Separable Approximations for Joint Capacity Control and Overbooking Decisions in Network Revenue Management

Alexander Erdelyi
School of Operations Research and Information Engineering,
Cornell University, Ithaca, New York 14853, USA
alex@orie.cornell.edu

Huseyin Topaloglu
School of Operations Research and Information Engineering,
Cornell University, Ithaca, New York 14853, USA
topaloglu@orie.cornell.edu

September 17, 2008

Abstract

We develop a network revenue management model to jointly make capacity control and overbooking decisions. Our approach is based on the observation that if the penalty cost of denying boarding to the reservations at the departure time were given by a separable function, then the dynamic programming formulation of the network revenue management problem would decompose by the itineraries and it could be solved by focusing on one itinerary at a time. Motivated by this observation, we use an iterative and simulation-based method to build separable approximations to the penalty cost that we incur at the departure time. Computational experiments compare our model with two benchmark strategies that are based on a deterministic linear programming formulation. The profits obtained by our model improve over those obtained by the benchmark strategies by about 3% on the average, which is a significant figure in the network revenue management setting. For the test problems with tight leg capacities, the profit improvements can be as high as 13%.

Capacity control and overbooking are two fundamental pieces of revenue management. Capacity control deals with the question of which itinerary requests should be accepted as the state of the reservations evolves over time. Overbooking deals with the question of by how much the physically available seats on a flight leg should be oversold given that the accepted itinerary requests may not show up at the departure time. It is clear that capacity control and overbooking decisions interact. Which itinerary requests we should accept depends on how much we are willing to overbook. How much we should overbook, in turn, depends on which itinerary requests we are willing to accept and the probability that these accepted itinerary requests show up at the departure time.

In this paper, we develop a network revenue management model to jointly make capacity control and overbooking decisions. We begin by formulating the problem as a dynamic program. This dynamic program involves a very high-dimensional state variable and cannot be solved by using traditional dynamic programming tools for problem instances of practical size. Our approach is aimed at finding approximate solutions to the dynamic program and it is based on the following observation. We let $\{1, \ldots, n\}$ be the set of possible itineraries, s_j be the number of reservations for itinerary j that show up at the departure time and $V(s)$ be the penalty cost of denying boarding to the reservations that show up at the departure time as a function of $s = (s_1, \ldots, s_n)$. The crucial observation is that if $V(\cdot)$ were a separable function, then the dynamic programming formulation of the problem would decompose by the itineraries and it could be solved by focusing on one itinerary at a time. Motivated by this observation, our approach uses an iterative and simulation-based method to construct separable approximations to $V(\cdot)$. We start with an arbitrary separable approximation. We solve the dynamic programming formulation of the problem by using the separable approximation to $V(\cdot)$ in the boundary condition of the dynamic program. This dynamic program can be solved efficiently as it decomposes by the itineraries and it provides approximations to the value functions. We then simulate the evolution of the system over time by making the itinerary acceptance decisions through the value function approximations. The challenge is to use the information that we obtain from the simulated trajectories of the system to update and improve the separable approximation to $V(\cdot)$.

Our work here builds on the previous work by Powell, Ruszczynski and Topaloglu (2004), where the authors propose an iterative and simulation-based method to construct separable approximations to the recourse functions that arise in two-stage stochastic programs. We borrow their method to construct separable approximations to the penalty cost that we incur at the departure time. An important distinguishing aspect of our work is that Powell et al. (2004) work with two-stage stochastic programs, whereas the planning horizon for the network revenue management problem includes multiple time periods. The application of their approximation strategy to capacity control and overbooking decisions in the network revenue management setting is also quite valuable as it turns out that this application provides noticeably better solutions than the existing benchmark methods. In particular, we report average profit improvements on the order of 3%. For the test problems with tight leg capacities, the profit improvements can be as high as 13%.

There is extensive literature on capacity control decisions under the assumption that overbooking is not possible and all of the reservations show up at the departure time. Talluri and van Ryzin

(2004) give a comprehensive coverage of this literature. On the other hand, the literature is quite thin when we consider capacity control and overbooking decisions simultaneously. Early models consider a version of the problem that takes place over a single flight leg rather than an airline network. For example, Beckmann (1958), Thompson (1961) and Coughlan (1999) develop static capacity control and overbooking models over a single flight leg that ignore the temporal dynamics of the arrivals of the itinerary requests. These models generally treat the total demand from a fare class as a random variable and compute booking limits for the different fare classes. Chatwin (1992), Chatwin (1999) and Subramanian, Stidhan and Lautenbacher (1999) focus on dynamic models that capture the arrival process for the itinerary requests more realistically. In particular, Subramanian et al. (1999) formulate the capacity control and overbooking problem as a dynamic program and demonstrate that if the probability of showing up varies by the fare classes, then the number of dimensions of the state variable is as large as the number of possible fare classes. In contrast, the dynamic programming formulation for the capacity control problem over a single flight leg involves a single-dimensional state variable when overbooking is not possible. Karaesmen and van Ryzin (2004*b*) describe an overbooking model that is applicable when different flight legs can serve as substitutes for each other.

When overbooking is not possible, a traditional approach for making capacity control decisions over an airline network involves solving a deterministic linear program that is formulated under the assumption that the arrivals of the itinerary requests take on their expected values. Bertsimas and Popescu (2003) show how to extend this linear program to make overbooking decisions. In particular, their approach solves a linear program, which can be visualized as a deterministic approximation to the overbooking problem. They use the optimal values of the dual variables associated with the seat availability constraints in the linear program to estimate the opportunity costs of the seats on different flight legs. In this case, an itinerary request is accepted only when the revenue from the itinerary request exceeds the total expected opportunity cost of the capacities consumed by this itinerary request. Since the linear program used by Bertsimas and Popescu (2003) is deterministic, it may not accurately capture the temporal dynamics of the arrivals of the itinerary requests. The goal of our approach is to capture these dynamics somewhat more accurately by directly working with the dynamic programming formulation of the capacity control and overbooking problem. Karaesmen and van Ryzin (2004*a*) develop an overbooking model that use the deterministic linear program to estimate the total expected revenue from the accepted itinerary requests. Gallego and van Ryzin (1997) show that the decisions made through a version of the deterministic linear program are asymptotically optimal as the leg capacities and the expected numbers of itinerary requests increase linearly with the same rate. Kleywegt (2001) formulates a pricing and overbooking model over an airline network and uses duality ideas to solve the model. Lastly, Rothstein (1985) provides an interesting account of the issues revolving around overbooking and remains relevant to this date.

In this paper, we make the following research contributions. 1) We develop a network revenue management model to jointly make capacity control and overbooking decisions. Our model takes the probabilistic nature of the arrivals of the itinerary requests and the show up decisions of the reservations into consideration. 2) Our model is based on constructing a separable approximation to the penalty cost of denying boarding to the reservations. We develop an iterative and simulation-based algorithm

to construct such an approximation. 3) We show that the aforementioned deterministic linear program provides an upper bound on the optimal total expected profit. This result is widely known when overbooking is not possible and all of the reservations show up at the departure time, but we are not aware of a reference to this result in the presence of overbooking. 4) Our computational experiments indicate that the capacity control and overbooking decisions made by our model are significantly better than those made by benchmark methods that are based on the deterministic linear program.

The rest of the paper is organized as follows. In Section 1, we present a dynamic programming formulation of the network revenue management problem with overbooking. In Section 2, we give an overview of our separable approximation strategy. In Section 3, we explain how to iteratively update and improve the separable approximations. In Section 4, we go over the deterministic linear program and show that this linear program provides an upper bound on the optimal total expected profit. In Section 5, we provide computational experiments. In Section 6, we conclude.

1 Problem Formulation

We have a set of flight legs that can be used to satisfy the itinerary requests that arrive randomly over time. At each time period, an itinerary request arrives and we need to decide whether to accept or reject the itinerary request. An accepted itinerary request generates a revenue and becomes a reservation, whereas a rejected itinerary request simply leaves the system. At the departure time of the flight legs, each reservation shows up with a certain probability and we need to decide which reservations should be allowed boarding. We incur a penalty cost for each reservation that is denied boarding. The objective is to maximize the total expected profit, which is the difference between the expected revenue from accepting the itinerary requests and the expected penalty cost of denying the reservations.

