Itinerary-Based Control with Nesting and Upsell

Chan Seng Pun*[1], Diego Klabjan[1], Fikri Karaesmen[2], and Sergey Shebalov[3]

[1]Industrial Engineering and Management Sciences, Northwestern University
[2]Department of Industrial Engineering, Koç University
[3]Airline Solutions, Sabre Holdings

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 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

*billpun@billpun.com

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 2 provides a general overview on several well-known seat allocation models. Section 3 presents our itinerary-based nesting model with upsell, and Section 4 elaborates the approximate dynamic programming algorithm that we apply to solve our problem. Several other heuristics and algorithms are also presented. Section 5 discusses asymptotic optimality of the nested allocation policy when no upsell is considered. Section 6 reports simulation results, and Section 7 concludes the paper.

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.

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

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

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

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

where $R_{i0} = 0$, $\Pi_{i0} = 0$, and $f_j(\cdot)$ is the demand density function for product $j = (i, c)$. The revenue function (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 (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.

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

$$V_{ic}(\mathbf{\Pi}_i^{c-1}, \xi, \boldsymbol{\eta}_i) = \begin{cases} \mathbb{E}\left[r_{ic} \min\{\xi - \Pi_{ic-1}, D_{ic} + \eta_{ic}\}\right. \\ \quad + V_{ic-1}\left(\mathbf{\Pi}_i^{c-2}, \xi - \min\{\xi - \Pi_{ic-1}, D_{ic} + \eta_{ic}\},\right. \\ \qquad \left.\left. \boldsymbol{\eta}_i + \mathbf{q}_{ic}(D_{ic} - (\xi - \Pi_{ic-1} - \eta_{ic})^+, \mathbf{p}_i(c), c)\right)\right] & \text{if } \xi \geq \Pi_{ic-1}, \\ \mathbb{E}[V_{ic-1}(\mathbf{\Pi}_i^{c-2}, \xi, \boldsymbol{\eta}_i + \mathbf{q}_{ic}(D_{ic}, \mathbf{p}_i(c), c))] & \text{otherwise,} \end{cases} \tag{3}$$

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:*

$$U^*(\mathbf{K}) = \max_{\mathbf{x} \in \mathcal{J}(\mathbf{K})} \sum_{i \in I} \mathbb{E}\left[Q_i(\mathbf{x}_i, \mathbf{D}_i)\right]. \tag{4}$$

The revenue function for each itinerary i is

$$Q_i(\mathbf{x}_i, \mathbf{d}_i) = \max_{\mathbf{z} \text{ integer}} \mathbb{E}\left[\sum_{c \in C_i} r_{ic} \min\left\{x_{ic} + z_{ic+1}, d_{ic} + \psi_{ic}\right\}\right] \tag{5}$$

subject to

$$z_{ic} = (x_{ic} + z_{ic+1} - d_{ic} - \psi_{ic})^+ \qquad c \in C_i \tag{6}$$

$$(U_{ic1}, \ldots, U_{ic|C_i|}) = \mathbf{q}_{ic}\left(d_{ic} - (x_{ic} + z_{ic+1} - \psi_{ic})^+, \mathbf{p}_i(c), c\right) \qquad c \in C_i \tag{7}$$

$$\sum_{c'=c}^{|C_i|-1} U_{ic'c} = \psi_{ic} \qquad c \in C_i \tag{8}$$

where expectation in (4) is taken over demand, and expectation in (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 (6) and (7). See Appendix A.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 (6) and (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

$$\begin{aligned} \mathcal{S}_i(\mathbf{x}_i, \mathbf{d}_i) = \max_{\mathbf{z} \text{ integer}} & \sum_{c \in C_i} r_{ic} \min\{x_{ic} + z_{ic+1}, d_{ic}\} \\ z_{ic} = (z_{ic+1} + x_{ic} - d_{ic})^+ & \qquad c \in C_i. \end{aligned} \tag{9}$$

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 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.

Table 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.

