserves. It considers non-flying activities, assigns a roster to each crew member, and selects a crew rank for each crew member on each operating flight that he or she is qualified. All these aspects are not captured in the work of Peterson et al. (2012). In addition, special strategies and algorithms are developed to efficiently generate alternative rosters that are likely to be selected.

The rest of the paper is structured as follows. Section 3.2 formally presents the mathematical details for each recovery problem. Section 3.3 introduces the Benders decomposition framework. Section 3.4 elaborates the strategies and algorithms for reducing the running time. Section 3.5 reports the performance of our fully integrated solution benchmarked against a partially integrated solution used in practice. We conclude the introduction with a literature review.

3.1.1 LITERATURE REVIEW

We briefly discuss material closely related to our work starting with Lettovsky (1997), who models the fully integrated recovery problem as a set covering problem and proposes the use of Benders decomposition because of the inherent block diagonal structure of the model. He provides a conceptual Benders algorithm at a hight level. However, no computational study has been performed. The main difference with our work is that they consider a single fleet and fixed schedule (no delays). Including several fleets and changes in the flight schedule significantly complicates the problem.

Bratu and Barnhart (2006) consider only the aircraft and passenger recovery problems with emphasis on minimizing delays and cancellation costs. Dominating flight copies are identified to reduce the size of the problem. The problem does not consider crew recovery. Maintenance requirements are enforced only if they are violated after the aircraft recovery problem is solved. Since the aircraft and passenger recovery problems are combined into one problem, no integration algorithm is necessary. However, the problem remains complex, and the method is not applicable when crew recovery is required.

Chunhua (2007) discusses an application of her work on integrated planning to integrated recovery. She proposes an integrated recovery model which is based on a dated fleet assignment model and a crew duty flow model, and a preprocessor that determines the set of swappable aircraft and crew sets as well as flight delay options by some predefined thresholds. However, no implementation details and computational results are provided.

Different from mathematical programming approaches, Abdelghany et al. (2008) propose a simulation-based tool that combines a schedule simulation model with a resource assignment optimization model to minimize flight delays and cancellations during irregular operations. While the simulation model predicts a set of disrupted flights based on the severity of anticipated disruptions, the optimization model finds possible flight delays and resource swapping opportunities given the set of the disrupted flights. The tool relies on a greedy algorithm and runs in a rolling horizon manner. However, it is approximate in nature, and does not consider interaction between resources (e.g. a valid crew connection depends on the corresponding aircraft routes).

A competition was set forth by ROADEF (see Palpant et al. (2009)) in 2009. It challenged researchers to solve the integrated recovery problem without crew recovery. Bisaillon et al. (2011), who won the first place, use a large neighborhood search heuristic to find a feasible solution. It begins by randomly selecting aircraft

to recover. The aircraft solution is then improved by delaying flights and solving shortest path problems to accommodate additional disrupted passengers. When disruptions are primarily at the flight schedule level, they significantly outperform Mansi et al. (2010), the second place team, who rely on a mixed integer model to handle maintenance requirements and use an oscillation strategy to improve the initial feasible aircraft solution. However, when aircraft and airport disruptions are abundant, the neighborhood search heuristic is inferior, as it struggles to handle complicated disruptions. Adding the crew component to either method nonetheless poses significant additional complexity and requires significant changes to the approaches.

The only fully integrated recovery attempt with a computational study is the work by Peterson et al. (2012). Similar to Lettovsky (1997), they use the Benders decomposition and treat the schedule recovery problem as the master problem. In addition, they allow the aircraft and crew recovery problems to be solved separately for each equipment. However, different from Lettovsky (1997), they model the integrated recovery problem by strings, solve the crew recovery problem at the crew member level, and accommodate disrupted passengers before the aircraft and crew recovery problems are solved. All subproblems are solved by column generation to generate feasible routes, rosters, and itineraries. They compare their integrated solution with the sequential solution obtained by solving each resource recovery problem sequentially, and show that a 50% cost reduction is possible for a one-hour single-hub closure scenario.

Although the approach of Peterson et al. (2012) is similar to ours at the high level, we do not rely on string generation. In fact, because of our splitting scheme for the aircraft recovery problem, no strings or routes are generated before the aircraft recovery problem is solved. This significantly reduces our solution time without trading off optimality, and allows larger disruption instances to be solved. In crew recovery, while Peterson et al. (2012) assume that each crew member is preassigned with one rank and can only serve one equipment, we remodel the crew recovery problem to allow cross-equipment assignments and rank substitutions. This extension is essential in practice, since crews and flight attendants are normally trained to operate a set of equipments, and a captain can act as a first officer on a flight with a captain already assigned.

3.2 Problem Definition

In this section, we describe our schedule, aircraft, crew, and passenger recovery problems in the order that an operator at an OCC would have sequentially solved them during recovery. We show their dependencies

based on our integrated model and algorithm.

The process flow is illustrated in Figure 3.1. It begins by solving the schedule recovery problem (SRM) given a disrupted flight schedule. With the optimal solution of the SRM, aircraft with similar maintenance requirements are grouped. The aircraft recovery problem (ARM) is then solved for each group of the aircraft to produce feasible aircraft routes. The solution of the ARM in turn is used by the crew recovery problem (CRM). At the end, a recovering plan is produced after the passenger recovery problem (PRM) is solved. This process is run multiple times until either the solution of the SRM is stable or a time limit is reached. In each iteration, Benders cuts are generated after each subproblem is solved, and fed back to the SRM to improve its solution.

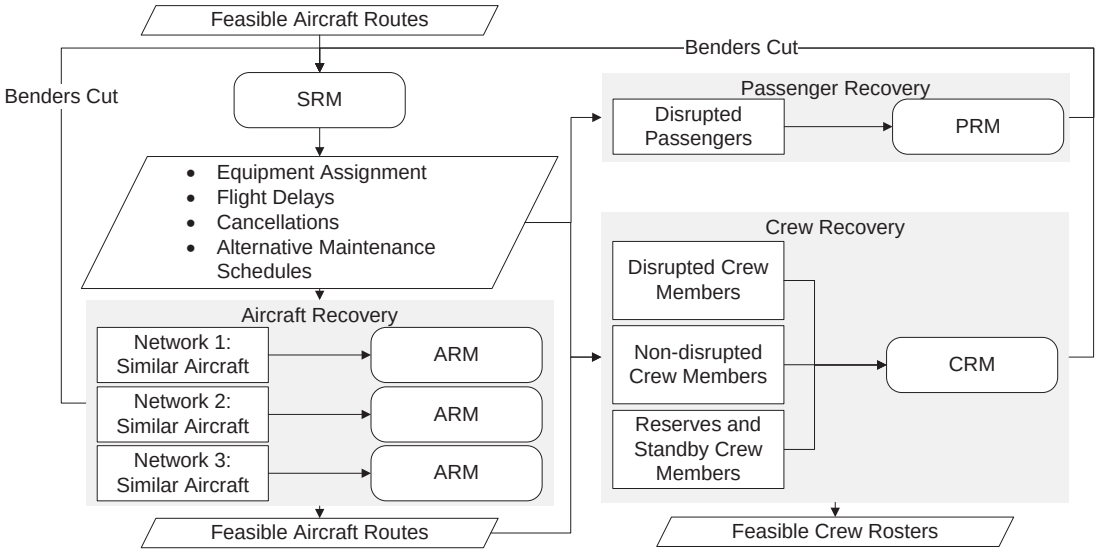

Figure 3.1: Benders decomposition for integrated recovery.

In our framework, only the fleeting, and flight cancellation and delay decisions from SRM are propagated to subsequent problems. An aircraft recovery solution can further cancel flights which affect the availability of flights for crew recovery. Since SRM does not include the flight cancellation variables in aircraft recovery, it is impossible for the crew recovery problem to pass back a Benders cut to the SRM including these variables. For this reason, our crew recovery model considers only flights selected by the SRM. The same holds for passenger recovery.

An alternative approach would be for the crew recovery problem to pass back Benders cuts to aircraft recovery, and the passenger recovery problem back to crew recovery. This will likely create a Benders algorithm with slow progress in initial iterations.

In our approach, SRM coordinates all of the remaining resource subproblems and thus makes significant progress in initial iterations. Due to limited available computational time, a Benders algorithm can perform only a dozen of iterations.

Besides the additional flight cancellation decisions, aircraft turns are another feedback between crew and aircraft. Such cuts would have to be passed back to the aircraft recovery problem by crew. Since SRM is our only 'coordinator,' we handle this heuristically by feeding the turns to crew recovery (with no cuts).

The remaining of this section is devoted to the aforementioned optimization models. We reuse notation for each resource recovery problem, and define new notation whenever necessary. For ease of exposition, only essential constraints are presented, and many business rules are omitted from the models but implemented.

3.2.1 SCHEDULE RECOVERY PROBLEM

For a flight to depart successfully, all its resources need to be available by its departure. At the beginning of the recovery process, flight copies are first generated to model delays. The schedule recovery problem (SRM) is solved to assign an equipment to each open flight. Its solution may delay or cancel flights to pave the way for better resource recovery solutions. In addition, for each flight, a different equipment may be assigned to accommodate additional disrupted passengers to further reduce the recovery cost. We model this problem as a multi-commodity flow problem, where each commodity corresponds to an equipment. The problem maximizes the total recovery bonus by delaying and canceling flights, and selecting an alternative maintenance schedule for each aircraft maintenance event (recall that an aircraft maintenance schedule is defined by the location and time that the corresponding maintenance event takes place). By allowing alternative maintenance schedules to be selected, we can split the aircraft recovery problem by aircraft with similar maintenance requirement. Details will be discussed in Section 3.2.2.

The problem is subject to the usual network flow constraints (together with additional business constraints), and follows closely the principles from the basic fleet assignment model with flight copies. We refer the reader to Klabjan (2005) or Sherali et al. (2006) to learn more about the station/time based network and ground arcs. We refer to FA as flight assignment, FC as flight coverage, GA as ground assignment, and M as maintenance. Let us define the following sets:

- $e \in E$ set of equipments,
- $a \in A_e$ set of aircraft of equipment e,

- $f \in F$ set of scheduled flights,
- $l \in L_f$ set of delayed copies for flight f (inclusive),
- $l \in L = \cup_{f \in F} L_f$ set of legs,
- $s \in S$ set of stations,
- $t \in T_s = \{1, \ldots, |T_s|\}$ set of arrival and departure time points at station s,
- $m \in M_a$ set of maintenance events of aircraft a,
- $\nu \in M_m^{Alt}$ set of alternative maintenance schedules for maintenance event m (inclusive).

The decision variables are:

- $x_{el}^{FA} = 1$ (binary) if equipment e is assigned to leg l,
- $0 \leq x_{te}^{GA} \leq |A_e|$ number of aircraft of equipment e positioned at time $t \in T_s$,
- $x_{\nu}^{M} = 1$ (binary) if alternative maintenance schedule ν is selected,
- $x_{f}^{FC} = 1$ (binary) if flight f is canceled.

Let us also define the following coefficients:

- b_{el}^{FA} bonus for assigning equipment e to leg l,
- b_{te}^{GA} bonus for positioning an aircraft of equipment e at time $t \in T_s$,
- b_{ν}^{M} bonus for choosing maintenance schedule ν,
- $\delta_{at}^{Pos} = 1_{at}^{End} - 1_{at}^{Cur}$, where $1_{at}^{Cur}(1_{at}^{End}) = 1$ if aircraft a is currently available (needs to be positioned) at time $t \in T_s$,
- $\delta_{lt}^{FA} = 1_{lt}^{Arr} - 1_{lt}^{Dep}$, where $1_{lt}^{Dep}(1_{lt}^{Arr}) = 1$ if leg l departs (arrives) at time $t \in T_s$,
- $\delta_{\nu t}^{M} = 1_{\nu t}^{Fnh} - 1_{\nu t}^{Srt}$, where $1_{\nu t}^{Srt}(1_{\nu t}^{Fnh}) = 1$ if maintenance schedule ν starts (ends) at time $t \in T_s$,
- p_{f}^{FC} penalty for canceling flight f.

Formally, the schedule recovery model (SRM) is

$$SRM^* = \max \sum_{e \in E} \sum_{l \in L} b_{el}^{FA} x_{el}^{FA} + \sum_{e \in E} \sum_{s \in S} \sum_{t \in T_s} b_{te}^{GA} x_{te}^{GA} + \sum_{e \in E} \sum_{a \in A_e} \sum_{m \in M_a} \sum_{\nu \in M_m^{Alt}} b_{\nu}^{M} x_{\nu}^{M} - \sum_{f \in F} p_f^{FC} x_f^{FC}$$

subject to

$$\sum_{l \in L} \delta_{lt}^{FA} x_{el}^{FA} + \sum_{m \in M_e} \sum_{\nu \in M_m^{Alt}} \delta_{\nu t}^{M} x_{\nu}^{M} + x_{t-1e}^{GA} - x_{te}^{GA} = \sum_{a \in A_e} \delta_{at}^{Pos} \quad t \in T_s, e \in E, s \in S \tag{3.1}$$

$$\sum_{e \in E} \sum_{l \in L_f} x_{el}^{FA} + x_f^{FC} = 1 \qquad f \in F \tag{3.2}$$

$$\sum_{\nu \in M_m^{Alt}} x_\nu^M \geq 1 \qquad m \in M_a, a \in A_e, e \in E. \tag{3.3}$$

Constraints (3.1) are the flow balancing constraints which consider the number of aircraft arriving, departing, undergoing maintenance, and on the ground. Constraints (3.2) state that each flight is either canceled or covered by a leg with an equipment assigned. Constraints (3.3) require a maintenance schedule to be selected for each maintenance event. The main difference with the standard fleet assignment model is the presence of maintenance variable x_ν^M. Furthermore, equipment positioning, slot and curfew restrictions, and gate capacity are considered, where equipment positioning constraints ensure that each station is filled with a required number of aircraft by the end of the recovery horizon, slot constraints restrict the number of equipments flying in and out from the stations at several time periods, curfew restrictions forbid certain aircraft to operate within a time window at a station, and gate capacity constraints guarantee enough gates for all arriving aircraft plus aircraft on the ground at any station and time point.