The itinerary requests arrive over the time periods $\{\tau, \ldots, 1\}$ and the flight legs depart at time period 0. We follow the standard convention in the network revenue management literature to index the time periods backwards. The set of flight legs is $\{1, \ldots, m\}$ and the set of itineraries is $\{1, \ldots, n\}$. The capacity on flight leg i is c_i. The probability that there is a request for itinerary j at time period t is p_{jt}. If we accept a request for itinerary j, then we generate a revenue of f_j and this reservation shows up at the departure time with probability q_j. If we allow boarding to a reservation for itinerary j, then we consume a_{ij} units of capacity on flight leg i. The penalty cost of denying boarding to a reservation for itinerary j is γ_j. We assume that the arrivals of the itinerary requests at different time periods and the show up decisions of different reservations are independent. For clarity of the presentation, we assume that the reservations are never canceled over the time periods $\{\tau, \ldots, 1\}$. However, all of the development in the paper goes through with small modifications in the presence of cancellations, as long as the cancellation decisions of different reservations and the cancellation decisions at different time periods are independent. Finally, we assume that we do not give refunds to the reservations that do not show up at the departure time, but this assumption is also for clarity of the presentation and it is straightforward to incorporate refunds.

We let x_{jt} be the total number of reservations for itinerary j at the beginning of time period t and use $x_t = (x_{1t}, \ldots, x_{nt})$ to capture the state of the reservations. Given that the number of

reservations for itinerary j at the beginning of time period 0 is x_{j0}, we use $s_{j0}(x_{j0})$ to denote the number of reservations for itinerary j that show up at the departure time. Due to the assumption that the show up decisions of different reservations are independent of each other, $s_{j0}(x_{j0})$ has a binomial distribution with parameters (x_{j0}, q_j). If the reservations that show up at the departure time are given by $s_0(x_0) = (s_{10}(x_{10}), \ldots, s_{n0}(x_{n0}))$, then we can compute the penalty cost of denying the reservations by solving the problem

$$V(s_0(x_0)) = \min \quad \sum_{j=1}^{n} \gamma_j \, y_j \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} \left[s_{j0}(x_{j0}) - y_j \right] \leq c_i \qquad i = 1, \ldots, m \tag{2}$$

$$y_j \leq s_{j0}(x_{j0}) \qquad j = 1, \ldots, n \tag{3}$$

$$y_j \in \mathbb{Z}_+ \qquad j = 1, \ldots, n, \tag{4}$$

where y_j is the number of reservations for itinerary j that we deny boarding. The objective function in problem (1)-(4) corresponds to the penalty cost of denying the reservations. Constraints (2) ensure that the reservations that we allow boarding do not violate the leg capacities, whereas constraints (3) ensure that the reservations that we deny boarding do not exceed the reservations that show up. Using x_t as the state variable at time period t and letting e_j be the n-dimensional unit vector with a one in the element corresponding to itinerary j, we can find the optimal policy by computing the value functions through the optimality equation

$$u_t(x_t) = \sum_{j=1}^{n} p_{jt} \max\{f_j + u_{t-1}(x_t + e_j), u_{t-1}(x_t)\} + \left[1 - \sum_{j=1}^{n} p_{jt}\right] u_{t-1}(x_t) \tag{5}$$

with the boundary condition that $u_0(x_0) = -\mathbb{E}\{V(s_0(x_0))\}$. We note that $V(\cdot)$ accounts for a cost figure, whereas the value functions $\{u_\tau(\cdot), \ldots, u_0(\cdot)\}$ account for a profit figure. Therefore, we negate the cost figure in the boundary condition of the optimality equation above. If the state of the reservations at the beginning of time period t is given by x_t and we have

$$f_j + u_{t-1}(x_t + e_j) \geq u_{t-1}(x_t), \tag{6}$$

then it is optimal to accept a request for itinerary j at time period t.

We note that the state variable in the optimality equation in (5) may involve many dimensions in practical applications and solving this optimality equation through traditional dynamic programming tools can be computationally difficult. In the next section, we develop a method to construct tractable approximations to the value functions.

2 Separable Approximations

The method that we use to construct approximations to the value functions is based on the observation that if $V(\cdot)$ is a separable function of the form

$$V(s_0(x_0)) = \sum_{j=1}^{n} V_j(s_{j0}(x_{j0})), \tag{7}$$

then the optimality equation in (5) decomposes by the itineraries and it can be solved efficiently by focusing on one itinerary at a time. This observation motivates us to build separable approximations to $V(\cdot)$ that have the form $\hat{V}(s_0(x_0)) = \sum_{j=1}^{n} \hat{V}_j(s_{j0}(x_{j0}))$. In this case, it is possible to solve the optimality equation in (5) with the approximate boundary condition that $u_0(x_0) = -\mathbb{E}\{\hat{V}(s_0(x_0))\}$.

We begin with the next proposition, which formally shows that if $V(\cdot)$ is a separable function of the form (7), then the optimality equation in (5) decomposes by the itineraries.

Proposition 1 *If $V(\cdot)$ is a separable function of the form (7), then the value functions $\{u_\tau(\cdot), \ldots, u_0(\cdot)\}$ computed through the optimality equation in (5) are separable functions of the form*

$$u_t(x_t) = \sum_{j=1}^{n} u_{jt}(x_{jt}),$$

where $\{u_{j\tau}(\cdot), \ldots, u_{j0}(\cdot)\}$ are computed through the optimality equation

$$u_{jt}(x_{jt}) = p_{jt} \max\{f_j + u_{j,t-1}(x_{jt}+1), u_{j,t-1}(x_{jt})\} + [1 - p_{jt}]\, u_{j,t-1}(x_{jt}) \tag{8}$$

with the boundary condition that $u_{j0}(x_{j0}) = -\mathbb{E}\{V_j(s_{j0}(x_{j0}))\}$.

Proof We show the result by induction over the time periods. By definition, we have $u_0(x_0) = -\mathbb{E}\{V(s_0(x_0))\} = -\sum_{j=1}^{n} \mathbb{E}\{V_j(s_{j0}(x_{j0}))\} = \sum_{j=1}^{n} u_{j0}(x_{j0})$ and the result holds for time period 0. Assuming that the result holds for time period $t-1$, we have

$$\begin{aligned}
u_t(x_t) &= \sum_{j=1}^{n} p_{jt} \max\{f_j + u_{t-1}(x_t + e_j) - u_{t-1}(x_t), 0\} + u_{t-1}(x_t) \\
&= \sum_{j=1}^{n} p_{jt} \max\{f_j + u_{j,t-1}(x_{jt}+1) - u_{j,t-1}(x_{jt}), 0\} + \sum_{j=1}^{n} u_{j,t-1}(x_{jt}) \\
&= \sum_{j=1}^{n} p_{jt} \max\{f_j + u_{j,t-1}(x_{jt}+1), u_{j,t-1}(x_{jt})\} + \sum_{j=1}^{n} [1 - p_{jt}]\, u_{j,t-1}(x_{jt}) \\
&= \sum_{j=1}^{n} u_{jt}(x_{jt}),
\end{aligned}$$

where the first equality follows by (5), the second equality follows by the induction assumption that $u_{t-1}(\cdot)$ is a separable function of the form $u_{t-1}(x_{t-1}) = \sum_{j=1}^{n} u_{j,t-1}(x_{j,t-1})$, the third equality follows by adding and subtracting $\sum_{j=1}^{n} p_{jt}\, u_{j,t-1}(x_{jt})$ and the fourth equality follows by (8). □

Thus, if $V(\cdot)$ is a separable function of the form (7), then the value functions $\{u_\tau(\cdot), \ldots, u_0(\cdot)\}$ can be computed by solving the optimality equation in (8). Since the optimality equation in (8) uses a single-dimensional state variable, it can be solved efficiently. We also note that the optimality equation in (8) corresponds to a network revenue management problem that involves only itinerary j. In this network revenue management problem, if the number of reservations for itinerary j that show up at the departure time is $s_{j0}(x_{j0})$, then we incur a penalty cost of $V_j(s_{j0}(x_{j0}))$.

Proposition 1 motivates using separable approximations to $V(\cdot)$. We emphasize that $V(\cdot)$ is, in general, not necessarily separable and using separable approximations to $V(\cdot)$ should be interpreted as a heuristic approach. We use an iterative and simulation-based method to construct separable approximations to $V(\cdot)$. Letting $\hat{V}^k(\cdot)$ be the separable approximation to $V(\cdot)$ at iteration k, we first obtain the value function approximations $\{\hat{u}_\tau^k(\cdot), \ldots, \hat{u}_0^k(\cdot)\}$ by solving the optimality equation in (5) with the boundary condition that $\hat{u}_0^k(x_0) = -\mathbb{E}\{\hat{V}^k(s_0(x_0))\}$. By Proposition 1, the value function approximations $\{\hat{u}_\tau^k(\cdot), \ldots, \hat{u}_0^k(\cdot)\}$ can be computed by focusing on one itinerary at a time. We then simulate the evolution of the system over the time periods $\{\tau, \ldots, 1\}$. In particular, we sample an itinerary request at each time period and decide whether to accept or reject this itinerary request by using $\{\hat{u}_\tau^k(\cdot), \ldots, \hat{u}_0^k(\cdot)\}$ as approximations to the value functions. At the departure time, we sample the reservations that show up and solve problem (1)-(4) to compute the penalty cost. The challenge is to use the information that is obtained by solving problem (1)-(4) to update and improve the separable approximation $\hat{V}^k(\cdot)$.

