TRANSPORTATION SCIENCE
Vol. 43, No. 3, August 2009, pp. 381–394
ISSN 0041-1655 | EISSN 1526-5447 | 09 | 4303 | 0381

informs

DOI 10.1287/trsc.1090.0262

An Approximate Dynamic Programming Approach to Network Revenue Management with Customer Choice

Dan Zhang
Desautels Faculty of Management, McGill University, Montreal, Quebec H3A 1G5, Canada,
dan.zhang@mcgill.ca

Daniel Adelman
Graduate School of Business, University of Chicago, Chicago, Illinois 60637,
dan.adelman@chicagogsb.edu

We consider a network revenue management problem where customers choose among open fare products according to some prespecified choice model. Starting with a Markov decision process (MDP) formulation, we approximate the value function with an affine function of the state vector. We show that the resulting problem provides a tighter bound for the MDP value than the *choice-based linear program*. We develop a column generation algorithm to solve the problem for a *multinomial logit choice model with disjoint consideration sets* (MNLD). We also derive a bound as a by-product of a decomposition heuristic. Our numerical study shows the policies from our solution approach can significantly outperform heuristics from the choice-based linear program.

Key words: network revenue management; choice behavior; dynamic programming
History: Received: August 2006; revisions received: October 2007 and June 2008; accepted: December 21, 2008.
Published online in *Articles in Advance* June 29, 2009.

While substantial research has been done on methods for solving the network revenue management problem, much less work has been done in solving the version where customers choose among available network products. Usually, when airlines open up a menu of fares for a given set of flights, customers will make substitutions between those available, or purchase nothing. Although incorporating customer choice is important in practice, methodologically is more difficult than the independent demand case, which already suffers from Bellman's "curse of dimensionality."

Recently, Liu and van Ryzin (2008) studied a linear programming formulation, which they call the *choice-based linear program*. In essence, their linear programming formulation is the choice equivalent of the widely used deterministic linear program (DLP) for network revenue management. It approximates the original stochastic problem by replacing stochastic demand with its expected value. They provide a column generation algorithm to solve the problem for the *multinomial logit choice model with disjoint consideration sets* (MNLD). The linear programming formulation is the same as the model proposed in Gallego et al. (2004), where the focus is on analyzing so-called flexible products.

The purpose of this paper is to extend the approximate dynamic programming approach of Adelman (2007) to the customer choice setting, and compare it to Liu and van Ryzin (2008). This is an emerging approach to a wide variety of problems in operations research, for which a lot of active research is ongoing (see, e.g., de Farias and Van Roy 2003). The idea is to formulate the underlying dynamic program as a linear program, and then make an affine functional approximation to the value function to obtain dynamic bid-prices in which marginal resource values change as a function of time. In the independent demand setting, these dynamic bid-prices perform better, in terms of both the bound and policy obtained, than the static bid-prices obtained from the standard linear program. We discover in this paper that this statement remains true when the method is extended to the choice setting. In fact, the gap between the bounds obtained empirically can be as much as 50%.

In addition to the parallel results to Adelman (2007), there are two unique contributions in this paper. First, we provide a way to solve the column generation subproblem. Unlike in Adelman (2007), the column generation subproblem for solving our linear program is a nontrivial nonlinear integer programming problem for general discrete choice models. For the MNLD choice model, the subproblem belongs to the class of integer *generalized fractional programs*. General algorithms for efficient solution of such problems are not currently

available (Schaible and Shi 2003). However, we show that when the resource consumption matrix is a 0-1 matrix, the subproblem is equivalent to a linear mixed integer programming problem, which can be solved effectively.

Second, we provide new theory and results on the decomposition heuristic, which was extended by Liu and van Ryzin (2008) to the choice setting. (See Talluri and van Ryzin 2004b for a discussion of the decomposition heuristic for network revenue management.) In this scheme, the network problem is decomposed into many tractable single-leg problems, which then give single-leg value functions that are both time and capacity dependent. These are then substituted into the right-hand side of Bellman's equation to construct a control policy. We show that, as a by-product of the decomposition heuristic, we obtain a statewise upper bound on the optimal value function, which we call the decomposition bound. When constructed using our dynamic bid-prices, the decomposition bound is tighter than the upper bound obtained from our approximate linear program, which, in turn, is tighter than the bound from the choice-based linear program. This new and improved bound was not known in the revenue management literature. Up to this point, it is customary to use the objective value from the DLP (or choice-based linear program when choice is involved) as the benchmark in numerical studies (see, for example, Liu and van Ryzin 2008). Hence the decomposition bound is a provably better benchmark than has been used previously.