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 (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 (6) and (7). In our application, we use piecewise linear functions as the basis functions. To have the objective function (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 $l \le y_i < l+1$. 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: $\mathbf{v}_i = \arg\min_{\boldsymbol{\delta}}\{\sum_{s=1}^{S_i}(\delta_{is} - \bar{v}_{is})^2 | \delta_{is+1} \le \delta_{is} \text{ for } s = 1, \ldots, S_i\}$.
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: $\quad x_{i\hat{c}} = x_{i\hat{c}} - 1$.
6: $\quad$ Apply the revenue estimation algorithm to compute revenue r and collect upsell information.
7: $\quad$ 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: $\quad x_{ic'} = x_{ic'} + 1$.
9: $\quad$ **if** $\hat{c} = c'$ or $x_{i\hat{c}} = 0$ **then**
10: $\quad\quad$ Find $\tilde{c}$ the next highest class with a positive allocation.
11: $\quad\quad$ **if** class $\tilde{c}$ exists **then**
12: $\quad\quad\quad \hat{c} = \tilde{c}$
13: $\quad\quad$ **else**
14: $\quad\quad\quad$ **return** $\mathbf{x}_i, r + \delta_{c'}, \delta_{c'}$.
15: $\quad\quad$ **end if**
16: $\quad$ **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 3 in Step 6), the margin estimation algorithm (Algorithm 4 in Step 7), and the EMSR-upsell algorithm (Algorithm 5 in 2) with their details documented in Appendices B.1, B.2, and B.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 (6) and (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.

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.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.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

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

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.

6 Computational Experiments

In Section 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 (3). The algorithm in Wollmer (1992) is developed directly based on the optimality condition of the revenue function (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 (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 (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 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.

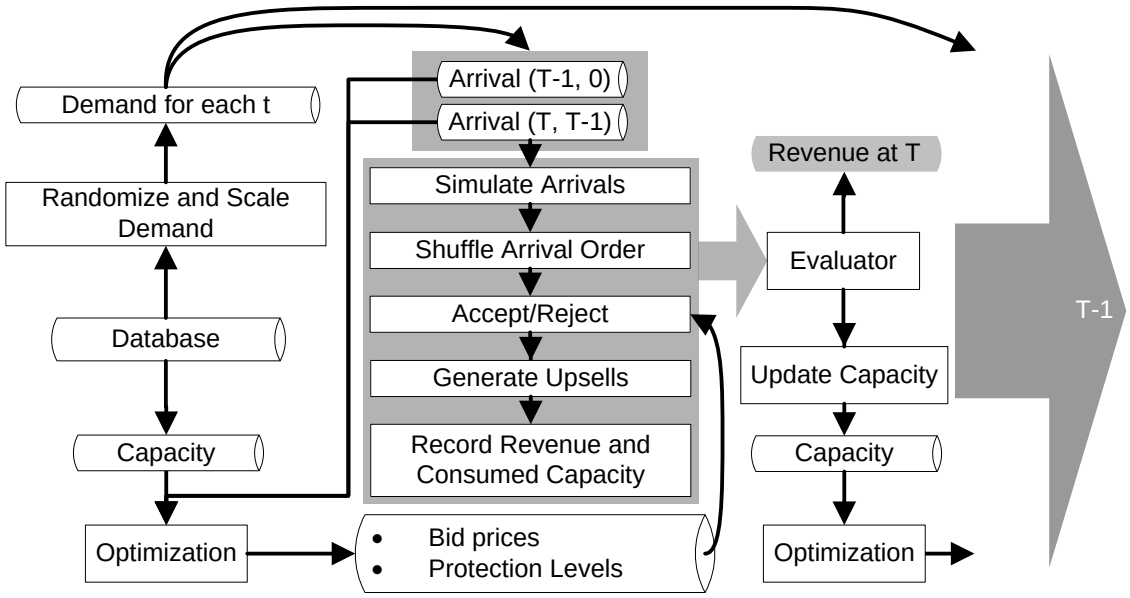

Figure 1: Flow chart of simulation at time T, the first reading day.

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.

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 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 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 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.

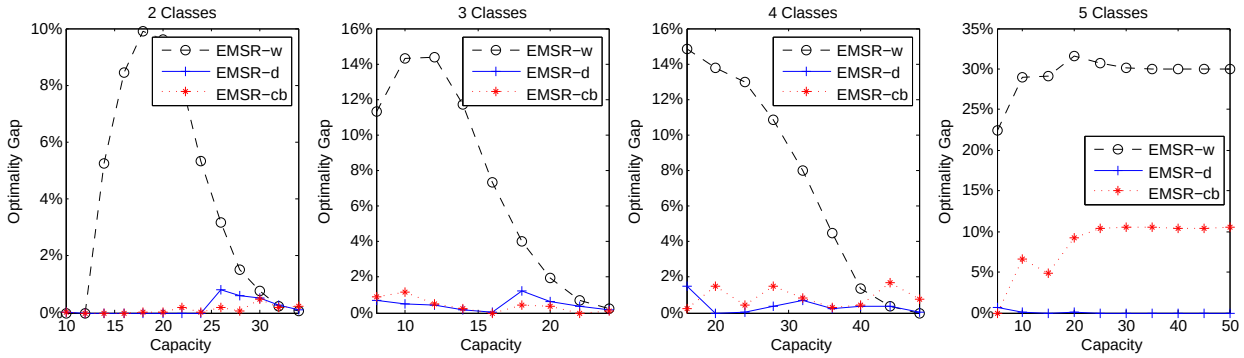

Figure 2: Optimality gap comparisons for two/three/four/five classes.