We give a description of our general approach in Figure 1. In Step 1, we begin by computing the value function approximations $\{\hat{u}_\tau^k(\cdot), \ldots, \hat{u}_0^k(\cdot)\}$. In Step 2, we simulate the behavior of the policy characterized by the value function approximations $\{\hat{u}_\tau^k(\cdot), \ldots, \hat{u}_0^k(\cdot)\}$. In Step 3, we update $\hat{V}^k(\cdot)$ to obtain the approximation $\hat{V}^{k+1}(\cdot)$ that we use at the next iteration. Our hope is that $\hat{V}^{k+1}(\cdot)$ approximates $V(\cdot)$ better than $\hat{V}^k(\cdot)$. For the moment, we do not specify how to update $\hat{V}^k(\cdot)$ and capture the updating procedure simply by the function $\mathcal{U}(\cdot,\cdot)$. This function takes the approximation $\hat{V}^k(\cdot)$ and the sample of the reservations that show up at the departure time, and returns a new approximation $\hat{V}^{k+1}(\cdot)$. The next section gives a precise description for the function $\mathcal{U}(\cdot,\cdot)$.

3 Updating Separable Approximations

We use two types of separable approximations to the penalty cost $V(\cdot)$ and both of these approximations are of the form $\hat{V}^k(s_0(x_0)) = \sum_{j=1}^n \hat{V}_j^k(s_{j0}(x_{j0}))$. In the first type of approximation, $\hat{V}_j^k(\cdot)$ is a linear function for all $j = 1, \ldots, n$. In the second type of approximation, $\hat{V}_j^k(\cdot)$ is a piecewise linear and convex function for all $j = 1, \ldots, n$.

The appealing aspect of linear and piecewise linear approximations is that such approximations can be characterized by a certain number of slope parameters. More specifically, we can characterize a linear approximation by a single slope parameter, whereas we can characterize a piecewise linear approximation by a sequence of slope parameters. The method that we use to update $\hat{V}_j^k(\cdot)$ is based on obtaining a new estimate of the slope of the penalty cost $V(\cdot)$ along the direction e_j and combining the new slope estimate with the slope parameters that characterize $\hat{V}_j^k(\cdot)$. In particular, if the reservations that show up at the departure time at iteration k are given by $s_0^k = (s_{10}^k, \ldots, s_{n0}^k)$, then we use

$$\vartheta_j^k(s_0^k) = V(s_0^k + e_j) - V(s_0^k) \tag{9}$$

to estimate the slope of the penalty cost $V(\cdot)$ along the direction e_j. The goal is to combine $\vartheta_j^k(s_0^k)$ with the slope parameters that characterize $\hat{V}_j^k(\cdot)$ and come up with the approximation $\hat{V}_j^{k+1}(\cdot)$. We consider updating linear and piecewise linear approximations separately in the next two sections.

Step 0 Initialize the separable approximation $\hat{V}^1(\cdot)$ arbitrarily. Set the iteration counter k to 1.

Step 1 Compute the value function approximations $\{\hat{u}^k_\tau(\cdot), \ldots, \hat{u}^k_0(\cdot)\}$ by solving the optimality equation in (5) with the boundary condition that $\hat{u}^k_0(x_0) = -\mathbb{E}\{\hat{V}^k(s_0(x_0))\}$. We note that since $\hat{V}^k(\cdot)$ is separable, we can actually compute $\{\hat{u}^k_\tau(\cdot), \ldots, \hat{u}^k_0(\cdot)\}$ through the optimality equation in (8).

Step 2 Initialize $x^k_\tau = (x^k_{1\tau}, \ldots, x^k_{n\tau})$ by setting $x^k_{j\tau} = 0$ for all $j = 1, \ldots, n$. Set the time counter t to τ.

Step 2.a Sample the itinerary request j^k_t at time period t by using the probabilities $(p_{1t}, \ldots, p_{nt})$. We note that there may not be any itinerary requests at a particular time period.

Step 2.b Letting $\mathbf{1}(\cdot)$ be the indicator function, if there is an itinerary request at time period t, then compute the state of the reservations $x^k_{t-1} = (x^k_{1,t-1}, \ldots, x^k_{n,t-1})$ at the next time period by

$$x^k_{t-1} = x^k_t + \mathbf{1}(f_{j^k_t} + \hat{u}^k_{t-1}(x^k_t + e_{j^k_t}) \geq \hat{u}^k_{t-1}(x^k_t))\, e_{j^k_t}.$$

Otherwise, let $x^k_{t-1} = x^k_t$.

Step 2.c Decrease t by 1. If we have $t \geq 1$, then go to Step 2.a.

Step 2.d Sample s^k_{j0} from the binomial distribution with parameters (x^k_{j0}, q_j) for all $j = 1, \ldots, n$ so that the reservations that show up at the departure time are given by $s^k_0 = (s^k_{10}, \ldots, s^k_{n0})$. Compute $V(s^k_0)$ by solving problem (1)-(4).

Step 3 Use the information that is obtained by solving problem (1)-(4) to update the separable approximation $\hat{V}^k(\cdot)$. For the moment, we capture the updating procedure by $\hat{V}^{k+1}(\cdot) = \mathcal{U}(\hat{V}^k(\cdot), s^k_0)$.

Step 4 Increase k by 1. Letting K be a fixed iteration counter limit, if we have $k \leq K$, then go to Step 1. Otherwise, return $\hat{V}^{K+1}(\cdot)$ and stop.

Figure 1: An iterative and simulation-based method to construct separable approximations to $V(\cdot)$.

3.1 Updating Linear Approximations

In this section, we assume that $\hat{V}^k_j(\cdot)$ is a linear function of the form $\hat{V}^k_j(s_{j0}(x_{j0})) = \hat{v}^k_j\, s_{j0}(x_{j0})$, where $\hat{v}^k_j$ is the slope parameter. Since $\hat{V}^k_j(\cdot)$ is eventually embedded in an optimization problem, its intercept does not affect the decisions and we set the intercept of $\hat{V}^k_j(\cdot)$ to zero without loss of generality. We use linear approximations primarily due to their simplicity, but we expect their performance to be not as good as the performance of piecewise linear approximations.

The method that we use to update linear approximations is quite straightforward. Assuming that $\hat{V}^k_j(\cdot)$ is a linear function of the form $\hat{V}^k_j(s_{j0}(x_{j0})) = \hat{v}^k_j\, s_{j0}(x_{j0})$ and letting $\vartheta^k_j(s^k_0)$ be as in (9), we compute v^{k+1}_j through

$$\hat{v}^{k+1}_j = \hat{v}^k_j + \alpha^k\, [\vartheta^k_j(s^k_0) - \hat{v}^k_j], \tag{10}$$

where $\alpha^k \geq 0$ is the step size parameter at iteration k. We use the approximation $\hat{V}^{k+1}_j(s_{j0}(x_{j0})) =$

$\hat{v}_j^{k+1}\, s_{j0}(x_{j0})$ at the next iteration. The updating procedure in (10) is based on the observation that if the new estimate of the slope of the penalty cost $V(\cdot)$ along the direction e_j is greater than $\hat{v}_j^k$, then we increase $\hat{v}_j^k$. Otherwise, we decrease $\hat{v}_j^k$.

3.2 Updating Piecewise Linear Approximations

In this section, we assume that $\hat{V}_j^k(\cdot)$ is a piecewise linear and convex function and it is nondifferentiable only at the positive integers. Assuming that the number of reservations for itinerary j that show up at the departure time is bounded by R_j, the relevant domain of $\hat{V}_j^k(\cdot)$ is $\{0,\ldots,R_j\}$. We characterize $\hat{V}_j^k(\cdot)$ by the slope parameters $\hat{v}_j^k = (\hat{v}_j^k(0),\ldots,\hat{v}_j^k(R_j-1))$, where $\hat{v}_j^k(r)$ is the slope of $\hat{V}_j^k(\cdot)$ over the interval $(r, r+1)$. Noting that the number of itinerary requests is bounded by τ, we can let $R_j = \tau$, but it is usually possible to use the arrival process for the itinerary requests to get a tighter bound on the number of reservations that show up at the departure time. Since $\hat{V}_j^k(\cdot)$ is convex, we need to have $\hat{v}_j^k(0) \leq \hat{v}_j^k(1) \leq \ldots \leq \hat{v}_j^k(R_j-1)$. Our use of convex approximations is motivated by the intuitive expectation that the marginal penalty cost from an additional reservation showing up at the departure time increases as the number of reservations that show up increases.