3.2.2 Aircraft Recovery Problem

Once the schedule recovery problem is solved, each maintenance event is given a maintenance schedule, and each flight may be delayed, canceled, or assigned with a different equipment. At this point, each operating flights has a unique departure time, arrival time, and equipment type. The aircraft recovery problem (ARM) is next solved to cover each operating flight and maintenance event by rerouting aircraft based on the flow of the equipments and the maintenance schedules selected. In the end, each assigned route must allow its aircraft to stay at the maintenance station for a time specified by the selected maintenance schedule, and to be positioned at a designated station by the end of the recovery horizon.

Although the aircraft recovery problem is typically modeled as a set covering problem for computational tractability, we model it differently by splitting it into two subproblems. The first subproblem is a multi-commodity flow problem with aircraft being the commodity. Aircraft are of the same commodity if they are assigned to the same equipment and have similar maintenance requirements. For every such group of aircraft, we construct an underlying network of flights that these aircraft can operate. For this reason, there is an one-to-one correspondence between the set of 'similar' aircraft and networks. Note that the aircraft recovery problem is decomposed by equipment.

Specifically, in order to group the aircraft by their maintenance requirements, a longest path problem for each aircraft is solved using a flight network constructed based on the equipment flow from the SRM. One network is constructed for each equipment. A node in a network encodes the station, time, and activity (arrival and departure), and each arc in the network is either a ground arc or corresponds to a leg. Two flight arcs are connected by a ground arc only if the aircraft connection (turn) time in between is sufficient. The cost of a path is the total flight time. While the source node corresponds to the first departure of the aircraft in the recovery time window, the sink node corresponds to the beginning of the maintenance event with its schedule selected by the SRM. If an aircraft traversing its longest path consumes all its remaining flight time before its required maintenance, the aircraft is consider *hot* and assigned to a unique aircraft network. The remaining aircraft are then grouped by their equipments and maintenance events.

The networks defined for the longest path problem are different from the networks used by the first aircraft subproblem, although the definitions of a node and arc remain the same. The networks defined above are only for grouping aircraft. They are at the equipment level, and the cost of a route only depends on the total flight time. On the other hand, the networks for the first subproblem are defined at the aircraft group level, and the cost of a route, detailed later, accounts for many other factors that allow the aircraft to be routed economically.

Once the first subproblem is solved, and the optimal aircraft flow in each group is determined, the second subproblem constructs an aircraft route for each aircraft while ensuring valid inbound and outbound connections. By splitting the problem, we reduce the complexity of the problem, so that larger disruption instances can be handled with ease. We also avoid the need to generate alternative aircraft routes on a large network, and hence, significantly reduce the running time for aircraft recovery.

FIRST ARM SUBPROBLEM

For each aircraft network, the first ARM subproblem determines the aircraft flow subject to aircraft maintenance and positioning constraints. We refer to AA as aircraft assignment, SR as scheduled route, AI as aircraft idle (unassigned), and OL as open leg with no aircraft assigned. Let us define the following sets:

- $n \in N_e$ set of aircraft networks of equipment e,
- $a \in A_n^{Net}$ set of aircraft in network n,
- $a \in A_l^{Sch}$ aircraft that uses leg l in its scheduled route,

- $t \in T_{ns} = \{1, \ldots, |T_{ns}|\}$ set of arrival and departure time points in network n at station s,

and the decision variables are:

- $y_{nl}^{AA} = 1$ (binary) if leg l is assigned to network n,
- $y_a^{SR} = 1$ (binary) if aircraft a keeps its scheduled route,
- $y_\nu^M = 1$ (binary) if maintenance schedule ν is selected,
- $0 \leq y_{tn}^{GA} \leq |A_n^{Net}|$ number of aircraft positioned at time $t \in T_{ns}$ and station s,
- $y_l^{OL} = 1$ (binary) if no aircraft is assigned to leg l.

In addition, we define the following coefficients:

- b_a^{SR} bonus for aircraft a to keep its scheduled route,
- b_{nl}^{AA} bonus for covering leg l by aircraft in network n,
- $\delta_{at}^{SR} = 1_{at}^{Arr} - 1_{at}^{Dep}$, where $1_{at}^{Dep}(1_{at}^{Arr}) = 1$ if aircraft $a \in A_n^{Net}$ is departing (arriving) at time $t \in T_{ns}$ according to its schedule route,
- $\delta_{lt}^{AA} = 1_{lt}^{Arr} - 1_{lt}^{Dep}$, $1_{lt}^{Dep}(1_{lt}^{Arr}) = 1$ if leg l departs (arrives) at time $t \in T_{ns}$,
- $\delta_{\nu t}^{M} = 1_{\nu t}^{Fnh} - 1_{\nu t}^{Srt}$, where $1_{\nu t}^{Srt}(1_{\nu t}^{Fnh}) = 1$ if maintenance schedule $\nu \in M_m^{Alt}$ for maintenance event $m \in M_a$, $a \in A_n^{Net}$ starts (ends) at time $t \in T_{ns}$,
- p_f^{OL} penalty for not covering flight f.

Given a set of flight assignments $\overline{\mathbf{x}}^{FA}$ and a set of maintenance schedules $\overline{\mathbf{x}}^{M}$ produced by the SRM, the first ARM subproblem for equipment $e \in E$ is

$$FARM_e^*\left(\overline{\mathbf{x}}^{FA}, \overline{\mathbf{x}}^{M}\right) = \max \sum_{n \in N_e} \sum_{l \in L} b_{nl}^{AA} y_{nl}^{AA} + \sum_{a \in A_e} b_a^{SR} y_a^{SR} + \sum_{m \in M_e} \sum_{\nu \in M_m^{Alt}} b_\nu^M y_\nu^M - \sum_{f \in F} p_f^{OL} \left(1 - \sum_{l \in L_f} y_l^{OL}\right)$$

subject to

$$\sum_{n \in N_e} y_{nl}^{AA} + \sum_{a \in A_l^{Sch}} y_a^{SR} + y_l^{OL} = \bar{x}_{el}^{FA} \qquad l \in L \tag{3.4}$$

$$\sum_{l \in L} \delta_{lt}^{AA} y_{nl}^{AA} + \sum_{a \in A_n^{Net}} \delta_{at}^{SR} y_a^{SR}$$

$$+ \sum_{m \in M_e} \sum_{\nu \in M_m^{Alt}} \delta_{\nu t}^{M} y_{\nu}^{M} + y_{t-1n}^{GA} - y_{tn}^{GA} = \sum_{a \in A_n^{Net}} \delta_{at}^{Pos} \quad t \in T_{ns}, n \in N_e, s \in S \tag{3.5}$$

$$y_{\nu}^{M} = \bar{x}_{\nu}^{M} \quad \nu \in M_m^{Alt}, m \in M_a, a \in A_e. \tag{3.6}$$

Constraints (3.4) ensure that each leg is either covered by a network, an aircraft on its scheduled route, or a cancellation event. Constraints (3.5) balance flows in the network with arrival legs, departure legs, aircraft on the ground, and aircraft that undergo maintenance. Constraints (3.6) require that each maintenance schedule selected by the SRM is covered. In addition, aircraft remaining flight time, aircraft positioning, maintenance resource, and station capacity requirements are implemented but not listed.

SECOND ARM SUBPROBLEM

The second ARM subproblem constructs routes for each individual aircraft while maximizing the assignment bonus. In this case, the commodity in the multi-commodity flow problem is an aircraft. In the network, each node is a leg, and an arc connects two nodes only if the turn time is satisfied. In addition, there is an artificial leg corresponding to the selected maintenance schedule in the first ARM subproblem. At the end, the problem produces a new aircraft route for each aircraft that are not idle. The problem is independently solved for each network $n \in N_e$ and $e \in E$. Let us define the following sets:

- C_{al}^{Out} set of feasible outbound legs from leg l if leg l is served by aircraft a,
- C_{al}^{In} set of feasible inbound legs to leg l if leg l is served by aircraft a,
- $L_n(\overline{\mathbf{y}} = (\overline{\mathbf{y}}^{AA}, \overline{\mathbf{y}}^{SR}))$ set of legs assigned to network n given an optimal solution $\overline{\mathbf{y}}$ to the FARM (this set also includes the maintenance artificial leg),

and the decision variable is w_{al}, a binary variable that indicates if leg l is covered by aircraft a. In addition, we define the following coefficients:

- $1_{al}^{Cur}(1_{al}^{End}) = 1$ if leg l is the first (last) leg of aircraft a,
- b_{al} bonus for covering leg l by aircraft a.

Formally, the second ARM subproblem for aircraft network $n \in N$ is

$$SARM_n^*(\overline{\mathbf{y}}) = \max \sum_{a \in A_n^{Net}} \sum_{l \in L_n(\bar{y})} b_{al} w_{al}$$

subject to

$$\sum_{a \in A_n^{Net}} \sum_{l' \in C_{al}^{Out}} w_{al'} + \sum_{a \in A_n^{Net}} 1_{al}^{Cur} w_{al} = 1 \quad l \in L_n(\overline{\mathbf{y}}) \tag{3.7}$$

$$\sum_{a \in A_n^{Net}} \sum_{l' \in C_{al}^{In}} w_{al'} + \sum_{a \in A_n^{Net}} 1_{al}^{End} w_{al} = 1 \quad l \in L_n(\overline{\mathbf{y}}) \tag{3.8}$$

$$\sum_{l' \in C_{al}^{Out}} w_{al'} - \sum_{l' \in C_{al}^{In}} w_{al'} + 1_{al}^{End} w_{al} - 1_{al}^{Cur} w_{al} = 0 \quad l \in L_n(\overline{\mathbf{y}}), a \in A_n^{Net} \tag{3.9}$$

Constraints (3.7) and (3.8) require each leg to be covered by either an aircraft on the ground at the beginning or end of the time horizon, or an aircraft from an inbound or outbound connection. Constraints (3.9) are the flow balance constraints for each aircraft. In our implementation, we also include the starting and ending positioning requirements for each aircraft, and we impose a penalty if a new route cannot be constructed for an operating aircraft.

3.2.3 Crew Recovery Problem

Given a set of aircraft along with their maintenance feasible routes, disrupted crew members are recovered by assigning them to different rosters. We reiterate here that SRM provided feedback to crew recovery albeit the fact that ARM might have canceled additional flights. Thus the set of all available flights and the underlying departure times are based on the SRM solution. On the other hand, the second ARM finds individual routes and thus aircraft turns. These are used in the crew recovery problem, but there is no feedback back to the SRM in terms of aircraft turns.

Before the crew recovery problem is solved, a set of feasible rosters is generated for each crew member. This is discussed in detail in Section 3.4.2. Roster generation considers all work rules, positioning requirements, fleet compatibility, rank substitutability, and potential use of reserves and standbys. In this section we assume that each crew member has a set of feasible crew rosters, and next we discuss the model that selects a single roster for each crew member.

A *qualification* is a set of similar equipments that a crew member is trained to operate. A *rank* of a crew member can be captain, first officer, or flight attendant. A captain can serve as a first officer if a different captain has already been assigned to a particular flight. However, a first officer cannot serve as a captain, and flight attendants are not substitutable by other ranks. We refer to RC as roster coverage, DH as deadhead, and OF as operating flights. Let us define the following sets:

- $d \in D$ set of deck types (e.g. flight deck and cabin),
- $r \in R^d$ set of crew ranks for a given deck type d, (If d refers to flight deck, R^d is typically $\{CA, FO\}$ for d being flight deck, and R^d is $\{FA\}$ for d being cabin, where CA refers to captain, FO refers to first officer, and FA refers to flight attendant),
- $q^e \in Q^E$ set of qualifications, and each q^e is a subset of E,
- $q^r \in Q^{R^d} = \{(CA, FO), (FO)\}$ if d refers to flight deck, and $Q^{R^d} = \{(FA)\}$ if d is cabin (these sets represent all possible rank substitution),
- $c \in C$ set of crew members,
- $c \in C^{Opr} \subseteq C$ set of crew members on duty excluding reserves and standbys,
- $v \in V_c$ set of all rosters for crew member c,
- $l \in L_v^{Crw}$ set of legs used by roster v,

and the decision variables are:

- $z_l^{DH} \geq 0$ number of deadheads on leg l,
- $z_v^{RC} \geq 0$ number of crew members covering roster v,
- $z_{el}^{OF} = 1$ (binary) if leg l assigned with equipment e is operated.

In addition, we define the following coefficients:

- $1_{erc}^{Qal} = 1$ if $e \in q^e \in Q^E$, $r \in q^r \in Q^R$, and $(q^e, q^r) = c \in C$,
- b_v^{RC} bonus for covering roster v,
- p_l^{Cnl} penalty for canceling leg l,
- p_l^{DH} penalty for deadheading on leg l,
- n_{er}^{Rnk} number of crew members of rank r required on equipment e,
- M a large number.