Empirically, we find that the decomposition heuristic with dynamic bid-prices gives superior policy performance as compared with the version using static bid-prices in Liu and van Ryzin (2008). This is significant in practice, because the decomposition heuristic with static bid-prices is generally regarded as one of the best strawmen available, and is widely used in industry. It should be noted that the dynamic bid-prices are generated by solving a much harder linear program problem. Our numerical experiments show that doing so is feasible for reasonably sized problems.

Brief Review of Literature

Our work is closely related to the revenue management and approximate dynamic programming literatures. For a comprehensive review of revenue management literature, see Talluri and van Ryzin (2004b). Many researchers have realized the deficiency of independent demand models, and therefore have studied problems with rich customer choice activities.

Brumelle et al. (1990) consider seat allocations for a two-class single-leg revenue management problem when the demand for the two classes are stochastically dependent because of consumer buy-up. Belobaba and Weatherford (1996) investigate variations to some well known heuristics to account for customer diversion among customer classes. Zhao and Zheng (2001) consider a two-class seat allocation model with passenger diversion. Talluri and van Ryzin (2004a) consider customer choice among fare classes on a single-leg flight. Zhang and Cooper (2005, 2009) consider seat allocation and pricing issues for multiple flights on the same origin and destination. Van Ryzin and Vulcano (2004) introduce a simulation-based optimization approach for network revenue management under a fairly general choice scheme. Gallego et al. (2004) propose a linear program to analyze revenue management for flexible products, which was subsequently adopted by Liu and van Ryzin (2008) to study network revenue management with customer choice. Bront, Mendez-Diaz, and Vulcano (2007) extend the work of Liu and van Ryzin (2008) to allow for overlapping consideration sets in the choice model. Jiang and Miglionico (2006) consider a network revenue management problem with consumer buy-up and explore several solution approaches. Subsequent to our paper, Kunnumkal and Topaloglu (2008) solve the same problem as ours using a Lagrangian approach to approximate dynamic programming (ADP).

The body of literature on ADP is relatively small but growing (see Bertsekas and Tsitsiklis 1996; Powell 2007 for a review). Our approach is most closely related to the approach for dynamic programs. Puterman (1994) gives an excellent review of the area. The functional approximation idea was first considered by Schweitzer and Seidmann (1985).

Organization of the Paper

The rest of the paper is organized as follows. Section 2 considers the affine functional approximation and the resulting, which we call approximate. Section 3 provides a column generation algorithm to solve the approximate for the MNLD choice model. Section 4 introduces heuristics from solutions of the approximate. Section 5 reports numerical results.

1. Problem Formulation

In this section, we provide the basic formulations used in the paper. The Markov decision process (MDP) formulation in §1.1 is essentially the same as the one presented in Liu and van Ryzin (2008). The linear program formulation in §1.2 was first studied by Gallego et al. (2004) and later considered by Liu and van Ryzin (2008).

1.1. MDP Formulation

For ease of exposition, we use airline terminology throughout the paper. Consider a flight network with m legs and the set of capacities $c = (c_1, \ldots, c_m)$, where c_i is the capacity of leg i. There are n products offered,

where a product is a flight itinerary and fare class combination. Let $N = \{1, \dots, n\}$ be the set of products. The fare for product j is f_j. The consumption matrix is an $(m \times n)$-matrix $A \equiv (a_{ij})$. The entry a_{ij} represents the integer amount of resource i required by a class j customer. The ith row A_i is the incidence vector for leg i, and the jth column A^j is the incidence vector for product j. There are τ discrete time periods that are counted forward, so period τ is the last period. To simplify notation, we reserve the symbols i, j, and t for legs, products, and time, respectively.

In each period, there is one customer arrival with probability λ, and no customer arrival with probability $1-\lambda$. When a customer arrives, the firm must decide what products to offer. Let $S \subseteq N$ be the offer set of the firm. Note that $S = \varnothing$ means that no product is offered. Given offer set S, the customer chooses the product $j \in S$ with probability $P_j(S)$, and makes no purchase with probability $P_0(S) = 1 - \sum_{j \in S} P_j(S)$.