Figure 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 5), 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.

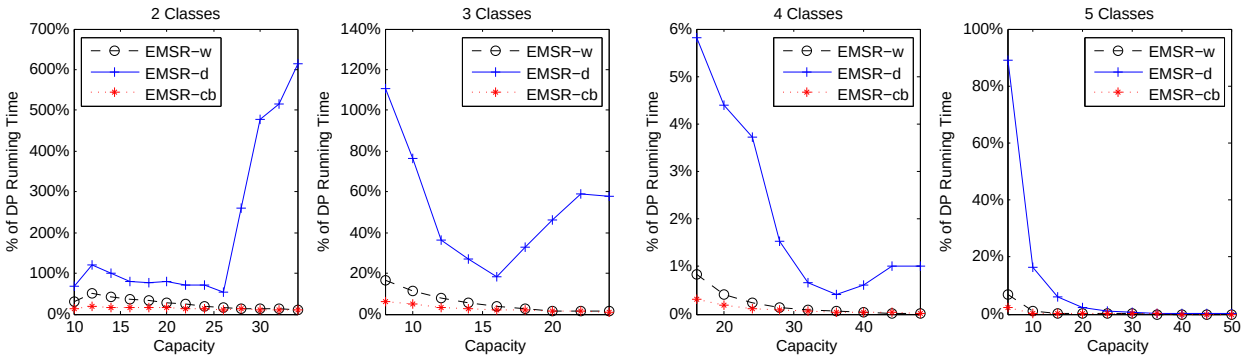

Figure 3: Percentage of the running time of DP for two/three/four/five classes from left to right, smaller the better.

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 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 3, and the percentages of running time increase are given in Table 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 6 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 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 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 (5) with piecewise linear functions, we need to accurately estimate marginal revenue at each itinerary allocation level. Figure 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.

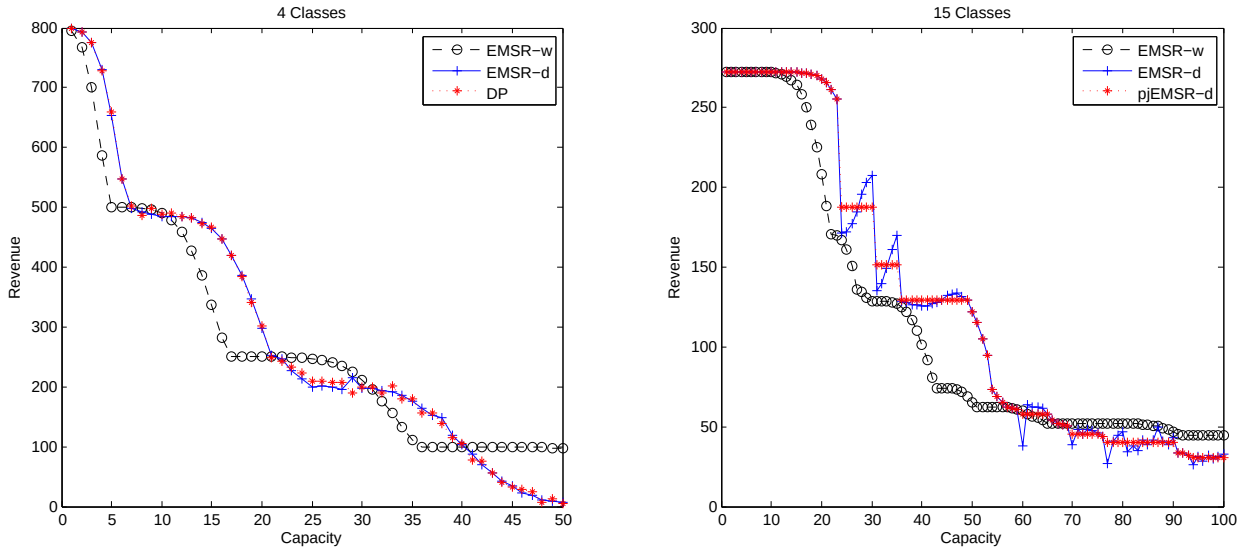

Figure 4: The marginal revenue curves for a four-class example (left) and a fifteen-class example (right).