The method that we use to update piecewise linear and convex approximations is similar to the one that we use for linear approximations, but we need to be careful to preserve convexity. Assuming that $\hat{V}_j^k(\cdot)$ is the piecewise linear and convex function characterized by the slope parameters $\hat{v}_j^k = (\hat{v}_j^k(0),\ldots,\hat{v}_j^k(R_j-1))$ and letting $e'(s_{j0}^k)$ be the R_j-dimensional unit vector with a one in the element corresponding to s_{j0}^k, we compute $w_j^k = (w_j^k(0),\ldots,w_j^k(R_j-1))$ through

$$w_j^k = \hat{v}_j^k + \alpha^k \left[\vartheta_j^k(s_0^k) - \hat{v}_j^k(s_{j0}^k)\right] e'(s_{j0}^k), \tag{11}$$

where $\alpha^k \geq 0$ is the step size parameter at iteration k and $\vartheta_j^k(s_0^k)$ is as in (9). If we consider the element of w_j^k corresponding to s_{j0}^k, then we have $w_j^k(s_{j0}^k) = \hat{v}_j^k(s_{j0}^k) + \alpha^k \left[\vartheta_j^k(s_0^k) - \hat{v}_j^k(s_{j0}^k)\right]$ and the updating procedure in (11) for the element corresponding to s_{j0}^k is similar to the updating procedure in (10). For all of the other elements of w_j^k, we have $w_j^k(r) = \hat{v}_j^k(r)$.

We do not necessarily have $w_j^k(0) \leq w_j^k(1) \leq \ldots \leq w_j^k(R_j-1)$ and the piecewise linear function characterized by the slope parameters $w_j^k = (w_j^k(0),\ldots,w_j^k(R_j-1))$ may not be convex. To preserve convexity, we define the set

$$\mathcal{V} = \{(z(0),\ldots,z(R_j-1)) \in \mathbb{R}^{R_j} : z(0) \leq z(1) \leq \ldots \leq z(R_j-1)\}$$

and let

$$\hat{v}_j^{k+1} = \operatorname*{argmin}_{z\in\mathcal{V}} \|z - w_j^k\|_2, \tag{12}$$

where $\|\cdot\|_2$ is the Euclidean norm on $\mathbb{R}^{R_j}$. We have $\hat{v}_j^{k+1}(0) \leq \hat{v}_j^{k+1}(1) \leq \ldots \leq \hat{v}_j^{k+1}(R_j-1)$ and the piecewise linear function $\hat{V}_j^{k+1}(\cdot)$ characterized by the slope parameters $\hat{v}_j^{k+1} = (\hat{v}_j^{k+1}(0),\ldots,\hat{v}_j^{k+1}(R_j-1))$ is convex. Problem (12) finds the piecewise linear and convex function that is "closest" to the piecewise linear function characterized by the slope parameters $w_j^k = (w_j^k(0),\ldots,w_j^k(R_j-1))$. We borrow the updating procedure in (11)-(12) from Powell et al. (2004), where the authors show that there is actually a closed-form solution to problem (12).

4 Deterministic Linear Program

An alternative solution method for the network revenue management problem described in Section 1 involves solving a deterministic linear program that is formulated under the assumption that the itinerary requests and the reservations that show up at the departure time take on their expected values. In particular, letting z_j be the number of requests for itinerary j that we plan to accept over the time periods $\{\tau, \ldots, 1\}$ and y_j be the number of reservations for itinerary j that we plan to deny boarding, this linear program has the form

$$\max \quad \sum_{j=1}^{n} f_j \, z_j - \sum_{j=1}^{n} \gamma_j \, y_j \tag{13}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} \, [q_j \, z_j - y_j] \leq c_i \qquad i = 1, \ldots, m \tag{14}$$

$$z_j \leq \sum_{t=1}^{\tau} p_{jt} \qquad j = 1, \ldots, n \tag{15}$$

$$y_j - q_j \, z_j \leq 0 \qquad j = 1, \ldots, n \tag{16}$$

$$z_j, \, y_j \geq 0 \qquad j = 1, \ldots, n. \tag{17}$$

The objective function in problem (13)-(17) corresponds to the difference between the revenue from accepting the itinerary requests and the penalty cost of denying the reservations. Problem (13)-(17) assumes that if we accept z_j requests for itinerary j, then $q_j \, z_j$ reservations for itinerary j show up at the departure time. Constraints (14) ensure that the reservations that we allow boarding do not violate the leg capacities. Constraints (15) ensure that the itinerary requests that we plan to accept do not exceed the expected numbers of the itinerary requests. Constraints (16) ensure that the reservations that we deny boarding do not exceed the reservations that show up.

The deterministic linear programming formulation for the network revenue management problem is widely known in the literature when overbooking is not possible and all of the reservations show up at the departure time; see Williamson (1992) and Talluri and van Ryzin (1998). Problem (13)-(17) extends this formulation to handle overbooking. Although this extension is quite intuitive, there does not appear to be too many references to problem (13)-(17) in the literature. As a matter of fact, the only reference we are aware of is Bertsimas and Popescu (2003).

There are two main uses of problem (13)-(17). First, this problem can be used to decide whether to accept or reject the itinerary requests. In particular, letting $\mu^* = (\mu_1^*, \ldots, \mu_m^*)$ be the optimal values of the dual variables associated with constraints (14) in problem (13)-(17), we can use μ_i^* to capture the opportunity cost of a unit of capacity on flight leg i. In this case, if the revenue from an itinerary request exceeds the total expected opportunity cost of the capacities consumed by the itinerary request or if the revenue from an itinerary request exceeds the expected penalty cost, then we accept the itinerary request. Specifically, if we have

$$f_j \geq \min \left\{ q_j \sum_{i=1}^{m} a_{ij} \, \mu_i^*, \, q_j \, \gamma_j \right\}, \tag{18}$$

then we accept a request for itinerary j. The decision rule above captures two effects. If the total expected opportunity cost of the seats consumed by an itinerary request is relatively small, then we tend to accept this itinerary request. However, even if the total expected opportunity cost of the seats consumed by an itinerary request is quite large, we may still be willing to accept the itinerary request, as long as the expected penalty cost is relatively small. If we have $f_j \geq q_j \gamma_j$, then we can, in expectation, generate revenue simply by accepting a request for itinerary j and denying the reservation at the departure time. The decision rule in (18) is also used by Bertsimas and Popescu (2003).

The second use of problem (13)-(17) is that its optimal objective value provides an upper bound on the optimal total expected profit. In other words, letting Z_{LP}^* be the optimal objective value of problem (13)-(17) and $\bar{0}$ be the n-dimensional vector of zeros, we have $u_\tau(\bar{0}) \leq Z_{LP}^*$. This result is widely known when overbooking is not possible and all of the reservations show up at the departure time; see Talluri and van Ryzin (1998). We have not seen a reference to this result in the presence of overbooking and the next proposition provides a formal proof.

Proposition 2 *We have* $u_\tau(\bar{0}) \leq Z_{LP}^*$.

Proof We let Z_j^* be the total number of requests for itinerary j that we accept over the time periods $\{\tau, \ldots, 1\}$ under the optimal policy and Y_j^* be the number of reservations for itinerary j that we deny boarding at the departure time under the optimal policy. We also let $s_j^*(k)$ take value 1 if the kth reservation for itinerary j shows up at the departure time, and take value 0 otherwise. In this case, the number of reservations for itinerary j that show up under the optimal policy can be written as

$$S_j^* = \sum_{k=1}^{Z_j^*} s_j^*(k).$$

Conditioning on Z_j^*, we note that $\mathbb{E}\{S_j^*\} = \mathbb{E}\{\sum_{k=1}^{Z_j^*} s_j^*(k)\} = \mathbb{E}\{\mathbb{E}\{\sum_{k=1}^{Z_j^*} s_j^*(k) \,|\, Z_j^*\}\} = \mathbb{E}\{q_j Z_j^*\} = q_j \mathbb{E}\{Z_j^*\}$. Letting D_j be the total number of requests for itinerary j over the time periods $\{\tau, \ldots, 1\}$, we also have

$$\sum_{j=1}^{n} a_{ij} [S_j^* - Y_j^*] \leq c_i \qquad i = 1, \ldots, m \tag{19}$$

$$Z_j^* \leq D_j \qquad j = 1, \ldots, n \tag{20}$$

$$Y_j^* \leq S_j^* \qquad j = 1, \ldots, n, \tag{21}$$

where (19) follows from the fact that the reservations that we allow boarding under the optimal policy do not violate the leg capacities, (20) follows from the fact that the itinerary requests that we accept under the optimal policy do not exceed the itinerary requests and (21) follows from the fact that the reservations that we deny under the optimal policy do not exceed the reservations that show up.