Given a feasible solution $\overline{\mathbf{x}}^{FA}$ to the SRM, the crew recovery model (CRM) is

$$CRM^*(\overline{\mathbf{x}}^{FA}) = \max \sum_{c \in C} \sum_{v \in V_c} b_v^{RC} z_v^{RC} - \sum_{l \in L} p_l^{DH} z_l^{DH} - \sum_{l \in L} p_l^{Cnl} \left(1 - \sum_{e \in E} z_{el}^{OF}\right)$$

subject to

$$\sum_{c\in C}\sum_{v\in V_c: l\in L_v^{Crw}} 1_{erc}^{Qal} z_v^{RC} - \sum_{r'\succeq r} z_{el}^{OF} n_{er'}^{Rnk} \geq 0 \qquad e\in E, r\in R^d, d\in D, l\in L \tag{3.10}$$

$$\sum_{c\in C}\sum_{v\in V_c: l\in L_v^{Crw}} z_v^{RC} - \sum_{e\in E}\sum_{d\in D}\sum_{r\in R^d} z_{el}^{OF} n_{er}^{Rnk} = z_l^{DH} \qquad l\in L \tag{3.11}$$

$$\sum_{c\in C}\sum_{v\in V_c: l\in L_v^{Crw}} z_v^{RC} - M\sum_{e\in E} z_{el}^{OF} \leq 0 \qquad l\in L \tag{3.12}$$

$$\sum_{v\in V_c} z_v^{RC} = 1 \qquad c\in C^{Opr} \tag{3.13}$$

$$z_{el}^{OF} = \overline{x}_{el}^{FA} \qquad e\in E, l\in L. \tag{3.14}$$

Constraints (3.10) are the flight coverage constraints. For any operating flight of equipment e (i.e. $z_{el}^{OF} = 1$), the minimum number of crew members has to be satisfied for each rank. Summation condition $r' \succeq r$ represents the substitutability of rank r' to rank r, but not vice versa (e.g. if r refers to first officer, then the summation is over both captain and first officer, and if r refers to captain, then the summation is only over captain). Constraints (3.11) count the number of crew members that are overflown (deadhead) on each leg. These constraints consider all equipments that the crew member is trained to operate, and allow a captain to dynamically change his/her rank depending on if a captain has already been assigned to a particular flight. Constraints (3.12) ensure that no crews are on any canceled flight. Constraints (3.13) imposes that each crew member on duty must be assigned a roster. Constraints (3.14) state that if the flight has been canceled by the SRM, then the flight is unavailable to route crew members. In addition, the bonus coefficient b_v^{NC} accounts for the cost of utilizing reserves or standbys if $v \in V_c$, and c refers to a reserve or standby crew member. A penalty is imposed if any crew member cannot satisfy his or her positioning requirements by the end of the recovery horizon.

3.2.4 Passenger Recovery Problem

After solving the schedule, aircraft, and crew recovery problems, any (delayed) flight with an aircraft and crew assigned can be used to carry disrupted passengers. Passengers are disrupted if they cannot fly to their destination on time. We model the passenger recovery problem as an assignment problem, which assigns disrupted passengers of the same booking to different itineraries subject to flight capacity constraints.

We refer to PA as passenger assignment and SP as disrupted passenger, and define the following:

- $b \in B$ set of bookings with a disrupted itinerary,
- $i \in I$ set of all itineraries constructed based on all flight copies that are generated in the preprocessing step to SRM,
- $i \in I_b \subseteq I$ set of itineraries for booking b that can serve as a substitute to the original itinerary,
- $l \in L_i^{Pax}$ set of legs used by itinerary i,
- b_{ib}^{PA} bonus for carrying a passenger in booking b by itinerary i,
- K_{el} number of remaining empty seats on leg l that can be used to carry disrupted passengers if equipment e is assigned,
- p_b^{SP} penalty for stranding passengers in booking b,
- n_b^{Pax} number of passengers in booking b.

The decision variable is u_{ib}^{PA}, which is the number of passengers in booking b assigned to itinerary $i \in I$. Given $\overline{\mathbf{x}}^{FA}$ a feasible solution to SRM, the passenger recovery model (PRM) is

$$PRM^*(\overline{\mathbf{x}}^{FA}) = \max \sum_{b \in B} \sum_{i \in I_b} b_{ib}^{PA} u_{ib}^{PA} - \sum_{b \in B} p_b^{SP} \left(n_b^{Pax} - \sum_{i \in I_b} u_{ib}^{PA} \right)$$

subject to

$$\sum_{b \in B} \sum_{i \in I_b : l \in L_i^{Pax}} u_{ib}^{PA} \leq \sum_{e \in E} K_{el} \bar{x}_{el}^{FA} \quad l \in L, \tag{3.15}$$

where constraints (3.15) are the flight capacity constraints. If a flight is canceled (i.e. $\sum_{e \in E} \bar{x}_{el}^{FA} = 0$), no passengers can be on board.

3.3 Benders Decomposition

In Section 3.2, we introduced the resource recovery problems that the OCC operator sequentially solves during recovery operations. While the models were presented in a sequential manner modeling current decision making process (while not providing the identical flow of information), it is easy to write a complete integrated model by using the exhibited notation.

Figure 3.2 shows our Benders decomposition framework for solving the integrated recovery problem. In order to get dual values, LP relaxations of subproblems are solved. In each iteration, before the LP relaxation of ARM/CRM/PRM is solved, feasible rosters are generated. Optimality cuts are constructed

based on the dual of the constraints that tie the SRM and the resource recovery problem together (details to be discussed later). The cut is then added back to the SRM for the next Benders iteration. Optimality cuts ensure that the solution of the SRM will not lead to a higher integrated recovery cost (lower recovery bonus in our case). Since the resource recovery problems are solved as LP relaxations, a post-processing step (branch-and-bound) based on the schedules and routes obtained is required to obtain an integer solution.

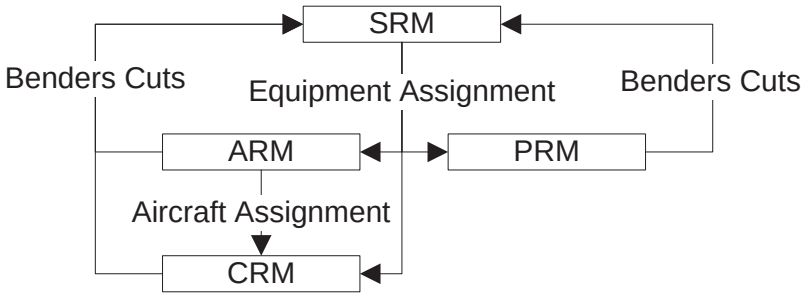

Figure 3.2: Benders decomposition for integrated recovery.

For aircraft recovery, the LP relaxation of the FPRM is solved to yield the dual solution required to construct the optimality cut. The procedure then computes an optimal integer solution, which is then fed to the SARM to construct new aircraft routes. Let $\{\pi_{el}^{FA}\}_{l\in L}$ be the duals of (3.4), $\{\pi_{m}^{M}\}_{m\in M_e}$ the duals of (3.6), and let $\theta_e^{ARM^*}$ the Benders dummy variable for equipment $e \in E$ to be added to the SRM. Formally, the optimality cuts are

$$FARM_e^*(\overline{\mathbf{x}}^{FA}, \overline{\mathbf{x}}^{M}) - \sum_{l\in L} \pi_{el}^{FA}\overline{x}_{el}^{FA} - \sum_{a\in A_e}\sum_{\nu\in M_a} \pi_{\nu}^{M}\overline{x}_{\nu}^{M}$$
$$+\sum_{l\in L} \pi_{el}^{FA}x_{el}^{FA} + \sum_{a\in A_e}\sum_{\nu\in M_a} \pi_{\nu}^{M}x_{\nu}^{M} \geq \theta_e^{ARM^*} \text{ for } e \in E.$$

We subtract the total dual amount from the primal objective value, and use the duals to pair with the decision variables of the master problem. On the other hand, if the relaxed problem is infeasible, slack variables are added to (3.6). The objective of the FARM is now to minimize the sum of the slacks. The duals are now interpreted as extreme rays that guide the FARM to a feasible solution. The formula for the feasibility cut is essentially the same as that for the optimality cut except that $\theta_e^{ARM^*}$ is replaced by 0. Infeasibility can occur if no route can be assigned to an aircraft. If $\overline{FARM}_e^*(\overline{\mathbf{x}}^{FA}, \overline{\mathbf{x}}^{M})$ is the optimal value of the feasibility problem for an equipment $e \in E$, then the ARM feasibility cut is

$$\overline{FARM}_e^*(\overline{\mathbf{x}}^{FA}, \overline{\mathbf{x}}^M) - \sum_{l \in L} \pi_{el}^{FA} \overline{x}_{el}^{FA} - \sum_{a \in A_e} \sum_{\nu \in M_a} \pi_\nu^M \overline{x}_\nu^M + \sum_{l \in L} \pi_{el}^{FA} x_{el}^{FA} + \sum_{a \in A_e} \sum_{\nu \in M_a} \pi_\nu^M x_\nu^M \geq 0.$$

For crew recovery, the LP relaxation of the CRM is solved to obtain the dual values associated with (3.14), which are then used to construct the optimality cut. Let $\{\lambda_{el}^{FA}\}$ be the duals of (3.14), and θ^{CRM^*} be the Benders dummy variable. The CRM optimality cut is

$$CRM^*(\overline{\mathbf{x}}^{FA}) - \sum_{l \in L} \sum_{e \in E} \lambda_{el}^{FA} \overline{x}_{el}^{FA} + \sum_{l \in L} \sum_{e \in E} \lambda_{el}^{FA} x_{el}^{FA} \geq \theta^{CRM^*}.$$

Similarly to the ARM, when CRM is infeasible due to inability to assign a roster to an on-duty crew member, we solve the corresponding feasibility problem. A slack variable is added to (3.14). The objective function is changed to minimize the sum of the slacks, and the duals become the extreme rays.

Note that PRM is always feasible. We only need to consider optimality cuts. Let $\{\gamma_l^{FA}\}$ be the duals of (3.15), and θ^{PRM^*} the Benders dummy variable. The PRM optimality cut is

$$PRM^*(\overline{\mathbf{x}}^{FA}) - \sum_{l \in L} \sum_{e \in E} \gamma_l^{FA} \overline{x}_{el}^{FA} + \sum_{l \in L} \sum_{e \in E} \gamma_l^{FA} x_{el}^{FA} \geq \theta^{PRM^*}.$$

Finally, the master problem becomes

$$\max SRM + \sum_{e \in E} \theta_e^{ARM^*} + \theta^{CRM^*} + \theta^{PRM^*}$$

subject to all the aforementioned Benders cuts and non-negativity of all the Benders dummy variables. In each iteration, depending on the feasibility of the resource recovery problems, feasibility and optimality cuts are added. The cuts are kept over all Benders iterations, and can also be iteratively removed based on the validity check in Peterson et al. (2012).

3.4 Implementation

3.4.1 Flight Copy Generation

In order to successfully recover the resources, flights can either be delayed or canceled. While flight cancellation is usually the last resort as the associated cost could be steep, delaying flights is very common.

In order to delay flights, flight copies are generated at the beginning of the recovery process. For each resource, if a misconnection is found in its route/roster/itinerary, a delay copy of the disrupted flight is created by delaying the disrupted flight minimally to resolve the misconnection. If the flight is delayed sufficiently long to induce a misconnection to the outbound flight operated by the same resource, then a delayed flight is similarly generated for the outbound flight.

3.4.2 CREW ROSTER GENERATION

To recover disrupted crew members, a roster generator is run to generate alternative rosters for each crew member who may be affected (directly or indirectly) by the disruption. A disrupted crew has at least one disrupted flight, and his or her roster cannot be repaired by using the same flights. It relies on a segment network that ensures valid segment connections and a customized breath-first search with priority queue (BFSPQ) to generate the first K feasible rosters with minimal cost. While generating rosters, the BFSPQ simultaneously accounts for any work rule of an accumulative nature (e.g. total flight time, total duty time, total number of calendar days, etc) over all levels (duty, pairing, and roster).

In the beginning of roster generation, we first build a segment network to ensure valid segment connections. In the segment network, each node encodes the location, time, and activity (arrival and departure). Two nodes are connected if they either belong to the same segment or exhibit a valid connection that satisfies both the minimum and maximum crew connection times.

To generate feasible rosters, we run the BFSPQ, which is based on the traditional breath-first search algorithm by adding legality checks on work rules. During the execution of the algorithm, the work rules are checked on the fly, and a priority queue is used to sort partially constructed rosters by their costs, which are the same as the coefficients in the objective of the CRM, and consist of the deadhead cost and the flight coverage bonus. Given a disrupted crew member and underlying original roster $\{s_1, \ldots, s_R\}$ of segments, the inputs to BFSPQ are $S^b = \{s_1, \ldots, s_{i-1}\}$, $S^e = \{s_{j+1}, \ldots, s_R\}$, and $\mathcal{S}$, which is a set of segments that the BFSPQ can use to construct alternative rosters, and it consists of all flights that fall within the time window defined by the arrival time of segment s_{i-1} and the departure time of segment s_{j+1}. Here, S^b and S^e are sequences of undisrupted segments. The algorithm starts from the same station as s_{i-1} and finds all possible connectible segments in $\mathcal{S}$. When legality is checked, if adding a flight violates only the maximum flying time, the flight is then used to deadhead the crew member in question. If appending S^e to the last segment of the partially constructed roster yields a feasible roster, the roster is stored. When a connectible

segment is found, the cost of the partially constructed roster is updated and added to the priority queue. The algorithm repeats until either no more connectible segments are found in $\mathcal{S}$ or the maximum number of rosters to be generated is reached. At the end, the algorithm returns the set of K feasible rosters with minimum cost.