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 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 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		

6.2.1 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.

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

6.2.2 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 7 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 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.

6.2.3 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 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.

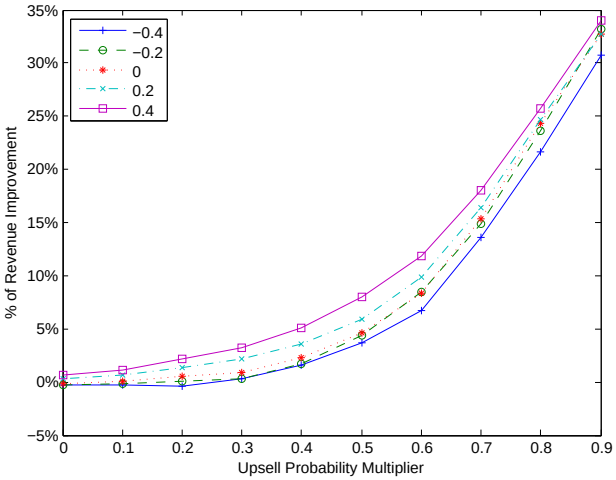

Figure 5: Percentage of revenue improvement against upsell probability multiplier when upsell probabilities are forecasted accurately.

Figure 6 and 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 8 in Appendix C for the exact numerical values. Figure 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 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.

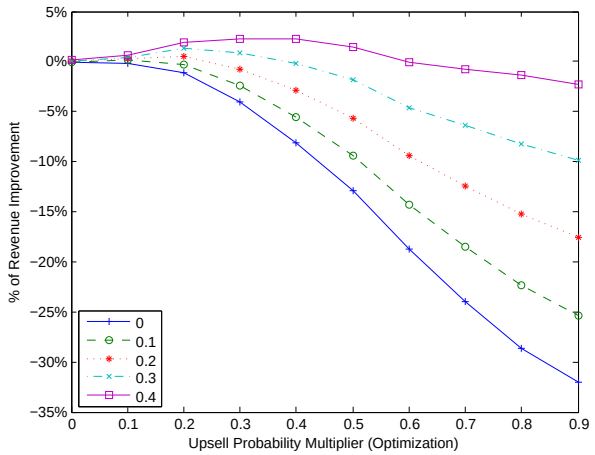

Figure 6: Percentage of revenue improvement against $m_2^{optimization}$ when $m_1 = 0$ and $m_2^{simulation} \leq 0.4$

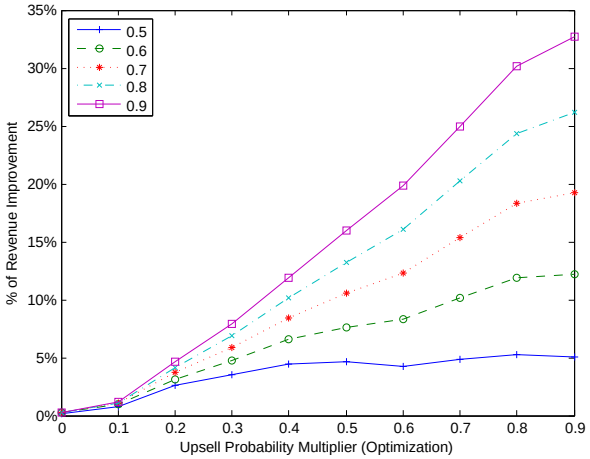

Figure 7: Percentage of revenue improvement against $m_3^{optimization}$ when $m_1 = 0$ and $m_3^{simulation} \geq 0.5$

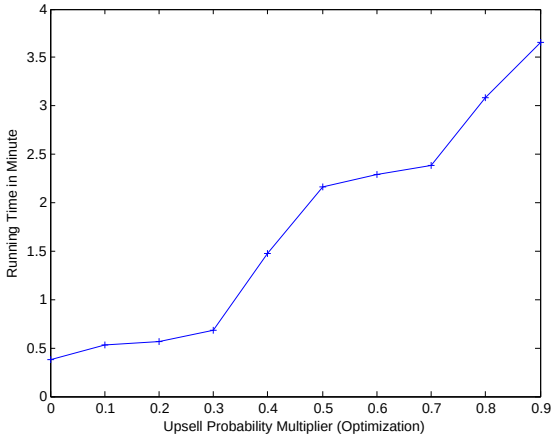

Figure 8: Average running time of the ADP algorithm in minute.