The total profit under the optimal policy is $\sum_{j=1}^n f_j Z_j^* - \sum_{j=1}^n \gamma_j Y_j^*$ so that the optimal total expected profit is $u_\tau(\bar{0}) = \sum_{j=1}^n f_j \mathbb{E}\{Z_j^*\} - \sum_{j=1}^n \gamma_j \mathbb{E}\{Y_j^*\}$. Taking expectations in (19)-(21) and noting that $\mathbb{E}\{S_j^*\} = q_j \mathbb{E}\{Z_j^*\}$ and $\mathbb{E}\{D_j\} = \sum_{t=1}^\tau p_{jt}$, it is easy to see that the solution $(\mathbb{E}\{Z_1^*\}, \ldots, \mathbb{E}\{Z_n^*\})$,

$(\mathbb{E}\{Y_1^*\},\ldots,\mathbb{E}\{Y_n^*\})$ is feasible to problem (13)-(17) and provides the objective value $\sum_{j=1}^n f_j\,\mathbb{E}\{Z_j^*\} - \sum_{j=1}^n \gamma_j\,\mathbb{E}\{Y_j^*\}$. Therefore, we have $Z_{LP}^* \geq \sum_{j=1}^n f_j\,\mathbb{E}\{Z_j^*\} - \sum_{j=1}^n \gamma_j\,\mathbb{E}\{Y_j^*\} = u_\tau(\bar{0})$. □

The upper bound in Proposition 2 may be useful when assessing the optimality gap of a suboptimal decision rule such as the one in (18).

5 Computational Experiments

In this section, we compare our separable approximation strategy with benchmark strategies that are based on the deterministic linear program. We begin by describing the benchmark strategies and the experimental setup. We then present our computational results.

5.1 Benchmark Strategies

We compare the performances of the following four benchmark strategies.

Linear approximations (LSA) This solution method is the separable approximation strategy with linear approximations. In our implementation, LSA divides the planning horizon into S equal segments and retunes the separable approximation to $V(\cdot)$ at the beginning of each segment. Given that the state of the reservations at the beginning of the lth segment is $x_{\tau(S-l+1)/S}$, we carry out the algorithm in Figure 1 by simulating the evolution of the system over the time periods $\{\tau(S-l+1)/S,\ldots,1\}$ and starting with the initial state of the reservations $x_{\tau(S-l+1)/S}$. This provides a separable approximation to $V(\cdot)$, say $\hat{V}(\cdot)$. Using the boundary condition that $\hat{u}_0(x_0) = -\mathbb{E}\{\hat{V}(s_0(x_0))\}$, we solve the optimality equation in (5) to obtain the value function approximations $\{\hat{u}_{\tau(S-l+1)/S}(\cdot),\ldots,\hat{u}_0(\cdot)\}$. We make the decisions by using these value function approximations in the decision rule in (6) until we reach the beginning of the next segment and retune the separable approximation to $V(\cdot)$. After some preliminary experimentation, we decided to use $K = 2,500$ in the algorithm in Figure 1.

Piecewise linear approximations (PSA) This solution method is the separable approximation strategy with piecewise linear approximations. Similar to LSA, PSA divides the planning horizon into S equal segments and retunes the separable approximation to $V(\cdot)$ at the beginning of each segment.

Deterministic linear program (DLP) This solution method uses the deterministic linear program in (13)-(17). DLP divides the planning horizon into S equal segments and resolves problem (13)-(17) at the beginning of each segment to retune the decision rule in (18). Given that the state of the reservations at the beginning of the lth segment is $x_{\tau(S-l+1)/S}$, we replace the right side of constraints (14) with $\{c_i - \sum_{j=1}^n a_{ij}\,q_j\,x_{j,\tau(S-l+1)/S} : i = 1,\ldots,m\}$, the right side of constraints (15) with $\{\sum_{t=1}^{\tau(S-l+1)/S} p_{jt} : j = 1,\ldots,n\}$ and the right side of constraints (16) with $\{q_j\,x_{j,\tau(S-l+1)/S} : j = 1,\ldots,n\}$, and solve problem (13)-(17). Letting $\mu^* = (\mu_1^*,\ldots,\mu_m^*)$ be the optimal values of the dual variables associated with constraints (14), we use these values in the decision rule in (18) until we resolve problem (13)-(17) at the beginning of the next segment.

Finite differences in the deterministic linear program (FDD) This solution method is due to Bertsimas and Popescu (2003) and it is also based on problem (13)-(17). Given that the state of the

reservations at the beginning of time period t is x_t, FDD approximates the optimal total expected profit over the remaining time periods by using the optimal objective value of the problem

$$\begin{aligned}
\max \quad & \sum_{j=1}^{n} f_j\, z_j - \sum_{j=1}^{n} \gamma_j\, y_j \\
\text{subject to} \quad & \sum_{j=1}^{n} a_{ij}\, [q_j\, z_j - y_j] \le c_i - \sum_{j=1}^{n} a_{ij}\, q_j\, x_{jt} && i = 1, \ldots, m \\
& z_j \le \sum_{t'=1}^{t} p_{jt'} && j = 1, \ldots, n \\
& y_j - q_j\, z_j \le q_j\, x_{jt} && j = 1, \ldots, n \\
& z_j,\, y_j \ge 0 && j = 1, \ldots, n.
\end{aligned}$$

Therefore, if we use $L_t(x_t)$ to denote the optimal objective value of the problem above, then FDD uses $L_t(x_t)$ as an approximation to $u_t(x_t)$. In this case, we can make the decisions by replacing $\{u_\tau(\cdot), \ldots, u_0(\cdot)\}$ in the decision rule in (6) with $\{L_\tau(\cdot), \ldots, L_0(\cdot)\}$.

Similar to DLP, FDD divides the planning horizon into S equal segments and retunes the decision rule at the beginning of each segment. Given that the state of the reservations at the beginning of the lth segment is $x_{\tau(S-l+1)/S}$, we compute $L_{\tau(S-l+1)/S}(x_{\tau(S-l+1)/S}) - L_{\tau(S-l+1)/S}(x_{\tau(S-l+1)/S} + e_j)$ for all $j = 1, \ldots, n$. Following the decision rule in (6), if we have

$$f_j + L_{\tau(S-l+1)/S}(x_{\tau(S-l+1)/S} + e_j) \ge L_{\tau(S-l+1)/S}(x_{\tau(S-l+1)/S}),$$

then we always accept a request for itinerary j until we reach the beginning of the next segment and retune the decision rule.

5.2 Experimental Setup

We consider an airline network that serves N spokes through a single hub. This is a key network structure that frequently arises in practice. There are two flight legs associated with each spoke. One of these flight legs is from the spoke to the hub and the other one is from the hub to the spoke. There is a high-fare and a low-fare itinerary that connects each origin-destination pair. Therefore, the number of flight legs is $2N$ and the number of itineraries is $2N(N+1)$. Figure 2 shows the structure of the airline network when we have eight spokes. The fare associated with a high-fare itinerary is κ times the fare associated with the corresponding low-fare itinerary. The penalty cost of denying boarding to a reservation for itinerary j is given by $\gamma_j = \theta\, f_j + \sigma \max\{f_{j'} : j' = 1, \ldots, n\}$, where θ and σ are two parameters that we vary. The goal of the parameter σ is to capture a situation where a denied reservation is given σ open tickets, in which case, the opportunity cost of denying boarding to a reservation can be as high as the revenue from σ tickets for the most expensive itinerary. The probability that a reservation shows up at the departure time is q and it does not depend on the itinerary. Noting that the total expected demand for the capacity on flight leg i is given by $\sum_{t=1}^{\tau} \sum_{j=1}^{n} a_{ij}\, q_j\, p_{jt}$, we measure the tightness of the leg capacities by

$$\rho = \frac{\sum_{i=1}^{m} \sum_{t=1}^{\tau} \sum_{j=1}^{n} a_{ij}\, q_j\, p_{jt}}{\sum_{i=1}^{m} c_i}.$$

We label our test problems by $(N, \kappa, \theta, \sigma, q, \rho)$ and use $N \in \{4, 8\}$, $\kappa \in \{4, 8\}$, $(\theta, \sigma) \in \{(4, 0), (8, 0), (1, 1)\}$, $q \in \{0.90, 0.95\}$ and $\rho \in \{1.2, 1.6\}$. This provides 48 test problems for our experimental setup. In all of our test problems, we have $\tau = 240$. Therefore, we have about 240 itinerary requests over the planning horizon. Considering the structure of the airline network and the tightness of the leg capacities, 240 itinerary requests allow us to have leg capacities ranging from 16 to 50.