The major difference between the BFSPQ and a K shortest path algorithm is that the K shortest path algorithm generally relies on augmenting the shortest path, backtracking from the end, or reversing edges to reduce its running time, and hence, is neither simple nor efficient to check complicated work rules on the fly. The implementation of Yen's algorithm in Martins et al. (2000) and the K shortest path algorithm in Eppstein (1999) are two such examples. We incorporated both implementations with rule checking. However, the resulting longer running times render them unsuitable for our implementation. In the next section, we discuss how feasible schedules are generated for each crew category.

ALGORITHMIC STRATEGY

The previous section specifies how to generate rosters for disrupted crew members. To obtain high quality solutions, rosters of other crew members must be considered. Figure 3.3 illustrates our general algorithmic strategy for roster generation, which is the bottleneck of the recovery process.

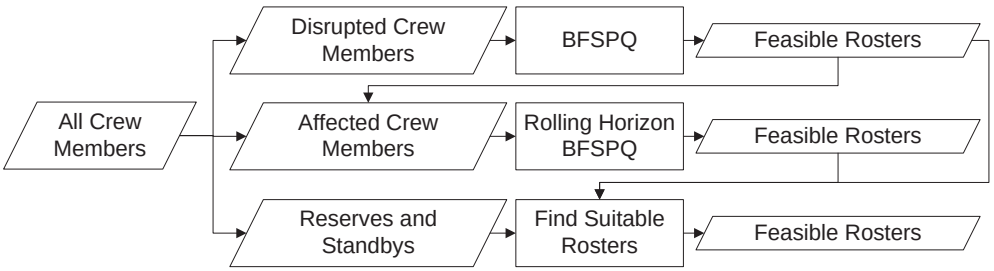

Figure 3.3: Roster generation process.

We first categorize crew members by disrupted crew members, affected crew members, and reserve crew members to reduce the running time for generating alternative rosters. Disrupted crew members are crew members who rosters have at least one disrupted flight, and we generate rosters for them first. Affected crew members are crew members whose rosters includes at least one flight that is part of a roster generated for a disrupted crew member. The inclusion of the affected crew members further reduces the recovery cost by allowing parts of their rosters to be swapped with disrupted rosters, and hence, provides more recovery

options for the disrupted crew members. Reserve crew members are necessary when no other option exists. They do not have a roster and can be called to operate on understaffed flights that otherwise cannot depart.

For each crew category, a different strategy is applied to generate rosters. For each disrupted crew member, we first directly apply the BFSPQ as described in the previous section with all connectible segments. This algorithm starts with the shortest string in which s_i and s_j are the first and last disrupted segments. Then, string S^d, which consists of all segments in the rosters not in $S^b \cup S^e$ is iteratively extended by appending segments from S^b and S^e until the required number of alternative rosters is reached.

Now, the generated rosters of disrupted crew members include segments of other crew members. All such crew members are considered affected. For each affected crew member, S^d is defined as the shortest sequence of segments on his or her roster that includes segments on generated rosters for disrupted crew members. Since the number of alternative rosters heavily depends on the length of S^d, which could be very long when it contains many flights that can be used to carry many different disrupted crew members, the BFSPQ is iteratively applied with a rolling horizon time window for each affected crew member.

The rolling horizon version of BFSPQ is primarily designed to generate alternative feasible rosters that minimally deviate from the original roster of the crew member in question, while mitigating the inefficiency of the BFSPQ when S^d contains many segments. As the rosters of the affected crew members are originally feasible, the aim of the rolling horizon BFSPQ is to efficiently generate some alternative rosters that may improve the overall solution quality of the CRM. As a result of using the rolling time window, the generated rosters are biased toward short local recovery. This contrasts to the direct application of BFSPQ on the disrupted crew members whose alternative rosters are ensured of quality but take the algorithm a longer time to find.

Let b be the index of the last segment before the first affected segment, let e be the index of the first segment after the last affected segment, let $\mathcal{R}$ be the set of alternative rosters generated thus far, and let φ be the threshold on the number of generated rosters above which the beginning of the rolling horizon time window slides forward. The rolling horizon algorithm starts with a small rolling horizon time window, where its beginning time is the arrival time of the b^{th} segment, and its ending time is the departure time of the $(b+1)^{th}$ segment. The end of the rolling horizon time window is iteratively extended until φ many rosters are found. Then, the rolling horizon time window is reset with the $(b+1)^{th}$ and $(b+2)^{th}$ segments, and the procedure repeats. The algorithm stops when the end of the rolling horizon time window meets the departure time of the e^{th} segment. At the end, we restrict $\mathcal{S}$ to include only segments in the alternative

rosters generated thus far, and the BFSPQ is run again with this new set of segments to obtain the first K alternative rosters. The algorithm is summarized in Algorithm 4.

Algorithm 4 Rolling Horizon BFSPQ

Require: $\{s_1, \ldots, s_R\}$, b, e

Set $\bar{b} = b$, $\bar{e} = b + 1$, $\mathcal{R} = \bar{\mathcal{R}} = \emptyset$
while $\bar{e} \leq e$ **do**
  Set
  - $S^b = \{s_1, \ldots, s_{\bar{b}}\}$ and $S^e = \{s_{\bar{e}}, \ldots, s_R\}$
  - $\mathcal{S}$ to include all segments that depart after $s_{\bar{b}}$ and arrive before $s_{\bar{e}}$
  - $\bar{\mathcal{R}} = BFSPQ(S^b, S^e, \mathcal{S})$

  if $|\bar{\mathcal{R}}| > \varphi$ or $\bar{e} = e$ **then**
    Move all strings in $\bar{\mathcal{R}}$ to $\mathcal{R}$
    $\bar{b} = \bar{b} + 1$
  end if
  $\bar{e} = \bar{e} + 1$
end while
Set
- $S^b = \{s_1, \ldots, s_b\}$ and $S^e = \{s_e, \ldots, s_R\}$
- $\mathcal{S}$ to include only segments in $\mathcal{R}$ and exclude segments in $S^b \cup S^e$

return $BFSPQ(S^b, S^e, \mathcal{S})$

For each reserve crew member, we, instead of running the BFSPQ, construct feasible rosters based on rosters already generated. For each feasible roster generated for a disrupted or affected crew member, a feasible sequence (if exists) is extracted and assigned to the reserve crew member if all work rules are satisfied. We do not attempt to construct feasible rosters by applying the BFSPQ directly, as we experienced computationally that running the BFSPQ to generate rosters for the reserve crew members demands a long running time without significantly improving the objective.

To further reduce the running time for generating rosters, we identify the *dominating* crew member, and feasible rosters are only generated for the dominating crew member. A crew member is dominating if his/her disrupted or affected sequence of segments completely covers disrupted or affected segments of other crew members. Such dominating relationship always exists, as a disruption affecting a roster of a crew member should also affect the rosters of other crew members. Once feasible rosters are generated for the dominating crew member, all segments used for generating the rosters are extracted and used to generate feasible rosters for each dominated crew member by running the BFSPQ.

Lastly, the BFSPQ is highly parallelized over disrupted and affected crew members. Since the lengths of the disrupted or affected rosters are not the same, work load needs to be balanced to fully harvest the benefit

from multi-thread computing. Toward this end, we sort the dominating crew members by the number of segments in the disrupted or affected region. By balancing the load, we are able to reduce the running time by about 30%.

3.5 COMPUTATIONAL STUDY

In this section, we provide a comprehensive computational study. We compare our fully integrated approach with the partially integrated approach, which follows the same algorithm and is based on the same implementation, except that only the schedule, aircraft, and passenger recovery problems are included in the Benders framework. After terminating the Benders algorithm, a post-processing step is applied to find an integral aircraft routing solution. Then, CRM is solved exactly once by using the same model and implementation. Since several airlines and a vendor solve the airline recovery problem by this partially integrated approach, we compare the fully integrated approach with the current state-of-the-art in practice.

This comparison is different from Peterson et al. (2012) who study the improvement of the fully integrated solution over a solution obtained by solving all resource recovery problems sequentially. Our results, on the other hand, show if integrating the CRM, frequently regarded as the bottleneck in the recovery process, is a profitable direction in terms of its associated marginal improvement and running time increase.

Because of the lack of itinerary data from our data provider, we only know the number of passengers and the associated average fare at the flight level. As a result, we do not solve the PRM directly, but we penalize passenger delays and itinerary cancellations in both our fully integrated solution and the partially integrated solution. A passenger is considered delayed if the new arrival time of his/her flight is later than the original arrival time. A passenger itinerary is canceled if the corresponding flight is canceled and no alternative flight is provided. The penalties are subtracted from the objectives of the ARM and CRM.

All tested scenarios are constructed based on a real world data set with all costs and business rules. It is a heavy hub-and-spoke network with 2 hubs, 3 equipments, 146 aircraft, 650 scheduled flights, 1300 crew members, and 160 reserved crew members. The tested scenarios are based on closing the busiest hub by one and two hours, and are summarized in Table 3.1 with their abbreviations listed in parenthesis. They are selected to cover the busiest hours of the targeted airport. A scenario is considered easy when the number of disrupted crew members at the end of the Benders algorithm in the partially integrated approach is small. It turns out, as expect, that 1-hour scenarios are, in general, easier to be recovered, and the most challenging

scenario is 12_2. The recovery time window for all the scenarios starts from the end of the disruption until 6 a.m. next day. Hence, the later the disruption, less time we have for recovery.

Table 3.1: Single-hub closure scenarios

1-hour Scenarios	2-hour Scenarios
(12_1) 12:00 p.m. - 1:00 p.m.	(11_1) 11:00 p.m. - 1:00 p.m.
(2_3) 2:00 p.m. - 3:00 p.m.	(12_2) 12:00 p.m. - 2:00 p.m.
(5_6) 5:00 p.m. - 6:00 p.m.	(2_4) 2:00 p.m. - 4:00 p.m.
	(5_7) 5:00 p.m. 7:00 p.m.

Table 3.2 lists some of the most important cost parameters. Although the list is by no means comprehensive, it contains important costs that drive the performance of the recovery solution. The first two bonuses incentivize aircraft to be on their original routes and their mandatory maintenances to be covered. The last two bonuses encourage crew members to cover as many flights as possible, and especially their originally scheduled flights. On the other hand, deadhead usage is highly discouraged. A penalty is imposed for any crew member not ending at his/her designated station by the end of the recovery horizon. A fixed cost is incurred for any reserve crew called. Passenger goodwill is discounted from the recovery bonus when passengers are delayed, or itineraries are canceled. Additional penalty based on the average fare of the flight is added for each passenger itinerary canceled.

Table 3.2: Important cost parameters

	Bonus		Penalty
Original Route Coverage	1,000	Deadhead	1,000
Maintenance Coverage	812	Crew Ending	20,000
Original Flight Coverage	500	Reserve	1,500
Other Flight Coverage[a]	150	Passenger Goodwill	1,000

We have conducted our computational study on a server with a 64-bit Window Server 2008 operating system. Its CPU is Intel Xeon X5560 with 2 processors and six 2.8 GHz cores per processor. ILOG Cplex 12.3 is used for optimization, and OpenMP 2.0 is used for parallelization within the Visual Studio 2008 C++ implementation. However, for optimal performance, we only utilize four cores due to the restriction of the cache memory per core. In our implementation, we stop the algorithm after 8 Benders iterations, since this is the iteration at which the integrated solution is stable, and the running time is reasonable. The number

[a]Bonus for covering flights that are not part of the original roster of a crew member.

of rosters to be generated is limited at 500 per each disrupted crew member and 300 per each affected crew member.

Figure 3.4 shows the percentages of the objective, which we call revenue, improvements when the integrated solution is used instead of the partially integrated solution.

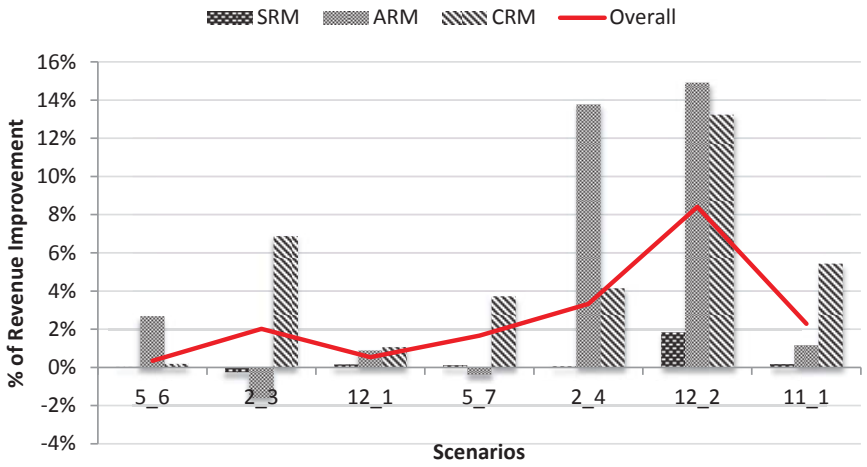

Figure 3.4: Percentages of revenue improvements for the SRM, ARM, CRM, integrated system.

We have the percentages broken down for each resource recovery problem. The objective value of the ARM is expected to decrease after the CRM is integrated, since trade-offs between the resource recovery problems are inevitable. However, we observe that by adding the CRM optimality cuts, we improve not only the performance of the CRM, but also the performance of the ARM in many scenarios. This is due to the fact that rosters are generated based on feasible routes given by the ARM solution. Thus, the CRM optimality cuts should also guide the ARM to a higher recovery bonus. On the other hand, the objective values of the SRM minus the optimal values of the Benders dummy variables show no improvement. This implies that the SRM is degenerated and many alternative optimal solutions exist. The integrated recovery algorithm iteratively finds from a pool of optimal SRM solutions a solution that yields the best recovery plan for both the aircraft and crews. The line on the figure shows the percentage of overall improvements. While improvements on 1-hour scenarios are not substantial, improvements on 2-hour scenarios are significant (approximately 2% or more). The limited improvements observed from the 1-hour scenarios can be due to the fact that disruptions in the scenarios can be easily fixed, e.g. after solving the SRM and ARM by Benders decomposition, the remaining disrupted crew members are only a few. Hence, the benefits from integrating the CRM are limited. For confidentiality reasons, we present only relative improvements. The

improvement in absolute value is measured based on the parameters provided by a solution vendor, and it varies from $\$50,000$ to 1 million dollars per scenario. Figures 3.5 and 3.6 summarize two important cost drivers.