The state at the beginning of any period t is an m-vector of unsold seats x. So the state space is $X = \{0, \dots, c_1\} \times \cdots \times \{0, \dots, c_m\}$. Let $v_t(x)$ be the maximum total expected revenue over periods $t, \dots, \tau$ starting at state x at the beginning of period t. The optimality equation is

$$\begin{aligned} v_t(x) &= \max_{S \subseteq N(x)} \left\{ \sum_{j \in S} \lambda P_j(S)(f_j + v_{t+1}(x - A^j)) \right. \\ &\qquad \left. + (\lambda P_0(S) + 1 - \lambda) v_{t+1}(x) \right\} \\ &= \max_{S \subseteq N(x)} \left\{ \sum_{j \in S} \lambda P_j(S)[f_j - (v_{t+1}(x) - v_{t+1}(x - A^j))] \right\} \\ &\qquad + v_{t+1}(x) \quad \forall\, t, x. \end{aligned} \tag{1}$$

The boundary conditions are $v_{\tau+1}(x) = 0$ for all x. In the above, the set

$$N(x) = \{j \in N\colon x \geq A^j\}$$

is the set of products that can be offered when the state is x.

The value function at the initial state c can be computed by the following linear program:

$$\begin{aligned} (\mathbf{D0}) \quad \min_{v(\cdot)} \quad & v_1(c) \\ & v_t(x) \geq \sum_{j \in S} \lambda P_j(S)(f_j + v_{t+1}(x - A^j)) \\ & \qquad + (\lambda P_0(S) + 1 - \lambda) v_{t+1}(x) \\ & \qquad \forall\, t, x, S \subseteq N(x) \end{aligned}$$

with decision variables $v_t(x)$ $\forall\, t, x$. It can be shown by induction that any feasible solution $\hat{v}(\cdot)$ to (**D0**) is an upper bound on $v_t(\cdot)$. See also Adelman (2007) for relevant discussions.

1.2. Choice-Based Linear Programming Formulation

We now review the Liu and van Ryzin (2008) model; see also Gallego et al. (2004). Let S denote the firm's offer set. Customer demand (viewed as continuous quantity) flows in at rate λ. If the set S is offered, product j is sold at rate $\lambda P_j(S)$ (i.e., a proportion $P_j(S)$ of the demand is satisfied by product j). Let $R(S)$ denote the revenue from one unit of customer demand when the set S is offered. Then

$$R(S) = \sum_{j \in S} f_j P_j(S).$$

Note that $R(S)$ is a scalar. Similarly, let $Q_i(S)$ denote the resource consumption rate on flight i, $i = 1, \dots, m$, given that the set S is offered. Let $Q(S) = (Q_1(S), \dots, Q_m(S))^T$. The vector $Q(S)$ satisfies $Q(S) = AP(S)$, where $P(S) = (P_1(S), \dots, P_n(S))^T$ is the vector of purchase probabilities.

Let $h(S)$ be the total time the set S is offered. Because the demand is deterministic as seen by the model and the choice probabilities are time homogeneous, only the total time a set is offered matters; i.e., we do not care about the order in which different offer sets are used. The objective is to find the total time $h(S)$ each set S should be offered to maximize the firm's revenue. The linear program can be written as follows:

$$\begin{aligned} (\mathbf{LP}) \quad z_{\mathrm{LP}} = \max_{h} \quad & \sum_{S \subseteq N} \lambda R(S) h(S) \\ & \sum_{S \subseteq N} \lambda Q(S) h(S) \leq c \qquad (2) \\ & \sum_{S \subseteq N} h(S) = \tau \qquad (3) \\ & h(S) \geq 0, \quad \forall\, S \subseteq N. \end{aligned}$$

Note that $\varnothing \subseteq N$, so that the decision variable $h(\varnothing)$ corresponds to the total time that no products are offered. In the no-choice case, i.e., when $P_j(S) = p_j$ $\forall\, j \in S$ and $P_j(S) = 0$ otherwise, it can be shown that (**LP**) is equivalent to the DLP model in the revenue management literature. In this sense, the (**LP**) model is an extension of (DLP) to the choice case.

2. Functional Approximation

The formulation (**D0**) has a huge number of decision variables and constraints, making its exact solution impractical for moderately sized problem instances. One way to reduce the size of the problem is to approximate $v_t(\cdot)$ by a set of preselected basis functions. One potential approach is to use a linearly parameterized function class

$$v_t(x) \approx \sum_{k=1}^{K} V_{t,k} \phi_k(x), \tag{4}$$

where $\phi_k(x)$ is a prespecified basis function and $V_{t,k}$ is the weight on $\phi_k(x)$. After plugging (4) into (**D0**), the weights can then be determined by solving the resulting linear program. In this paper, we consider an affine functional approximation, which is a special case of (4).