After some preliminary runs, we decided to use the step size parameter $\alpha^k = 200/(400 + k)$ for LSA and $\alpha^k = 20/(40 + k)$ for PSA. We terminate the algorithm in Figure 1 after 2,500 iterations. We use $S = 5$ for LSA and PSA, and $S = 20$ for DLP and FDD. Some preliminary runs showed that increasing S further beyond 20 does not improve the performances of DLP and FDD significantly.

5.3 Computational Results

Our main computational results are summarized in Tables 1 and 2. Tables 1 and 2 respectively show the results for the test problems with four and eight spokes. The first column in these tables gives the characteristics of the test problem. The second column gives the upper bound on the optimal total expected profit provided by the optimal objective value of problem (13)-(17). The next four columns give the total expected profits obtained by using the decision rules provided by LSA, PSA, DLP and FDD. These total expected profits are estimated by simulating the performances of the different solution methods under multiple demand trajectories. We use common random numbers when simulating the performances of the different solution methods. The seventh column gives the percent gap between the total expected profits obtained by LSA and DLP. This column also includes a "✓" whenever LSA performs better than DLP, a "×" whenever DLP performs better than LSA and a "⊙" whenever there is no statistically significant performance gap between LSA and DLP at 95% level. The interpretations of the last three columns are similar to that of the seventh column, but the last three columns compare LSA with FDD, PSA with DLP and PSA with FDD.

The results indicate that PSA performs significantly better than the other solution methods. The performance gap between PSA and DLP can be as high as 13% and the performance gap between PSA and FDD can be as high as 7%. In all of our test problems, the performance of PSA is either statistically indistinguishable from or better than those of DLP and FDD. Although not quite as well as PSA, LSA performs reasonably well when compared with DLP and FDD. The performance of LSA is comparable to or better than that of FDD in 41 out of 48 test problems. FDD performs consistently better than DLP and this observation is in agreement with the computational experiments in Bertsimas and Popescu (2003). The success of PSA in comparison with DLP and FDD is especially remarkable considering that PSA retunes its decision rule five times over the planning horizon, whereas DLP and FDD retune their decision rules 20 times.

To give a feel for the problem characteristics that affect the performance gap between the different solution methods, Figure 3 plots the performance gap between PSA and FDD for the 24 test problems with eight spokes. The test problems in the horizontal axis of this figure are arranged in such a fashion that every block of two consecutive test problems differ only in the tightness of the leg capacities and every block of four consecutive test problems share the same penalty costs. Comparing blocks of

two consecutive test problems in Figure 3 indicates that the performance gap between PSA and FDD tends to increase as the leg capacities get tighter. For test problems with tight leg capacities, if we "mistakenly" accept a request for a low-fare itinerary, then we either deny boarding to this reservation at the departure time or forgo the opportunity of accepting a high-fare itinerary later on. For these test problems, PSA appears to make significantly better decisions. Comparing blocks of four consecutive test problems in Figure 3 indicates that the performance gap between PSA and FDD tends to increase as the penalty cost increases. For test problems with large penalty costs, if we overbook more than "necessary," then the only way to make up for this mistake is to deny boarding to the reservations and the cost of this mistake is high. For these test problems, PSA provides noticeably better performance than FDD. Similar observations hold for the test problems with four spokes.

An interesting aspect of LSA and PSA is their sensitivity to the frequency with which we retune their decision rules. For the 24 test problems with four spokes, the solid and dashed data series in Figure 4 respectively plot the improvements in the performances of LSA and PSA when we increase S from one to five. Figure 4 indicates that PSA is not very sensitive to the frequency with which we retune the decision rule and it can obtain good solutions even when we tune the decision rule only once at the beginning of the planning horizon. On the other hand, LSA appears to be quite sensitive. It is especially encouraging that PSA does not require frequent retuning. Although we do not present the detailed numbers here, it is worthwhile to mention that the performance of PSA with $S = 1$ is comparable to that of FDD with $S = 20$.

Table 3 shows the CPU seconds required to construct a separable approximation to $V(\cdot)$ by carrying out the algorithm in Figure 1 for 2,500 iterations. The second column in this table gives the total CPU seconds for LSA. We give a breakdown for these CPU seconds in the next two columns. The third column gives the CPU seconds spent by LSA on solving problem (1)-(4) to compute the penalty cost at the departure time, whereas the fourth column gives the CPU seconds spent by LSA on all other operations. The interpretations of the last three columns are similar, but the last three columns focus on the CPU seconds for PSA. The main factors that affect the CPU seconds are the number of spokes and whether we use linear or piecewise linear approximations. Therefore, we give the average CPU seconds over different test problems. All of our computational experiments are carried out on a Pentium IV PC running Windows XP with 2.4 GHz CPU and 1GB RAM. Table 3 indicates that the CPU seconds for LSA are substantially shorter than those for PSA. This is expected as PSA solves problem (12) to preserve convexity. For both LSA and PSA, the CPU seconds spent on solving problem (1)-(4) increase by about a factor of eight when we double the number of spokes. We note that the number of itineraries, and hence the number of decision variables in problem (1)-(4), increase by about a factor of four when we double the number of spokes. The CPU seconds spent on all other operations remain relatively stable when we increase the number of spokes. It turns out that the CPU seconds for solving problem (13)-(17) are much less than one second and we do not give the CPU seconds for DLP and FDD.

Figure 5 plots how the total expected profit obtained by PSA for test problem $(4, 8, 1, 1, 0.95, 1.6)$ changes as a function of the iteration counter limit K in the algorithm in Figure 1. The figure indicates

that the performance of PSA improves as we increase K, but the total expected profit is quite satisfactory even with $K = 250$. Our choice of $K = 2,500$ appears to be on the conservative side. We prefer to err on the conservative side, since it is not possible to predict the improvement in the performance of PSA without actually increasing the iteration counter limit and testing the new iteration counter limit through simulation. Nevertheless, practitioners, who are interested in a reasonable balance between solution quality and computational burden, may use less conservative stopping rules and be content with a smaller iteration counter limit.

6 Conclusions

In this paper, we developed a network revenue management model to jointly make capacity control and overbooking decisions. Our approach is based on the observation that if the penalty cost of denying boarding to the reservations at the departure time were given by a separable function, then the dynamic programming formulation of the problem would decompose by the itineraries. Using this observation, we constructed separable approximations to the penalty cost. Computational experiments demonstrated that our model can provide significantly better decisions than benchmark strategies that are based on the deterministic linear program.

The algorithm in Figure 1 represents only one approach for constructing separable approximations to the penalty cost. A natural research direction to pursue is to try to develop other approaches for constructing separable approximations. Better separable approximations to the penalty cost are likely to generate better capacity control and overbooking decisions.

References

Beckmann, M. J. (1958), 'Decision and team problems in airline reservations', *Econometrica* **26**(1), 134–145.

Bertsimas, D. and Popescu, I. (2003), 'Revenue management in a dynamic network environment', *Transportation Science* **37**, 257–277.

Chatwin, R. E. (1992), 'Multiperiod airline overbooking with a single fare class', *Operations Research* **46**(6), 805–819.

Chatwin, R. E. (1999), 'Continuous-time airline overbooking with time-dependent fares and refunds', *Transportation Science* **33**(2), 182–191.

Coughlan, J. (1999), 'Airline overbooking in the multi-class case', *The Journal of the Operational Research Society* **50**(11), 1098–1103.

Gallego, G. and van Ryzin, G. (1997), 'A multiproduct dynamic pricing problem and its applications to yield management', *Operations Research* **45**(1), 24–41.

Karaesmen, I. and van Ryzin, G. (2004*a*), Coordinating overbooking and capacity control decisions on a network, Technical report, Columbia Business School.

Karaesmen, I. and van Ryzin, G. (2004*b*), 'Overbooking with substitutable inventory classes', *Operations Research* **52**(1), 83–104.