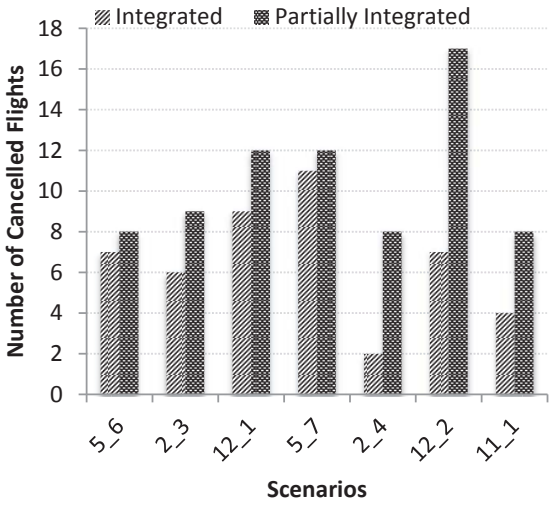

Figure 3.5: Number of canceled flights.

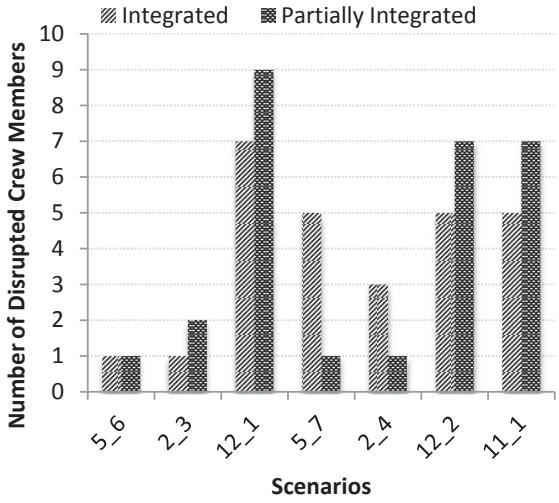

Figure 3.6: Number of disrupted crews.

Figure 3.5 compares the numbers of the canceled flights from the two approaches. In all scenarios, our approach reduces the number of canceled flights in the range of 1 to 10, where 10 fewer canceled flights is from scenario 12_2, which covers the busiest hour of the airport on the selected day.

Figure 3.6 shows the numbers of the disrupted crew members. The change in the number of disrupted crew members across scenarios ranges from -4 to 2, where 4 more disrupted crew members corresponds to scenario 5_7, whose ARM solution is changed to a degree that it allows the CRM to significantly increase the assignment bonus at the expense of having more disrupted crew members.

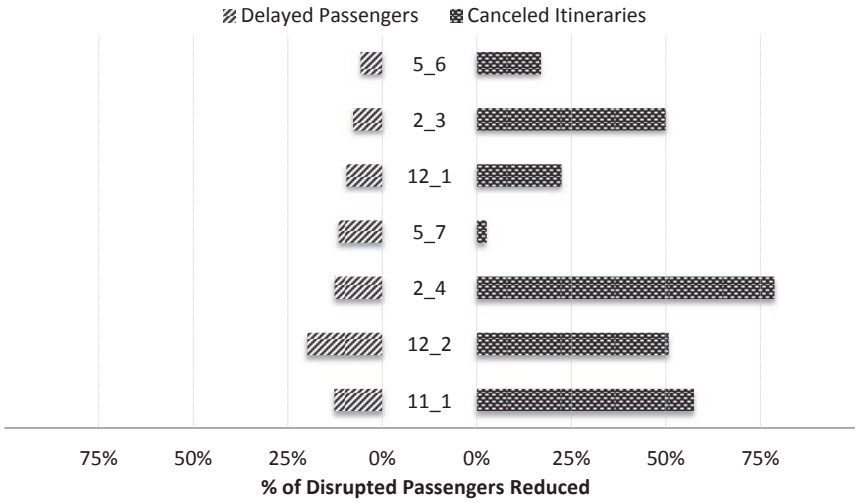

Figure 3.7: Percentage of delayed passengers and canceled itineraries.

Figure 3.7 shows the reductions of the delayed passengers and canceled itineraries in percentage. When the fully integrated approach is applied, delayed passengers are reduced by 10% on average. The percentage is higher when the disruption scenario is more challenging. On the other hand, we observe that canceled itineraries are reduced by about 25% in the 1-hour scenarios, and about 55% in the 2-hour scenarios with the exception that scenario 5_7 reduces the value by only about 3%. Both delayed passengers and canceled itineraries are significantly reduced due to partly fewer canceled flights and partly the positive effect of the Benders framework that allows the equipment, aircraft, and crew assignments to be iteratively adjusted for a better recovery solution.

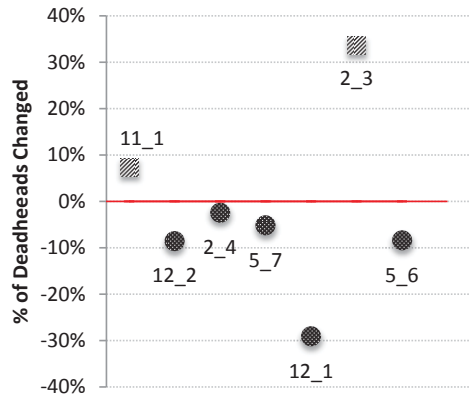

Figure 3.8: Percentage of deadheads reduced.

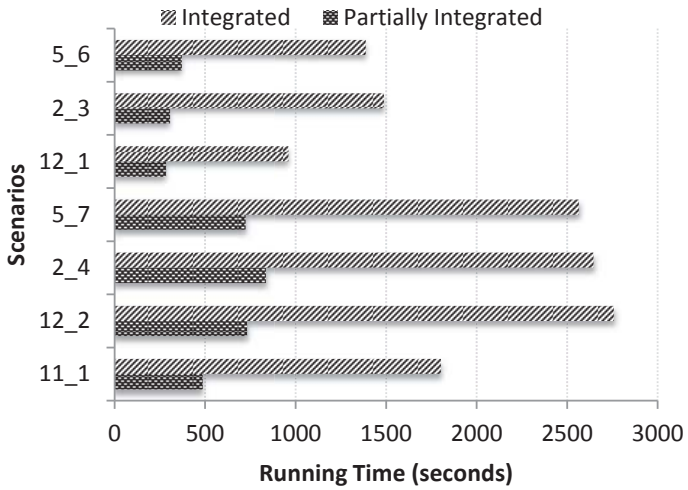

Figure 3.9: Running time comparisons.

Figure 3.8 shows the percentages of deadheads changed. Clearly, most scenarios do not reduce deadhead usage except for scenarios 11_1 and 2_3. This implies that deadheads are not the improvement driver under the default cost setting. Later in Figure 3.10, we show the impact on deadheads when the deadhead cost is doubled.

Figure 3.9 shows the running time of the integrated solution for each scenario. The running times range from 20 minutes up to 50 minutes. The harder the scenario is, more running time is needed, and about three times more running time is required to solve our integrated recovery problem. Note that by increasing the number of computer cores, we estimate that the running time can be reduced by half if the cache can be enlarged to handle the memory demanding roster generation process.

We have increased the running time of the sequential approach by increasing the number of feasible rosters for both the disrupted and affected crew members to be generated up to the point where the objective of the CRM does not change anymore. We observe that the performance of the sequential solution by opening up the number of rosters is improved by less than 1% overall in the 2-hour scenarios, and is not improved for any 1-hour scenario. Hence, the partially integrated approach remains significantly outperformed by our approach in all scenarios.

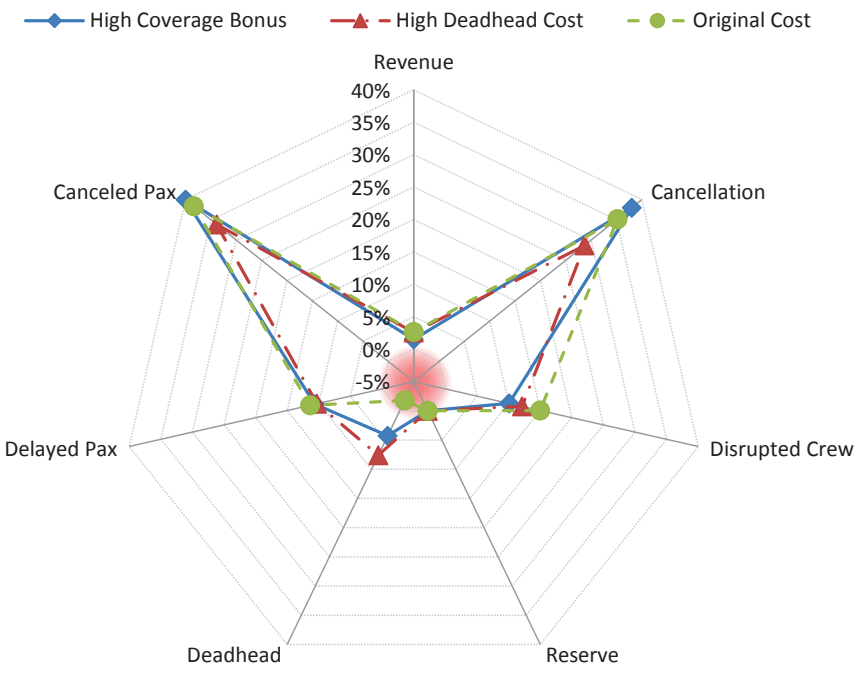

Figure 3.10: Results for different cost settings.

The same computational study is also conducted for two different cost settings: one with doubled cov-

erage bonuses (the original and other flight coverage revenues in Table 3.2), and one with doubled deadhead cost. Figure 3.10 shows the average percentage reduced over all disruption scenarios for each key performance indicator. The central red area indicates the percentage increased comparing to the partially integrated solution. We observe that when the deadhead cost is doubled, deadhead reductions are almost doubled, and when the coverage bonus is doubled, we obtain fewer cancellations and canceled itineraries. High coverage bonus also discourages deadheads as it is more beneficial to cover more flights with the same number of available crew members. Overall, the revenue improvement does not significantly deviate across the three different cost settings.

Moreover, we evaluate the performance of our solution by shortening the recovery time window by imposing the end of the recovery time window at 6 p.m. The revenue improvements are summarized in Figure 3.11. Note that scenarios 5_6 and 5_7 are now essentially the same, and hence scenario 5_7 is not included in the figure. Although the overall revenue improvements are less pronounced, the improvement trend is similar to the one for the 24-hour recovery time window in Figure 3.4.

We have also tested if applying cuts differently will affect our results. We tried applying cuts every other Benders iteration, the last iteration only, and the last 8 iterations only (with a total of 16 Benders iterations). The performances were inferior.

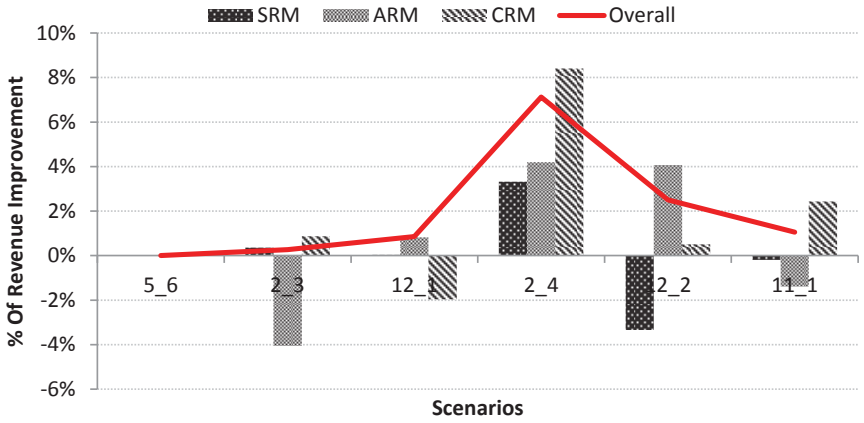

Figure 3.11: Percentage of revenue improvement when the recovery time window is limited to 12 hours.

3.6 CONCLUSION

We have studied a fully integrated recovery problem as well as the Benders decomposition framework that decomposes the problem into resource recovery problems. Many different and innovative modeling and algorithmic strategies are proposed to include business requirements and to reduce the solution time. We evaluated the performance of our proposed solution based on a real world data set provided by a major solution vendor, and observed significant improvements over the partially integrated solution used by the industry.

Our solution performs well on small disruption instances and significantly better on large disruption instances. A fewer number of canceled flight, disrupted crew members, delayed passengers, and canceled itineraries generally lead to a better integrated recovery solution. The harder and longer the disruption scenario is, the more improvement we observed. Furthermore, our solution is robust to different cost settings, and provides an improvement of 2.6% on average and 8% maximum, which accounts for up to one million in saving per recovery. It does not only help airlines to reduce their recovery costs in the long run, but it also allows them to make close-to-real-time recovery decisions whenever disruptions occur.

Although our solutions are significantly better, we have identified two possible research directions that may further improve the fully integrated methodology. One direction is to adapt the splitting strategy of ARM to CRM by having SRM to select alternative crew connections, so that the roster generation process may not be necessary. This may require the SRM to select an alternative crew connection to satisfy for each crew member. The other direction is to investigate if the total recovery cost and running time can be further reduced by switching the order of the Benders subproblems. Some evidences can be found in Klabjan et al. (2002) and Mercier et al. (2005). They are about resource planning, and it is unclear if similar performances can be achieved when the underlying problem concerns operations recovery.