2.1. Formulation

Consider the affine functional approximation

$$v_t(x) \approx \theta_t + \sum_i V_{t,i} x_i, \tag{5}$$

where $V_{t,i}$ estimates the marginal value of a seat on flight i in period t, and θ_t is a constant offset. We assume $\theta_{\tau+1} = 0$ and $V_{\tau+1,i} = 0\ \forall i$.

Plugging (5) into (**D0**), we obtain

$$(\mathbf{D1})\quad \min_{\theta,V} \theta_1 + \sum_i V_{1,i} c_i \tag{6}$$

$$\theta_t - \theta_{t+1} + \sum_i \left(V_{t,i} x_i - V_{t+1,i} \left(x_i - \sum_{j \in S} \lambda P_j(S) a_{ij} \right) \right) \geq \sum_{j \in S} \lambda P_j(S) f_j \quad \forall t, x, S \subseteq N(x). \tag{7}$$

The dual of (**D1**) is

$$(\mathbf{P1})\quad z_{\mathrm{P1}} = \max_{Y} \sum_{t,x,S \subseteq N(x)} \left(\sum_{j \in S} \lambda P_j(S) f_j \right) Y_{t,x,S}$$

$$\sum_{x, S \subseteq N(x)} x_i Y_{t,x,S} = \begin{cases} c_i & \text{if } t = 1 \\ \displaystyle\sum_{x, S \subseteq N(x)} \left(x_i - \sum_{j \in S} \lambda P_j(S) a_{ij} \right) Y_{t-1,x,S} & \forall i, t \\ & \forall t = 2, \ldots, \tau \end{cases} \tag{8}$$

$$\sum_{x, S \subseteq N(x)} Y_{t,x,S} = \begin{cases} 1 & \text{if } t = 1 \\ \displaystyle\sum_{x, S \subseteq N(x)} Y_{t-1,x,S} & \\ & \forall t = 2, \ldots, \tau \end{cases} \tag{9}$$

$$Y \geq 0.$$

The constraint (9) can be replaced by

$$\sum_{x, S \subseteq N(x)} Y_{t,x,S} = 1 \quad \forall t. \tag{10}$$

Therefore we can interpret the decision variables $Y_{t,x,S}$ as approximated state-action probabilities; i.e., $Y_{t,x,S}$ is the probability that the state is x and the set S is offered at time t. The constraint (8) is a flow balance constraint, which says that the mass of each resource i flowing into time t must equal that flowing out.

2.2. Relationship to (LP)

To derive (**LP**) from (**P1**), define

$$h(S) \equiv \sum_{t,x} Y_{t,x,S} \quad \forall S \subseteq N. \tag{11}$$

Note that $Y_{t,x,S} = 0\ \forall S \nsubseteq N(x)$. Since $Y_{t,x,S}$ can be interpreted as the probability that the state is x and the set S is offered in period t, the right-hand side of (11) can be interpreted as the total time the set S is offered throughout the time horizon, which is exactly the interpretation of decision variable $h(S)$ in (**LP**). Then, the objective function in (**P1**) can be written as

$$\sum_{S \subseteq N} \left(\sum_{j \in S} \lambda P_j(S) f_j \right) h(S) = \sum_{S \subseteq N} \lambda R(S) h(S).$$

Summing (8) over t, we obtain

$$\sum_{t,x,S \subseteq N} x_i Y_{t,x,S} = c_i + \sum_{t=2}^{\tau} \sum_{x, S \subseteq N} \left(x_i - \sum_{j \in S} \lambda P_j(S) a_{ij} \right) Y_{t-1,x,S} \quad \forall i.$$

Canceling terms and rearranging, we obtain

$$c_i = \sum_{t=1}^{\tau-1} \sum_{x, S \subseteq N} \sum_{j \in S} \lambda P_j(S) a_{ij} Y_{t,x,S} + \sum_{x, S \subseteq N} x_i Y_{\tau,x,S}. \tag{12}$$

If $Y_{t,x,S} > 0$, we must have $x_i \geq a_{ij}\ \forall i,\ j \in S$; so $x_i \geq \sum_{j \in S} \lambda P_j(S) a_{ij}$. It then follows that

$$\sum_{x, S \subseteq N} x_i Y_{\tau,x,S} \geq \sum_{x, S \subseteq N} \sum_{j \in S} \lambda P_j(S) a_{ij} Y_{\tau,x,S}.$$