Kleywegt, A. J. (2001), An optimal control problem of dynamic pricing, Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Powell, W. B., Ruszczynski, A. and Topaloglu, H. (2004), 'Learning algorithms for separable approximations of stochastic optimization problems', *Mathematics of Operations Research* **29**(4), 814–836.

Rothstein, M. (1985), 'OR and the airline overbooking problem', *Operations Research* **33**(2), 237–248.

Subramanian, J., Stidhan, S. and Lautenbacher, C. J. (1999), 'Airline yield management with overbooking, cancellations and no-shows', *Transportation Science* **33**(2), 147–167.

Talluri, K. T. and van Ryzin, G. J. (2004), *The Theory and Practice of Revenue Management*, Kluwer Academic Publishers.

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

Thompson, H. R. (1961), 'Statistical problems in airline reservation control', *OR* **12**(3), 167–185.

Williamson, E. L. (1992), Airline Network Seat Control, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

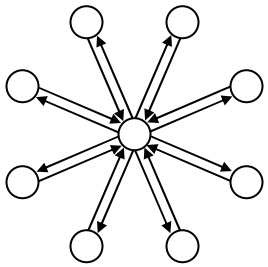

Figure 2: Structure of the airline network when we have eight spokes.

Problem	Profit	Tot. exp. profit obtained by				LSA vs.		PSA vs.	
$(N, \kappa, \theta, \sigma, q, \rho)$	bnd.	LSA	PSA	DLP	FDD	DLP	FDD	DLP	FDD
$(4, 4, 4, 0, 0.90, 1.2)$	15,223	14,383	14,442	14,283	14,386	0.70 ($\odot$)	-0.02 ($\odot$)	1.10 ($\checkmark$)	0.39 ($\odot$)
$(4, 4, 4, 0, 0.90, 1.6)$	20,997	19,281	19,506	19,085	19,395	1.02 ($\odot$)	-0.59 ($\odot$)	2.16 ($\checkmark$)	0.57 ($\checkmark$)
$(4, 4, 4, 0, 0.95, 1.2)$	23,450	21,778	22,062	21,962	21,998	-0.85 ($\times$)	-1.01 ($\times$)	0.45 ($\checkmark$)	0.29 ($\odot$)
$(4, 4, 4, 0, 0.95, 1.6)$	21,753	20,316	20,532	20,208	20,373	0.53 ($\odot$)	-0.28 ($\odot$)	1.58 ($\checkmark$)	0.77 ($\checkmark$)
$(4, 4, 8, 0, 0.90, 1.2)$	23,136	21,220	21,487	21,014	21,144	0.97 ($\odot$)	0.36 ($\odot$)	2.20 ($\checkmark$)	1.60 ($\checkmark$)
$(4, 4, 8, 0, 0.90, 1.6)$	12,177	10,733	11,047	10,326	10,620	3.79 ($\checkmark$)	1.05 ($\odot$)	6.53 ($\checkmark$)	3.86 ($\checkmark$)
$(4, 4, 8, 0, 0.95, 1.2)$	19,206	17,278	17,838	17,285	17,491	-0.04 ($\odot$)	-1.23 ($\times$)	3.10 ($\checkmark$)	1.95 ($\checkmark$)
$(4, 4, 8, 0, 0.95, 1.6)$	15,995	14,279	14,526	14,059	14,316	1.54 ($\checkmark$)	-0.25 ($\odot$)	3.22 ($\checkmark$)	1.45 ($\checkmark$)
$(4, 4, 1, 1, 0.90, 1.2)$	18,418	16,519	16,894	16,362	16,529	0.95 ($\odot$)	-0.06 ($\odot$)	3.15 ($\checkmark$)	2.17 ($\checkmark$)
$(4, 4, 1, 1, 0.90, 1.6)$	10,626	9,631	9,727	9,159	9,372	4.90 ($\checkmark$)	2.69 ($\checkmark$)	5.84 ($\checkmark$)	3.65 ($\checkmark$)
$(4, 4, 1, 1, 0.95, 1.2)$	19,782	18,133	18,332	17,797	17,968	1.85 ($\checkmark$)	0.91 ($\odot$)	2.91 ($\checkmark$)	1.98 ($\checkmark$)
$(4, 4, 1, 1, 0.95, 1.6)$	17,345	15,769	16,019	15,264	15,522	3.21 ($\checkmark$)	1.57 ($\checkmark$)	4.71 ($\checkmark$)	3.10 ($\checkmark$)
$(4, 8, 4, 0, 0.90, 1.2)$	30,754	29,200	29,445	29,286	29,329	-0.29 ($\odot$)	-0.44 ($\times$)	0.54 ($\checkmark$)	0.39 ($\checkmark$)
$(4, 8, 4, 0, 0.90, 1.6)$	31,744	30,439	30,679	30,324	30,483	0.38 ($\odot$)	-0.15 ($\odot$)	1.16 ($\checkmark$)	0.64 ($\checkmark$)
$(4, 8, 4, 0, 0.95, 1.2)$	28,983	27,446	27,533	27,386	27,445	0.22 ($\odot$)	0.00 ($\odot$)	0.53 ($\checkmark$)	0.32 ($\odot$)
$(4, 8, 4, 0, 0.95, 1.6)$	23,995	22,820	22,901	22,720	22,825	0.44 ($\checkmark$)	-0.02 ($\odot$)	0.79 ($\checkmark$)	0.33 ($\checkmark$)
$(4, 8, 8, 0, 0.90, 1.2)$	26,932	25,559	25,825	25,115	25,182	1.74 ($\checkmark$)	1.47 ($\checkmark$)	2.75 ($\checkmark$)	2.49 ($\checkmark$)
$(4, 8, 8, 0, 0.90, 1.6)$	30,670	28,231	28,527	27,314	27,731	3.25 ($\checkmark$)	1.77 ($\checkmark$)	4.25 ($\checkmark$)	2.79 ($\checkmark$)
$(4, 8, 8, 0, 0.95, 1.2)$	33,136	31,418	31,816	31,134	31,267	0.90 ($\checkmark$)	0.48 ($\odot$)	2.14 ($\checkmark$)	1.73 ($\checkmark$)
$(4, 8, 8, 0, 0.95, 1.6)$	27,926	26,188	26,533	25,456	25,781	2.80 ($\checkmark$)	1.55 ($\checkmark$)	4.06 ($\checkmark$)	2.83 ($\checkmark$)
$(4, 8, 1, 1, 0.90, 1.2)$	26,673	24,777	25,114	23,050	23,446	6.97 ($\checkmark$)	5.37 ($\checkmark$)	8.22 ($\checkmark$)	6.64 ($\checkmark$)
$(4, 8, 1, 1, 0.90, 1.6)$	31,470	29,209	29,498	27,215	28,178	6.82 ($\checkmark$)	3.53 ($\checkmark$)	7.74 ($\checkmark$)	4.47 ($\checkmark$)
$(4, 8, 1, 1, 0.95, 1.2)$	21,959	20,441	20,668	19,411	19,832	5.04 ($\checkmark$)	2.98 ($\checkmark$)	6.09 ($\checkmark$)	4.05 ($\checkmark$)
$(4, 8, 1, 1, 0.95, 1.6)$	26,138	23,656	24,040	22,010	22,825	6.96 ($\checkmark$)	3.51 ($\checkmark$)	8.44 ($\checkmark$)	5.05 ($\checkmark$)

Table 1: Computational results for the test problems with four spokes.