REFERENCES

Abdelghany, K. F., Abdelghany, A. F., and Ekollu, G. (2008). An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, 185(2):825 – 848. [cited at p. 73]

Amaruchkul, K., Copper, W. L., and Gupta, D. (2007). Single-leg air-cargo revenue management. *Transportation Science*, 41(4):457–469. [cited at p. 50]

Amaruchkul, K., Copper, W. L., and Gupta, D. (2010). A note on air-cargo capacity contracts. *Production and Operations Management*. http://dx.doi.org/10.1111/j.1937-5956.2010.01158.x. [cited at p. 49]

Barnhart, C., Boland, N. L., Clarke, L. W., Johnson, E. L., Nemhauser, G. L., and Shenoi, R. G. (1998). Flight string models for aircraft fleeting and routing. *Transportation Science*, 32(3):208–220. [cited at p. 71]

Belobaba, P. P. and Weatherford, L. R. (1996). Comparing decision rules that incorporate customer diversion in perishable asset revenue management situations. *Decision Sciences Journal*, 27(2):343–363. [cited at p. 17]

Bisaillon, S., Cordeau, J.-F., Laporte, G., and Pasin, F. (2011). A large neighbourhood search heuristic for the aircraft and passenger recovery problem. *A Quarterly Journal of Operations Research*, 9(2):139–157. [cited at p. 73]

Bitran, G. and Caldentey, R. (2003). An overview of pricing models for revenue management. *Manufacturing & Service Operations Management*, 5(3):203 – 229. [cited at p. 14]

Bratu, S. and Barnhart, C. (2006). Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, 9(3):279–298. [cited at p. 73]

Brumelle, S. L. and McGill, J. I. (1993). Airline seat allocation with multiple nested fare classes. *Operations Research*, 41(1):127–137. [cited at p. 16, 17, 50]

Chen, L. and Homem-de-Mello, T. (2010). Re-solving stochastic programming models for airline revenue management. *Annals of Operations Research*, 177(1):91–114. [cited at p. 17]

Chew, E.-P., Huang, H.-C., Johnson, E. L., Nemhauser, G. L., Sokol, J. S., and Leong, C.-H. (2006). Short-term booking of air cargo space. *European Journal of Operational Research*, 174(3):1979 – 1990. [cited at p. 50]

Chunhua, G. (2007). *Airline Integrated Planning and Operations*. PhD thesis, Georgia Institute of Technology. [cited at p. 73]

Cooper, W. L. (2002). Asymptotic behavior of an allocation policy for revenue management. *Operations Research*, 50(4):720–727. [cited at p. 15, 29, 30, 31]

Curry, R. E. (1990). Optimal airline seat allocation with fare classes nested by origins and destinations. *Transportation Science*, 24(3):193–204. [cited at p. 15, 16, 20, 24, 29, 43, 50, 111]

de Boer, S. V., Freling, R., and Piersma, N. (2002). Mathematical programming for network revenue management revisited. *European Journal of Operational Research*, 137(1):72 – 92. [cited at p. 14]

DHL (2012). Volumetric weight. [Online; accessed 8-October-2012]. http://www.dhl.com/en/tools/volumetric_weight_express.html. [cited at p. 53]

Eppstein, D. (1999). Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673. [cited at p. 89]

Fiig, T., Isler, K., Hopperstad, C., and Belobaba, P. P. (2010). Optimization of mixed fare structures: Theory and applications. *Journal of Revenue and Pricing Management*, 9(1/2):152–170. [cited at p. 17]

Gallego, G., Li, L., and Ratliff, R. (2009). Choice-based emsr methods for single-leg revenue management with demand dependencies. *Journal of Revenue and Pricing Management*, 8(2/3):207–240. [cited at p. 16, 17, 21, 29, 31, 32, 33, 34, 43, 111]

Hellermann, R. (2004). *Capacity Options for Revenue Management: Theory and Applications in The Air Cargo Industry*. Springer, 1st edition. [cited at p. 49]

Higham, N. J. (2002). Computing the nearest correlation matrix - a problem from finance. *IMA Journal of Numerical Analysis*, 22(3):329–343. [cited at p. 62]

Higle, J. L. (2007). Bid-price control with origin-destination demand: A stochastic programming approach. *Journal of Revenue and Pricing Management*, 5(4):291–304. [cited at p. 17]

Karaesmen, I. (2001). *Three Essays on Revenue Management*. PhD thesis, Columbia University. [cited at p. 49]

Kasilingam, R. G. (1997). Air cargo revenue management: Characteristics and complexities. *European Journal of Operational Research*, 96(1):36 – 44. [cited at p. 49]

Klabjan, D. (2005). Column generation (chapter 16): Large-scale models in the airline industry. *Springer*, pages 168 – 196. [cited at p. 76]

Klabjan, D., Johnson, E. L., Nemhauser, G. L., Gelman, E., and Ramaswamy, S. (2002). Airline crew scheduling with time windows and plane-count constraints. *Transportation Science*, 36(3):337 – 348. [cited at p. 99]

Lettovsky, L. (1997). *Airline Operations Recovery: An Optimization Approach*. PhD thesis, Georgia Institute of Technology. [cited at p. 71, 72, 73, 74]

Levin, Y., Nediak, M., and Topaloglu, H. (2012). Cargo capacity management with allotments and spot market demand. *Operations Research*, 60(2):351–365. [cited at p. 50]

Levina, T., Levin, Y., McGill, J., and Nediak, M. (2011). Network cargo capacity management. *Operations Research*, 59(4):1008–1023. [cited at p. 50]

Littlewood, K., editor (1972). *Forecasting and control of passenger bookings*, volume 12. The 12th AGIFORS Symposium, Nathanya, Israel. [cited at p. 16]

Luo, S., Çakanyıldırım, M., and Kasilingam, R. G. (2009). Two-dimensional cargo overbooking models. *European Journal of Operational Research*, 197(3):862 – 883. [cited at p. 50]

Mansi, R., Hanafi, S., Wilbaut, C., and Clautiaux, F. (2010). Disruptions in the airline industry: math-heuristics for re-assigning aircraft and passengers simultaneously. *Accepted in European Journal of Industrial Engineering*. [cited at p. 74]

Martins, E., Queir, E., Martins, V., Margarida, M., and Pascoal, M. M. B. (2000). A new implementation of Yen's ranking loopless paths algorithm. *4OR*, pages 121–133. [cited at p. 89]

McGill, J. I. and Van Ryzin, G. J. (1999). Revenue management: Research overview and prospects. *Transportation Science*, 33(2):233. [cited at p. 14]

Mercier, A., Cordeau, J.-F., and Soumis, F. (2005). A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32:1451–1476. [cited at p. 99]

Pak, K. and Dekker, R. (2004). Cargo revenue management: Bid-prices for a 0-1 multi knapsack problem. ERIM report series research in management, Rotterdam School of Management, Erasmus Universiteit Rotterdam, The Netherlands. [cited at p. 50]

Palpant, M., Boudia, M., Robelin, C.-A., Gabteni, S., and Laburthe, F. (2009). ROADEF 2009 challenge: Disruption management for commercial aviation. [Online; accessed 3-August-2012]. http://challenge.roadef.org/2009/files/challenge_en.pdf. [cited at p. 73]

Peterson, J. D., Sőlveling, G., Johnson, E. L., Clarke, J.-P., and Shebalov, S. (2012). An optimization approach to airline integrated recovery. *Transportation Science*, Articles in Advance:1–19. [cited at p. 71, 72, 74, 87, 92]

Popescu, A. (2006). *Air Cargo Revenue and Capacity Management*. PhD thesis, Georgia Institute of Technology. [cited at p. 49]

Powell, W., Ruszczynski, A., and Topaloglu, H. (2004). Learning algorithms for separable approximations of discrete stochastic optimization problems. *Mathematics of Operations Research*, 29(4):814–836. [cited at p. 24, 25, 27]

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics. Wiley-Interscience, 1st edition. [cited at p. 20, 26]

RITA (2012a). On-time performance - flight delays at a glance. [Online; accessed 3-August-2012]. http://www.transtats.bts.gov/HomeDrillChart.asp. [cited at p. 70]

RITA (2012b). U.S. air carrier traffic statistics. [Online; accessed 8-October-2012]. http://www.bts.gov/xml/air_traffic/src/datadisp.xml. [cited at p. 44]

Robinson, L. W. (1995). Optimal and approximate control policies for airline booking with sequential nonmonotonic fare classes. *Operations Research*, 43(2):252–263. [cited at p. 19]

Sherali, H. D., Bish, E. K., and Zhu, X. (2006). Airline fleet assignment concepts, models, and algorithms. *European Journal of Operational Research*, 172(1):1 – 30. [cited at p. 76]

Slager, B. and Kapteijns, L. (2004). Implementation of cargo revenue management at KLM. *Journal of Revenue & Pricing Management*, 3(1):80–90. [cited at p. 49]

Talluri, K. and van Ryzin, G. (1998). An analysis of bid-price controls for network revenue management. *Management Science*, 44(11):1577–1593. [cited at p. 14, 19]

Talluri, K. and Van Ryzin, G. (1999). A randomized linear programming method for computing network bid prices. *Transportation Science*, 33(2):207–216. [cited at p. 15, 20, 32]

Talluri, K. and van Ryzin, G. (2004). *The Theory and Practice of Revenue Management*. International Series in Operations Research & Management Science. Springer, 1st edition. [cited at p. 20, 50]

Topaloglu, H. (2009). On the asymptotic optimality of the randomized linear program for network revenue management. *European Journal of Operational Research*, 197(3):884–896. [cited at p. 32]

van Ryzin, G. J. and McGill, J. I. (2000). Revenue management without forecasting of optimization: An adaptive algorithm for determining airline seat protection levels. *Management Science*, 46(6):760–775. [cited at p. 16]

Williamson, E. L. (1992). *Airline Network Seat Inventory Control: Methodologies and Revenue Impacts*. PhD thesis, Massachusetts Institution of Technology, Cambridge, Massachusetts. [cited at p. 14]

Wollmer, R. D. (1992). An airline seat management model for a single leg route when lower fare classes book first. *Operations Research*, 40(1):26–37. [cited at p. 16, 21, 31, 34, 43, 50]

Zhang, D. and Adelman, D. (2009). An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science*, 43(3):381–394. [cited at p. 17]

Appendix A

CHAPTER 1: APPENDIX

A.1 PROOFS

A.1.1 PROPOSITION 1

Proof. For a given itinerary i, suppose the number of remaining seats ξ_i and a matrix of observed accumulated upsells $\boldsymbol{\eta}_i$ are given at time c (note that class and time share the same index by our low-to-high fare arrival order assumption). Let us add a superscript to both ξ_i and $\boldsymbol{\eta}_i$ to represent the number of remaining seats and observed accumulated upsells at time c. We then rely on the following relationships to prove the proposition:

$$\xi_i^c - \Pi_{ic-1} = x_{ic} + z_{ic+1} \text{ for } l \leq c \tag{A.1}$$

and

$$\varphi_{il} = \eta_{il}^c + \sum_{c'>l}^{c} U_{ic'l} \text{ for } l \leq c. \tag{A.2}$$

Equation (A.1) describes the number of seats available for class c. Its LHS is the difference between the remaining number of seats available for all classes and the number of seats protected for all higher classes $c-1,\ldots,1$, and hence, it is the remaining number of seats available for class c. Note that ξ_i^l is a random variable for classes $l = 1,\ldots,c-1$, and it is always no less than Π_{il-1} by the definition of $\mathcal{N}(\mathbf{x}_i)$, i.e. $\xi_i^c = \sum_{l=1}^{c} x_{il} = \Pi_{ic} \geq \Pi_{ic-1}$, and by the way that the number of remaining seats is determined for $V_{il-1}(\cdot,\cdot,\cdot)$, i.e. $\xi_i^l - \min\{\xi_i^l - \Pi_{il-1}, D_{il} + \eta_{il}^l\} \geq \xi_i^l - \xi_i^l + \Pi_{il-1} \geq \Pi_{il-1}$. On the other hand, the RHS is the number of seats reserved for class c plus the total accumulated empty seats from class $c+1$, and thus, is also the remaining number of seats available for class c. Now, z_{ic+1} is a random variable. Using the fact that $\Pi_{ic} = \sum_{c'=1}^{c} x_{ic'}$, equation (A.1) also implies

$$\xi_i^{c-1} = \sum_{c'\leq c-1} x_{ic'} + z_{ic}. \tag{A.3}$$

Equation (A.2) describes how upsells are accumulated to other higher classes given the observed upsells $\boldsymbol{\eta}_i$. Its LHS φ_{il} is the number of total upsells to class l on itinerary i by our definition. On the RHS, η^c_{il} is the given number of total upsells to class l in the beginning of time period c, and $\sum_{c'>l}^{c} U_{ic'l}$ is the accumulated upsells from classes $c, \ldots, l+1$. Hence, adding them together yields the total number of upsells to class l. Note that (A.2) implies both

$$\eta^c_{ic} = \varphi_{ic} \tag{A.4}$$

and

$$\boldsymbol{\eta}^c_i + \mathbf{U}_{ic} = \boldsymbol{\eta}^{c-1}_i, \tag{A.5}$$

and if initial upsells are not given, equation (A.2) is essentially the same as constraint (1.8) for class l.