Hence (12) implies that

$$c_i \geq \sum_{t,x,S} \sum_{j \in S} \lambda P_j(S) a_{ij} Y_{t,x,S} = \sum_{S \subseteq N} \lambda Q_i(S) h(S).$$

Summing (9) over t, we get

$$\sum_{S \subseteq N} h(S) = \tau.$$

The arguments above show that $z_{\mathrm{LP}} \geq z_{\mathrm{P1}}$. Furthermore, similar to Proposition 1 in Adelman (2007), we can show that any feasible solution to (**D0**) gives an upper bound to the optimal value from the dynamic program. Since (**D1**)–(**P1**) gives a feasible solution to (**D0**), it follows that $z_{\mathrm{P1}} \geq v_1(c)$. We summarize the results in the following proposition.

Proposition 1. *Any feasible solution to* (**P1**) *yields a feasible solution to* (**LP**) *having the same objective value. Hence* $z_{\mathrm{LP}} \geq z_{\mathrm{P1}} \geq v_1(c)$.

Theorem 1 in Adelman (2007) shows that a similar relation holds when there is no customer choice among products for a classic revenue management problem. Liu and van Ryzin (2008) show that the bound z_{LP} is asymptotically optimal, i.e., converges to $v_1(c)$, as demand, capacity, and time horizon scale linearly. It follows from Proposition 1 that the bound z_{P1} is also asymptotically optimal.

We note that solutions to (**P1**) overcome some of the difficulties encountered when trying to use solutions

of (**LP**) in the dynamic setting of (1); namely, a solution to (**LP**) only gives the time duration for which each offer set should be used, but does not specify an order in which the sets should be used. The solution to (**P1**), however, allows different sets to be used at different times. In fact, from constraint (9), the decision variable $Y_{t,x,S}$ may be interpreted as approximate state-action probabilities.

Next, we show that there exists an optimal solution (V^*, θ^*) to (**D1**) that is time monotonic. Given a feasible solution Y to (**P1**), define the first-time resource i is used by

$$t_i^* = \arg\min\left\{ t \in \{1,\dots,\tau\}\colon \exists x, S \text{ with } Y_{t,x,S} > 0, \sum_{j\in S} P_j(S)a_{ij} > 0 \right\}. \quad (13)$$

We have the following monotonicity result.

THEOREM 1. *Assume $c_i > 0\ \forall i$. There exists an optimal solution (θ^*, V^*) of (**D1**) and a set of indices $\{\tilde{t}_i^*\colon \forall i\}$ such that*

$$\theta_t^* \geq \theta_{t+1}^* \quad \forall t \quad (14)$$

$$V_{t,i}^* = V_{t+1,i}^* \quad \forall i,\ t = 1,\dots,\tilde{t}_i^* - 1 \quad (15)$$

$$V_{t,i}^* \geq V_{t+1,i}^* \quad \forall i,\ t = \tilde{t}_i^*,\dots,\tau \quad (16)$$

$$V^*, \theta^* \geq 0. \quad (17)$$

PROOF. The proof follows Adelman (2007) (see the online appendix available at http://trsc.pubs.informs.org/ecompanion.html). □

In the proof, $\tilde{t}_i^* = t_i^*$ when t_i^* exists; otherwise, $\tilde{t}_i^* = \tau$. Conditions (15)–(16) show that V^* is nonincreasing over time. Since $V_{t,i}^*$ can be interpreted as the approximate marginal value of resource i at time t, this result is intuitively appealing, because as time moves forward, we have less opportunities to sell.

3. Column Generation Algorithm

3.1. General Case

The program (**P1**) has a large number of variables but relatively few constraints, so it can be potentially solved via column generation. In this section, we develop such an algorithm.