Problem	Profit	Tot. exp. profit obtained by				LSA vs.		PSA vs.	
$(N, \kappa, \theta, \sigma, q, \rho)$	bnd.	LSA	PSA	DLP	FDD	DLP	FDD	DLP	FDD
$(8, 4, 4, 0, 0.90, 1.2)$	22,706	20,286	20,426	20,338	20,463	-0.26 (⊙)	-0.87 (×)	0.43 (⊙)	-0.18 (⊙)
$(8, 4, 4, 0, 0.90, 1.6)$	17,715	15,674	15,823	15,555	15,783	0.76 (⊙)	-0.70 (⊙)	1.70 (✓)	0.25 (⊙)
$(8, 4, 4, 0, 0.95, 1.2)$	21,809	19,579	19,829	19,640	19,795	-0.31 (⊙)	-1.10 (×)	0.95 (✓)	0.17 (⊙)
$(8, 4, 4, 0, 0.95, 1.6)$	15,625	13,804	13,982	13,771	14,025	0.23 (⊙)	-1.61 (×)	1.50 (✓)	-0.31 (⊙)
$(8, 4, 8, 0, 0.90, 1.2)$	19,963	17,346	17,603	16,783	17,223	3.24 (✓)	0.71 (⊙)	4.65 (✓)	2.16 (✓)
$(8, 4, 8, 0, 0.90, 1.6)$	13,868	11,618	11,961	11,408	11,740	1.80 (✓)	-1.05 (⊙)	4.62 (✓)	1.85 (✓)
$(8, 4, 8, 0, 0.95, 1.2)$	21,134	18,470	18,740	18,203	18,589	1.45 (✓)	-0.64 (⊙)	2.86 (✓)	0.81 (✓)
$(8, 4, 8, 0, 0.95, 1.6)$	17,056	14,127	14,484	13,768	14,251	2.54 (✓)	-0.88 (⊙)	4.95 (✓)	1.61 (✓)
$(8, 4, 1, 1, 0.90, 1.2)$	20,019	17,623	17,867	16,785	17,274	4.76 (✓)	1.98 (✓)	6.05 (✓)	3.32 (✓)
$(8, 4, 1, 1, 0.90, 1.6)$	15,712	13,114	13,498	12,488	13,057	4.78 (✓)	0.44 (⊙)	7.48 (✓)	3.26 (✓)
$(8, 4, 1, 1, 0.95, 1.2)$	16,179	13,818	14,096	13,427	13,816	2.83 (✓)	0.01 (⊙)	4.74 (✓)	1.98 (✓)
$(8, 4, 1, 1, 0.95, 1.6)$	19,803	16,860	17,171	16,119	16,792	4.40 (✓)	0.40 (⊙)	6.13 (✓)	2.21 (✓)
$(8, 8, 4, 0, 0.90, 1.2)$	35,075	32,882	32,879	32,742	32,905	0.43 (✓)	-0.07 (⊙)	0.42 (✓)	-0.08 (⊙)
$(8, 8, 4, 0, 0.90, 1.6)$	24,105	22,273	22,336	22,120	22,285	0.69 (✓)	-0.05 (⊙)	0.97 (✓)	0.23 (⊙)
$(8, 8, 4, 0, 0.95, 1.2)$	33,872	31,773	31,808	31,832	31,954	-0.19 (⊙)	-0.57 (×)	-0.08 (⊙)	-0.46 (⊙)
$(8, 8, 4, 0, 0.95, 1.6)$	25,920	24,051	24,069	23,844	24,056	0.86 (✓)	-0.02 (⊙)	0.93 (✓)	0.06 (⊙)
$(8, 8, 8, 0, 0.90, 1.2)$	31,831	28,996	29,046	28,228	28,680	2.65 (✓)	1.09 (✓)	2.82 (✓)	1.26 (✓)
$(8, 8, 8, 0, 0.90, 1.6)$	37,769	33,740	33,955	32,225	33,149	4.49 (✓)	1.75 (✓)	5.09 (✓)	2.37 (✓)
$(8, 8, 8, 0, 0.95, 1.2)$	28,695	26,146	26,245	25,549	25,984	2.29 (✓)	0.62 (⊙)	2.65 (✓)	0.99 (✓)
$(8, 8, 8, 0, 0.95, 1.6)$	32,840	29,691	29,804	28,585	29,292	3.72 (✓)	1.34 (✓)	4.09 (✓)	1.72 (✓)
$(8, 8, 1, 1, 0.90, 1.2)$	29,394	26,367	26,660	23,601	24,840	10.49 (✓)	5.79 (✓)	11.47 (✓)	6.83 (✓)
$(8, 8, 1, 1, 0.90, 1.6)$	28,433	25,515	25,534	22,104	23,688	13.37 (✓)	7.16 (✓)	13.43 (✓)	7.23 (✓)
$(8, 8, 1, 1, 0.95, 1.2)$	26,884	24,131	24,396	22,162	23,190	8.16 (✓)	3.90 (✓)	9.16 (✓)	4.94 (✓)
$(8, 8, 1, 1, 0.95, 1.6)$	28,228	25,454	25,701	22,559	24,123	11.37 (✓)	5.23 (✓)	12.22 (✓)	6.14 (✓)

Table 2: Computational results for the test problems with eight spokes.

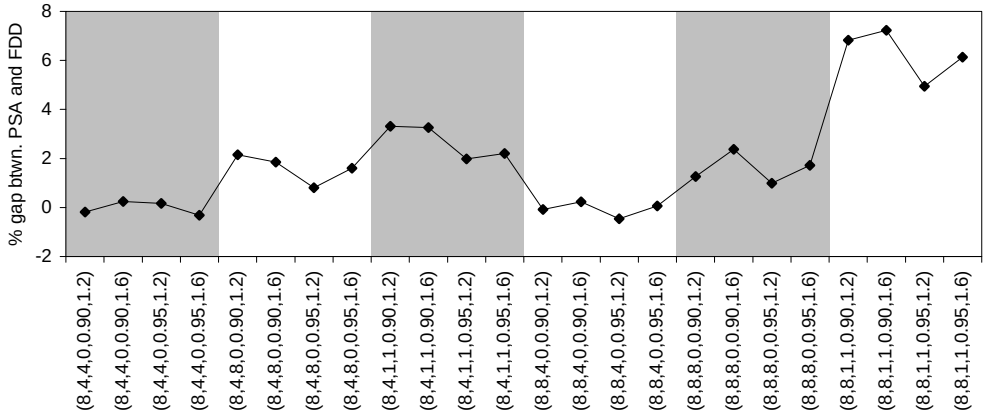

Figure 3: Performance gaps between PSA and FDD for the test problems with eight spokes.

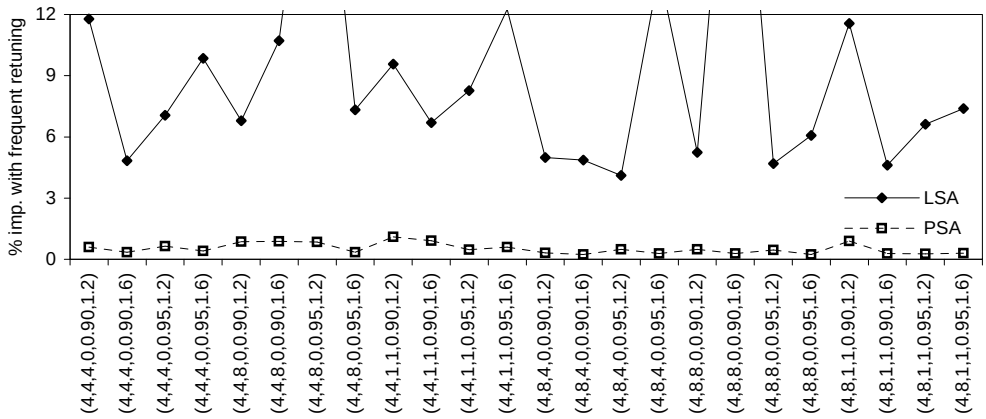

Figure 4: Improvement in the performances of LSA and PSA for the test problems with four spokes when we retune the decision rule more frequently.

	CPU secs. for LSA			CPU secs. for PSA		
No. of spokes	Total	Prob. (1)-(4)	Other	Total	Prob. (1)-(4)	Other
4	13.94	12.41	1.53	118.13	15.08	103.05
8	97.34	95.48	1.86	220.86	106.11	114.75

Table 3: CPU seconds for LSA and PSA required to construct a separable approximation to $V(\cdot)$.

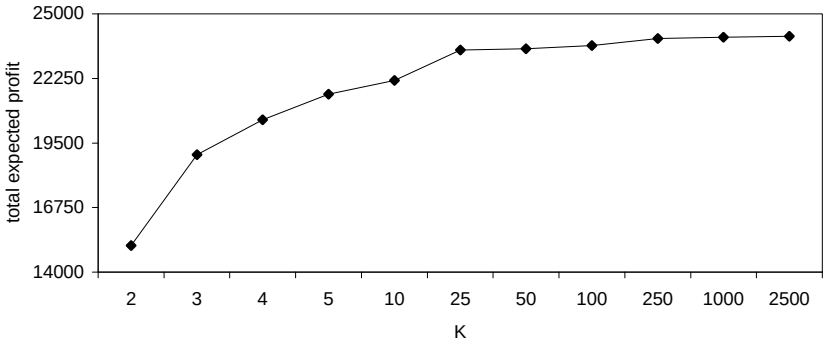

Figure 5: Performance of PSA for test problem $(4, 8, 1, 1, 0.95, 1.6)$ as a function of the iteration counter limit K in the algorithm in Figure 1.