We prove the proposition by induction. For the base case when $c = 1$, we have $V_{i1}(\emptyset, \xi^1_i, \boldsymbol{\eta}^1_i) = \mathbb{E}[r_{i1} \min\{\xi^1_i, D_{i1} + \eta^1_{i1}\}] = \mathbb{E}[r_{i1} \min\{x_{i1} + z_{i2}, D_{i1} + \psi_{i1}\}]$. Suppose $V_{ic-1}(\boldsymbol{\Pi}^{c-2}_i, \xi^{c-1}_i, \boldsymbol{\eta}^{c-1}_i) = \mathbb{E}[\sum_{c'=1}^{c-1} r_{ic'} \min\{x_{ic'} + z_{ic'+1}, D_{ic'} + \psi_{ic'}\}]$. We have

$$\begin{aligned}
V_{ic}(\boldsymbol{\Pi}^{c-1}_i, \xi^c_i, \boldsymbol{\eta}^c_i) =& \mathbb{E}[r_{ic} \min\{\xi^c_i - \Pi_{ic-1}, D_{ic} + \eta^c_{ic}\} \\
&+ V_{ic-1}(\boldsymbol{\Pi}^{c-2}_i, \xi^c_i - \min\{\xi^c_i - \Pi_{ic-1}, D_{ic} + \eta^c_{ic}\}, \\
&\qquad \boldsymbol{\eta}^c_i + \mathbf{q}_{ic}(D_{ic} - (\xi^c_i - \Pi_{ic-1} - \eta^c_{ic})^+), \mathbf{p}_i(c), c)] \\
=& \mathbb{E}\Bigg[r_{ic} \min\left\{x_{ic} + z_{ic+1}, D_{ic} + \psi_{ic}\right\} \\
&+ V_{ic-1}\Bigg(\boldsymbol{\Pi}^{c-2}_i, \sum_{c' \le c-1} x_{ic'} + x_{ic} + z_{ic+1} - \min\left\{x_{ic} + z_{ic+1}, D_{ic} + \psi_{ic}\right\}, \\
&\qquad \boldsymbol{\eta}^c_i + \mathbf{q}_{ic}\left(D_{ic} - \left(x_{ic} + z_{ic+1} - \psi_{ic}\right)^+, \mathbf{p}_i(c), c\right)\Bigg)\Bigg] \\
=& \mathbb{E}\Bigg[r_{ic} \min\left\{x_{ic} + z_{ic+1}, D_{ic} + \psi_{ic}\right\} \\
&+ V_{ic-1}\Bigg(\boldsymbol{\Pi}^{c-2}_i, \sum_{c' \le c-1} x_{ic'} + (x_{ic} + z_{ic+1} - D_{ic} - \psi_{ic})^+, \\
&\qquad \boldsymbol{\eta}^c_i + \mathbf{q}_{ic}\left(D_{ic} - \left(x_{ic} + z_{ic+1} - \psi_{ic}\right)^+, \mathbf{p}_i(c), c\right)\Bigg)\Bigg] \\
=& \mathbb{E}\left[r_{ic} \min\left\{x_{ic} + z_{ic+1}, D_{ic} + \psi_{ic}\right\} + V_{ic-1}\left(\boldsymbol{\Pi}^{c-2}_i, \xi^{c-1}_i, \boldsymbol{\eta}^c_i + \mathbf{U}_{ic}\right)\right] \\
=& \mathbb{E}\left[\sum_{c'=1}^{c} r_{ic'} \min\left\{x_{ic'} + z_{ic'+1}, D_{ic'} + \psi_{ic'}\right\}\right].
\end{aligned}$$

The second equality is by (A.1) and (A.4). The fourth equality is by definitions (1.6) and (1.7), and equations (A.2) and (A.3). The last equality is by (A.5) and our induction assumption on $c-1$. Recall that we do not

have to consider the case when $\xi_i^c < \Pi_{ic-1}$ as $\xi_i^c \geq \Pi_{ic-1}$ by mapping $\mathcal{N}(\mathbf{x}_i)$ □

A.1.2 LEMMA 1

Proof. We define $\bar{D}_j$ to be the expected demand for product j, $(\mathbf{x}^P, \mathbf{z}^P)$ to be an optimal solution to $\overline{P}(\mathbf{K})$, and $\mathbf{x}^{SP}$ to be an optimal solution to $\overline{SP}(\mathbf{K})$. We prove the lemma by showing that both $\overline{SP}(\mathbf{K}) \geq \overline{P}(\mathbf{K})$ and $\overline{P}(\mathbf{K}) \geq \overline{SP}(\mathbf{K})$ hold.

1. Suppose we are given $(\mathbf{x}^P, \mathbf{z}^P)$ an optimal solution to $\overline{P}(\mathbf{K})$.
   a) If $z_{ic}^P = 0$ for all $c \in C_i$ and $i \in I$, it is easy to see that the solution is also feasible to $\overline{SP}(\mathbf{K})$.
   b) If $z_{ic}^P > 0$ for some $c \in C_i$ and $i \in I$, suppose that $\bar{c}$ is the lowest class such that $z_{i\bar{c}}^P > 0$ for itinerary i, it must be then the case that the number of allocated seats for class $\bar{c}$ is more than necessary, i.e. $x_{i\bar{c}}^P > \bar{D}_{i\bar{c}}$. We can then remove seats from $x_{i\bar{c}}^P$ until either $z_{i\bar{c}}^P = 0$, or there exists a higher class $\tilde{c} < \bar{c}$ such that one accepted booking for class $\tilde{c}$ has to be rejected due to too few empty seats from lower classes. In the latter case, the removed seats from class $\bar{c}$ is added to class $\tilde{c}$ to maintain the same number of the accepted bookings. By repeating this procedure for the remaining classes in the low-to-high fare arrival order for each itinerary, we can obtain a new solution $(\bar{\mathbf{x}}^P, \bar{\mathbf{z}}^P)$ that yields the same objective value and has $\bar{z}_{ic}^P = 0$ for all $c \in C_i$ and $i \in I$. Since neither the total allocation to each itinerary has been increased nor the number of accepted bookings has been reduced, such a solution is feasible to $\overline{SP}(\mathbf{K})$.

   In both cases, the solutions are feasible to $\overline{SP}(\mathbf{K})$, and we have $\overline{SP}(\mathbf{K}) \geq \overline{P}(\mathbf{K})$.
2. Suppose we are now given $\mathbf{x}^{SP}$ an optimal solution to $\overline{SP}(\mathbf{K})$.
   a) For itinerary i, if all $\bar{D}_{ic} \geq x_{ic}^{SP}$, then the corresponding $\mathbf{z}_i$ computed using (1.9) is a vector with zeros. Thus, the solution $(\mathbf{x}^{SP}, \mathbf{z} = \mathbf{0})$ is feasible to $\overline{P}(\mathbf{K})$.
   b) For itinerary i, suppose $\mathcal{C}_i$ is a set of classes with $\bar{D}_{i\tilde{c}} < x_{i\tilde{c}}^{SP}$ for each $\tilde{c} \in \mathcal{C}_i$.
      i. If we have $\bar{D}_{ic} = x_{ic}^{SP}$ for all $c \notin \mathcal{C}_i$, we can then reduce $x_{i\tilde{c}}^{SP}$ until it is equal to $D_{i\tilde{c}}$ for all $\tilde{c} \in \mathcal{C}_i$. The resulting solution does not change the original objective value and has $\mathbf{z}_i = \mathbf{0}$. Hence, it is also feasible to $\overline{P}(\mathbf{K})$.
      ii. If we have $\bar{D}_{i\bar{c}} > x_{i\bar{c}}^{SP}$ for any $\bar{c} \notin \mathcal{C}_i$, then solution $\mathbf{x}_i^{SP}$ is not optimal, since we can extract a seat from any one of the classes in $\mathcal{C}_i$ and add the extracted seat to class $\bar{c}$ to obtain a higher objective value. It contradicts our optimality assumption on $\mathbf{x}^{SP}$.

In all cases, we show that either $\mathbf{x}^{SP}$ can be converted to a feasible solution to $\overline{P}(\mathbf{K})$ or $\mathbf{x}^{SP}$ is not optimal to induce a contradiction, and hence, we have $\overline{SP}(\mathbf{K}) \leq \overline{P}(\mathbf{K})$ as required.

□

A.1.3 LEMMA 2

Proof. We refer to HL as high-to-low fare arrival order, LH as low-to-high fare arrival order, and R as random arrival order. Let $\mathbf{D}^o$ be a demand sample with arrival order o. We have the following observations

Observation 1. *We have* $\mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^{HL}) \geq \mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^{R}) \geq \mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^{LH})$.

Proof. This is based on the fact that seats occupied by low-yield passengers in the LH arrival order can be given to high-yield passengers in both the R and HL arrival orders. When the arrival order is HL, all seats will first be filled with high-yield passengers before low-yield passengers are considered. When the arrival order is R, some of the seats allocated to low-yield classes will be sold to high-yield passengers first. When the arrival order is LH, the seats will be sold to low-yield passengers first. Hence, we can obtain the stated bounding properties. □

Observation 2. *We have* $\mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^{HL}) = \mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^{R}) = \mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^{LH})$.

Proof. This is based on the fact that the partitioned allocation of $SP(\mathbf{K})$ is arrival order independent. Each class of passengers has its own allocation. Changing merely the arrival order without changing the magnitude of the demand does not change the number of passengers that we accept using any partitioned allocation of $SP(\mathbf{K})$. □

Now, we are ready to prove the lemma. Suppose $\mathbf{x}^{\pi}$ is an optimal solution to problem π, and $\mathbf{z}^{\pi}$ are the extracted empty seats based on the optimal solution to problem π. Given both an optimal solution $(\mathbf{x}^{P}, \mathbf{z}^{P})$ to problem $P(\mathbf{K})$ and an optimal solution $\mathbf{x}^{SP}$ to problem $SP(\mathbf{K})$ with $\mathbf{z}^{SP}$ computed using (1.9), we have

$$\begin{aligned}
\mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^{HL}) \geq \mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^{R}) \geq \mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^{LH}) &= \sum_{i\in I}\sum_{c\in C_i} r_{ic}\min\{x^{P}_{ic} + z^{P}_{ic+1}, D^{LH}_{ic}\} \\
&\geq \sum_{i\in I}\sum_{c\in C_i} r_{ic}\min\{x^{SP}_{ic} + z^{SP}_{ic+1}, D^{LH}_{ic}\} \\
&\geq \sum_{i\in I}\sum_{c\in C_i} r_{ic}\min\{x^{SP}_{ic}, D^{LH}_{ic}\} \\
&= \mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^{LH}) = \mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^{R}) = \mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^{HL}).
\end{aligned}$$

The first several inequalities are due to our first observation. On the R.H.S, the first equality is due to the definition of $P(\mathbf{K})$, and the first inequality is due to the optimality of $(\mathbf{x}^P, \mathbf{z}^P)$. The second inequality is due to the fact that $z^{SP}_{ic+1} \geq 0$ for all $c \in C_i$ and $i \in I$, and the last several equalities are due to the definition of $SP(\mathbf{K})$ and the last observation. Hence, we have $\mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^R) \geq \mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^R)$, which implies $\mathbb{E}\mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^R) \geq \mathbb{E}\mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^R)$.

Lastly, the inequality of $\mathbb{E}\mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^R) \geq \mathbb{E}\mathcal{R}^{DLP(\mathbf{K})}(\mathbf{D}^R)$ is clear, since we have

$$\begin{aligned}
\mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^R) &= \mathcal{R}^{SP(\mathbf{K})}(\mathbf{D}^{HL}) \\
&= \sum_{i\in I}\sum_{c\in C_i} r_{ic}\min\{x^{SP}_{ic}, D^{HL}_{ic}\} \\
&\geq \sum_{i\in I}\sum_{c\in C_i} r_{ic}\min\{\lfloor x^{DLP}_{ic}\rfloor, D^{HL}_{ic}\} \\
&= \mathcal{R}^{DLP(\mathbf{K})}(\mathbf{D}^{HL}) = \mathcal{R}^{DLP(\mathbf{K})}(\mathbf{D}^R),
\end{aligned}$$

where $\lfloor x \rfloor$ is the flooring operation that takes the largest integer not exceeding x. The inequality is due to the optimality of $\mathbf{x}^{SP}$ over all partitioned allocation policies. □

A.2 ALGORITHMS

A.2.1 UPSELL REVENUE ESTIMATION ALGORITHM

The upsell revenue estimation algorithm estimates the upsell revenue given a set of demand samples and class-level partitioned allocation. It heavily relies on the recursive structure of (1.6) and (1.7), and takes the set of class-level partitioned allocation levels $\{x_c\}$, the set of demand samples $\{\zeta^n_c\}$, the set of upsell probabilities $\{p_{cc}\}$, and returns the average revenue over all demand samples, and the corresponding number of rejected bookings and upsells.

Let α^n_c be a binary variable that indicates if a class-c booking is rejected, let β^n_c be a binary variable that indicates if an upsell to class c is rejected due to insufficient allocated seats, and let $\gamma^n_{cc'}$ be a binary variable that indicates if a rejected class-c booking successfully upsells to class c'. These three indicators are used to store information for the upsell margin estimation algorithm (Algorithm 6, presented later) to efficiently and marginally estimate the marginal revenue if an additional seat is given to any one of the classes. The algorithm is summarized in Algorithm 5.

Algorithm 5 Upsell revenue estimation algorithm

Require: x_c for $c \in C$, $p_{cc'}$ for $c, c' \in C$, and ζ_c^n for $c \in C$ and $n = 1, \dots N$.