Figure 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.

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?

REFERENCES

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.

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

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

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.

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

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

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.

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.

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.

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.

Littlewood, K., editor (1972). *Forecasting and control of passenger bookings*, volume 12. The 12th AGIFORS Symposium, Nathanya, Israel.

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

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.

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

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

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

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

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

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.

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.

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

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.

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

APPENDICES

A Proofs

A.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:

$$\xi_i^c - \Pi_{ic-1} = x_{ic} + z_{ic+1} \text{ for } l \leq c \tag{10}$$

and

$$\varphi_{il} = \eta_{il}^c + \sum_{c'>l}^{c} U_{ic'l} \text{ for } l \leq c. \tag{11}$$

Equation (10) 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 (10) also implies

$$\xi_i^{c-1} = \sum_{c' \leq c-1} x_{ic'} + z_{ic}. \tag{12}$$

Equation (11) 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_{il}^c 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 (11) implies both

$$\eta_{ic}^c = \varphi_{ic} \tag{13}$$

and

$$\boldsymbol{\eta}_i^c + \mathbf{U}_{ic} = \boldsymbol{\eta}_i^{c-1}, \tag{14}$$

and if initial upsells are not given, equation (11) is essentially the same as constraint (8) for class l.

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

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

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

The second equality is by (10) and (13). The fourth equality is by definitions (6) and (7), and equations (11) and (12). The last equality is by (14) 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 $\xi_i^c \ge \Pi_{ic-1}$ by mapping $\mathcal{N}(\mathbf{x}_i)$ □

A.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 $\overline{SP}(\mathbf{K}) \ge \overline{P}(\mathbf{K})$ and $\overline{P}(\mathbf{K}) \ge \overline{SP}(\mathbf{K})$ 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 $\overline{SP}(\mathbf{K}) \ge \overline{P}(\mathbf{K})$.
2. Suppose we are now given $\mathbf{x}^{SP}$ an optimal solution to $\overline{SP}(\mathbf{K})$.
   (a) For itinerary i, if all $\bar{D}_{ic} \ge x_{ic}^{SP}$, then the corresponding $\mathbf{z}_i$ computed using (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 $\overline{SP}(\mathbf{K}) \le \overline{P}(\mathbf{K})$ as required.

□

A.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* $\mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^{HL}) \ge \mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^{R}) \ge \mathcal{R}^{P(\mathbf{K})}(\mathbf{D}^{LH})$.

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 (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. □

B Algorithms

B.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 (6) and (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 4, 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 3.

Algorithm 3 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 $n = 1, \dots N$.

Set $r = 0$, $\alpha_c^n = 0$, $\beta_c^n = 0$, and $\gamma_{cc'}^n = 0$ for $c, c' \in C$ and $n = 1, \dots, N$.
for each demand sample **do**
  Initialize $z = 0$, $r' = 0$, and $u_{cc'} = 0$ for $c, c' \in C$.
  for $c = |C|, \dots, 1$ **do**
    $\varphi = \sum_{c'=c+1}^{|C|} u_{c'c}$.
    $r' = r' + r_c \min\{x_c + z, \zeta_c^n + \varphi\}$.
    if $\zeta_c^n > \{x_c + z - \varphi\}^+$ **then**
      $\alpha_c^n = 1$.
      if c is not the highest class **then**
        Generate $\{u_{cc'}\}_{c'=1,\dots,|C|}$ based on $\mathcal{B}(\zeta_c^n - (x_c + z - \varphi)^+, \mathbf{p}(c))$.
        for $c' = 1, \dots, |C|$ **do**
          if $u_{cc'} > 0$ **then**
            $\gamma_{cc'} = 1$
          end if
        end for.
      end if
    end if
    if $\varphi > x_c + z$ **then**
      $\beta_c^n = 1$.
    end if
    $z = (x_c + z - \varphi - \zeta_c^n)^+$.
  end for
  $r = r + r'/N$.
end for
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 (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 (6) in step 21.

B.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 B.1 for definitions). It requires the same inputs as the upsell revenue estimation algorithm (Algorithm 3) 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 4.

Algorithm 4 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.

B.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 5.

Algorithm 5 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.

C Tables

This section includes all results used to create the figures in the main document. Table 6 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 6: 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 7 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 7: 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 8 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 8: 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%