1: Set $r = 0$, $\alpha_c^n = 0$, $\beta_c^n = 0$, and $\gamma_{cc'}^n = 0$ for $c, c' \in C$ and $n = 1, \dots, N$.
2: **for** each demand sample **do**
3: Initialize $z = 0$, $r' = 0$, and $u_{cc'} = 0$ for $c, c' \in C$.
4: **for** $c = |C|, \dots, 1$ **do**
5: $\varphi = \sum_{c'=c+1}^{|C|} u_{c'c}$.
6: $r' = r' + r_c \min\{x_c + z, \zeta_c^n + \varphi\}$.
7: **if** $\zeta_c^n > \{x_c + z - \varphi\}^+$ **then**
8: $\alpha_c^n = 1$.
9: **if** c is not the highest class **then**
10: Generate $\{u_{cc'}\}_{c'=1,\dots,|C|}$ based on $\mathcal{B}(\zeta_c^n - (x_c + z - \varphi)^+, \mathbf{p}(c))$.
11: **for** $c' = 1, \dots, |C|$ **do**
12: **if** $u_{cc'} > 0$ **then**
13: $\gamma_{cc'} = 1$
14: **end if**
15: **end for**.
16: **end if**
17: **end if**
18: **if** $\varphi > x_c + z$ **then**
19: $\beta_c^n = 1$.
20: **end if**
21: $z = (x_c + z - \varphi - \zeta_c^n)^+$.
22: **end for**
23: $r = r + r'/N$.
24: **end for**
25: **return** r, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$.

For each demand sample, the algorithm starts from the lowest class and computes the total upsell to other classes. Revenue is collected according to (1.5) in step 6. If there exists a rejected booking in class c, α_c^k is set to one. If c is not the highest class, and c' is the class that the rejected type-c booking is upselling to, $\gamma_{cc'}$ is set to one. The algorithm then checks if upsells to class c occupy all seats allocated to class c. If it is the case, β_c^k is set to one to record the fact that there exists at least one rejected upsell to class c. The three variables $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ record information required to compute the marginal revenue in margin estimation algorithm. By recording these rejection and upsell information, we can, instead of computing the finite differences based on the estimated revenue for each class when one more seat is added, reduce the number of algorithmic operations by first computing the base revenue (before a seat is added) and marginally estimating the marginal revenues for all classes. This reduces the running time significantly when the number of classes is high. At the end, the algorithm updates the number of empty seats available for higher classes using (1.6) in step 21.

A.2.2 MARGIN ESTIMATION ALGORITHM

The margin estimation algorithm computes the revenue margin if one more seat is given to a particular class in question. All necessary information to compute the margin is encoded in variables $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ (see Appendix A.2.1 for definitions). It requires the same inputs as the upsell revenue estimation algorithm (Algorithm 5) without the set of demand samples. In addition, the class in question $\hat{c}$ is also needed to indicate to which class the seat should be added in order to compute the corresponding marginal revenue. The algorithm is summarized in Algorithm 6.

Algorithm 6 Margin estimation algorithm

Require: $\hat{c}$, x_c for $c \in C$, $\{\alpha_c^n\}$, $\{\beta_c^n\}$, and $\{\gamma_{cc'}^n\}$ for $c, c' \in C$ and $n = 1, \ldots, N$.

1: Initialize $m = 0$.
2: **for** $n = 1, \ldots, N$ **do**
3: Initialize $m' = 0$.
4: **if** $\beta_{\hat{c}}^n = 1$ **then**
5: $m' = r_{\hat{c}}$.
6: **else**
7: **for** $c' = c, \ldots, 1$ **do**
8: **if** $\alpha_{c'}^n = 1$ **then**
9: $\tilde{c} = \max_{c=c'-1,\ldots,1}\{\gamma_{c'c}^n \geq 1\}$
10: **if** $\tilde{c}$ exists **then**
11: $m' = r_{c'} - r_{\tilde{c}}$.
12: **else**
13: $m' = r_{c'}$.
14: **end if**
15: **go to** step 19.
16: **end if**
17: **end for**
18: **end if**
19: Set $m \leftarrow m + m'/N$.
20: **end for**
21: **return** m

The margin estimation algorithm starts with checking if there exists a rejected upsell from any lower classes to class $\hat{c}$ due to insufficient allocated seats. If a rejected upsell exists and one more seat was given, the rejected upsell should have been captured instead of being rejected. The algorithm then returns the fare of class $\hat{c}$ in step 5. If such a rejected upsell does not exist, then for any higher classes $c' = 1, \ldots, \hat{c} - 1$, the algorithm checks both if there exists a rejected booking and if such a rejected booking results in an upsell. If both conditions are satisfied, the algorithm adjusts the margin according to step 11. This reflects the fact that if one more seat was allocated to class $\hat{c}$, the upsell should have not been occurred due to the nesting nature

of the allocation policy. Therefore, the resulting marginal revenue should be non-positive. Otherwise, if an upsell cannot be found, the margin is set to be the fare of class c' in step 13.

A.2.3 EMSR-UPSELL ALGORITHM

The EMSR algorithm takes the number of total allocated seats y, a set of upsell probabilities $\mathbf{p}$, the probability that a rejected class-c booking ever upsells $\theta_c = \sum_{c'=1}^{c-1} p_{cc'}$, and the average fare over all classes above or equal to class c $q_c = \sum_{c'=1}^{c} r_c \mathbb{E}D_c / \sum_{c'=1}^{c} \mathbb{E}D_c$. It returns a partitioned allocation and an approximated revenue margin for each itinerary allocation level. The algorithm is summarized in Algorithm 7.

Algorithm 7 EMSR-upsell algorithm

Require: y, $p_{cc'}$ for $c, c' \in C_i$, and θ_c and q_c for $c \in C$.
1: Initialize $\Pi = 0$ and $S'_s = 0$ for $s = 0, \ldots, y$.
2: **for** $c = 1, \ldots, |C|$ **do**
3: **for** $s = \Pi, \ldots, y$ **do**
4: $S'_s = r_c(1 - F_c(s - \Pi)) + \sum_{s'=0}^{y-\Pi} f_c(s') S_{y-s'}$, where F_c is the c.d.f. of class-c demand, and f_c is the p.d.f. of class-c demand.
5: **end for**
6: **if** c is not the lowest class **then**
7: $\Pi' = \arg\min_{s=\Pi,\ldots,y}\{(r_{c+1} - \theta_{c+1} q_c) \geq S'_s(1 - \theta_{c+1})\}$.
8: $S_s = S'_s$ for $s = \Pi, \ldots, y$.
9: $x_c = \max\{\Pi' - \Pi, 0\}$.
10: $\Pi = \Pi'$.
11: **else**
12: $S_s = S'_s$ for $s = 0, \ldots, y$.
13: **end if**
14: **end for**
15: **return** $\mathbf{x} = \mathcal{P}(\mathbf{\Pi}, y)$, $\mathbf{S}$.

The algorithm computes the marginal revenue based on the optimality conditions in Curry (1990). Once the marginal revenues are computed, the optimal protection level is determined for the class in question based on the adapted fare-adjusted criterion in Gallego et al. (2009) in step 8. The slope vector is then updated, and the corresponding partitioned allocation is computed.

A.3 TABLES

This section includes all results used to create the figures in the main document. Table A.1 shows the running time of the upsell heuristic (Algorithm 2) for each number of classes and demand (λ) we tested. In general, the average running time increases when the number of classes and demand increase.

Table A.1: Average running time of the upsell heuristic and the associated demand factor

	Average Running Time (s)					Average Demand Factor				
$\lvert C\rvert \backslash \lambda$	10	20	30	40	50	10	20	30	40	50
2	21.61	31.93	36.36	37.52	36.95	0.48	0.97	1.45	1.94	2.42
3	8.95	18.28	30.56	42.07	49.94	0.50	1.01	1.51	2.01	2.52
4	18.97	35.26	52.70	68.25	82.66	0.52	1.04	1.55	2.07	2.59
5	32.33	57.19	82.19	106.25	124.61	0.53	1.05	1.58	2.11	2.64
6	51.95	87.38	119.88	150.91	173.12	0.54	1.07	1.61	2.14	2.67
7	69.53	117.82	163.87	203.67	236.21	0.54	1.08	1.62	2.17	2.71
8	99.63	159.36	211.73	259.14	295.22	0.55	1.09	1.64	2.18	2.72
9	112.35	193.95	269.92	332.24	382.36	0.55	1.10	1.65	2.20	2.75
10	142.37	235.57	317.42	390.32	450.86	0.55	1.10	1.66	2.21	2.76

Table A.2 shows the minimum, average, and maximum demand factors over all flights for each demand multiplier. When the demand multiplier is 0, the average demand factor is 80% representing that the network is 80% full. When the demand factor is above 100%, the number of bookings is more than the number of seats available.

Table A.2: Minimum, average, and maximum demand factors over all flights for different demand multipliers

Demand Multiplier	min	avg	max
-0.4	0.02	0.48	1.18
-0.2	0.03	0.65	2.13
0	0.03	0.8	2.4
0.2	0.04	0.96	2.81
0.4	0.06	1.1	2.97

Table A.3 shows the average percentage of revenue improvement by using the proposed nested allocation policy instead of the RLP bid-prices for each demand multiplier and upsell probability multiplier. The average is taken over all generated demand sample paths, one sample path for each simulation experiment.

Table A.3: Average percentage of revenue improvement by using the nested allocation policy.

Upsell Probability Multiplier\ Demand Multiplier	-0.4	-0.2	0	0.2	0.4
0	-0.33%	-0.26%	-0.11%	0.28%	0.63%
0.1	-0.33%	-0.19%	0.12%	0.60%	1.13%
0.2	-0.39%	0.08%	0.56%	1.34%	2.13%
0.3	0.30%	0.35%	0.90%	2.12%	3.25%
0.4	1.62%	1.69%	2.29%	3.53%	5.09%
0.5	3.64%	4.38%	4.64%	5.92%	7.96%
0.6	6.76%	8.50%	8.31%	9.90%	11.81%
0.7	13.64%	14.92%	15.36%	16.41%	18.06%
0.8	21.67%	23.65%	24.32%	24.70%	25.77%
0.9	30.77%	33.21%	32.75%	32.72%	34.06%

Appendix B

CHAPTER 2: APPENDIX

B.1 ROLLING HORIZON IMPLEMENTATION OF THE RISK NEUTRAL PROBLEM

We describe the modifications made to the risk neutral problem so that it can be solved in a rolling horizon manner. The modifications are based on adding a time dimension to the decision variables and coefficients. To describe the problem, in addition to the notations defined for CAP, we add the following:

- $D(\underline{t}, \bar{t})$ set of demand units that can be shipped between time $\underline{t}$ and $\bar{t}$,
- $T(D(\underline{t}, \bar{t}), f)$ set of time periods in which flight f can be used to carry demand in $D(\underline{t}, \bar{t})$,
- w_{ft} total consumed weight on flight f by time t,
- v_{ft} total consumed volume on flight f by time t,
- n_{fpt} total number of type-p positions consumed on flight f by time t,
- y_{tfpu} (integer) number of type-u ULDs assigned to type-p position on flight f and date t. Note that $\sum_{t \in T} y_{tfpu} = y_{fpu}$, which is the total number of type-p positions required to hold all assigned type-u ULDs.

The rolling horizon version of RNP is

$$\max \sum_{d \in D(\underline{t},\bar{t})} \sum_{i \in I^D(d)} \sum_{u \in U^D(d)} r_{di} \rho^w_{du} x_{diu}$$

$$\sum_{d \in D(\underline{t},\bar{t})} \sum_{i \in I^{DF}(d,f)} \sum_{u \in U^D(d)} \rho^w_{du} x_{diu} +$$

$$\sum_{p \in P(f)} \sum_{u \in U^P(p)} \sum_{t \in T(D(\underline{t},\bar{t}),f)} \tau^w_u y_{tfpu} \leq w_f - w_{ft-1} \qquad f \in F \tag{B.1}$$

$$\sum_{d\in D(\underline{t},\bar{t})}\sum_{i\in I^{DF}(d,f)}\sum_{u\in U^{D}(d)}\rho^{v}_{du}x_{diu}+$$

$$\sum_{p\in P(f)}\sum_{u\in U^{P}(p)}\sum_{t\in T(D(\underline{t},\bar{t}),f)}\tau^{v}_{u}y_{tfpu}\leq v_{f}-v_{ft-1}\qquad f\in F \tag{B.2}$$

$$\sum_{d\in D(\underline{t},\bar{t})}\sum_{i\in I^{DF}(d,f)}x_{diu}\leq\sum_{p\in P(f)}y_{tfpu}\qquad t\in T(D(\underline{t},\bar{t}),f),u\in U^{F}(f),f\in F \tag{B.3}$$

$$\sum_{t\in T(D(\underline{t},\bar{t}),f)}\sum_{u\in U^{P}(p)}\rho^{p}_{u}y_{tfpu}\leq n_{fp}-n_{fpt-1}\qquad p\in P(f),f\in F \tag{B.4}$$

Constraints (B.1), (B.2), and (B.4) are similar to constraints (2.3), (2.4), and (2.6), which are the upper bounds on the remaining weight, volume, and number of positions respectively. Constraints (B.3) are the same as constraints (2.5) with a time dimension added.

VITA

Chan Seng (Bill) Pun, the middle child of Pun Wai and Chan Hang Chon, was born in Macau SAR, where he completed his high school diploma before he continued his undergraduate education in the United States. In June 2007, he received his bachelor degree in the department of Industrial and Systems Engineering from the University of Washington, Seattle, and moved to Chicago to pursue a doctorate in Industrial Engineering and Management Sciences at Northwester University majoring in optimization. His expertises include revenue management, approximate dynamic programming, large scale optimization, parallel computing, and business intelligence. In November 2012, he moved to Dallas, Texas to work for Sabre Holdings as a senior operations research consultant. More information is available at http://www.billpun.com.