First, it is easy to find an initial feasible solution to start the column generation algorithm. There is one corresponding to closing all products in each period; i.e., let

$$Y_{t,x,S} = \begin{cases} 1 & \text{if } x = c, S = \varnothing \\ 0 & \text{otherwise} \end{cases} \quad \forall t, x, S.$$

At a given iteration, suppose the dual solution is (V, θ). Let $\pi_{t,x,S}$ be the reduced profit of the column corresponding to x, S in period t. The maximum reduced profit can be computed by solving

$$\begin{aligned} &\max_{t,x,S\subseteq N(x)} \pi_{t,x,S} \\ &= \max_{t,x,S\subseteq N(x)} \sum_{j\in S} \lambda P_j(S) f_j \\ &\quad - \sum_i \left(V_{t,i}x_i - V_{t+1,i}\left(x_i - \sum_{j\in S} \lambda P_j(S) a_{ij} \right) \right) - \theta_t + \theta_{t+1} \\ &= \max_{t,x,S\subseteq N(x)} \sum_{j\in S} \lambda P_j(S) \left[f_j - \sum_i a_{ij} V_{t+1,i} \right] \\ &\quad - \sum_i (V_{t,i} - V_{t+1,i}) x_i - \theta_t + \theta_{t+1}. \end{aligned}$$

If the objective value is greater than 0, then we add the column corresponding to the optimal solution to the existing set of columns for (**P1**); otherwise, optimality is attained. For fixed $t \geq 1$, we need to solve the following optimization problem:

$$\begin{aligned} (\mathbf{S0}) \quad \max_{x,S} \ & \sum_{j\in S} \lambda P_j(S) \left[f_j - \sum_i a_{ij} V_{t+1,i} \right] \\ & - \sum_i (V_{t,i} - V_{t+1,i}) x_i - \theta_t + \theta_{t+1} \\ & x_i \geq a_{ij} \quad \forall i, j \in S \quad (18) \\ & x_i \in \{0,\dots,c_i\} \quad \forall i. \end{aligned}$$

The effectiveness of a column generation algorithm hinges on efficient solution of the column generation subproblems. For general choice probability P, (**S0**) is potentially a nonlinear integer constrained optimization problem, which is quite difficult to solve. Because of constraint (18), the optimization problem is not separable in x and S. Even if x is fixed, the optimization on S is a combinatorial optimization, which is potentially difficult by itself. For the MNLD, Liu and van Ryzin (2008) develop an efficient ranking procedure to solve the optimization on S; see also Gallego et al. (2004). However, such a procedure cannot be extended to solve (**S0**) because it also optimizes over x.

3.2. MNLD

In MNLD, each customer is interested in a subset of the products. Let $L = \{1,\dots,\bar{l}\}$ be the set of customer segments. We assume that customer segment $l \in L$ has consideration set $N_l \subseteq N$. We assume $\forall l_1 \neq l_2$, $N_{l_1} \cap N_{l_2} = \varnothing$; i.e., different customer segments have disjoint consideration sets. Within each segment, customer choice follows a multinomial logit (MNL) model. Under MNL, the choice probability can be defined by a choice vector. To completely specify the choice probability, we need to specify the preference value v_{lj} for $l \in L$, $j \in N_l$, and the no-purchase value v_{l0} for $l \in L$. In general, a choice set S can also be

represented by an availability vector. We use a binary vector u_l to denote the product availability for segment l such that $u_{lk} = 1\{k \in S\}$. For convenience, we use vector and set notations interchangeably. Then, the probability a segment l customer purchases product $j \in N_l$ is

$$\widetilde{P}_{lj}(u_l) = \frac{u_{lj} v_{lj}}{\sum_{j \in N_l} u_{lj} v_{lj} + v_{l0}}.$$

To accommodate the MNLD choice model in the framework of §1.1, we can assume an arriving customer first chooses which segment he belongs to, and then chooses products within the given segment. In particular, we assume the probability an arriving customer belongs to segment l is λ_l/λ, where $\sum_l \lambda_l = \lambda$. It then follows that for $j \in N_l$,

$$P_j(S) = \frac{\lambda_l \widetilde{P}_{lj}(u_l(S))}{\lambda},$$

where $u_{lk}(S) = 1\{k \in S\}$ for all $k \in N_l$.

3.3. Column Generation for MNLD

In the following, we show that for MNLD, (**S0**) can be reduced to a linear integer program, the solution of which is relatively easy. Let $\lambda_l \ \forall\, l \in L$ be such that $\sum_{l \in L} \lambda_l = \lambda$, where λ_l is the probability of a customer arrival in segment l for a given period. Plugging in the MNLD choice probabilities, (**S0**) becomes

$$\begin{aligned}
(\textbf{S-MNLD}) \quad \max_{x,u} \ & \sum_{l \in L} \left(\frac{\lambda_l \sum_{j \in N_l} u_{lj} v_{lj} [f_j - \sum_i a_{ij} V_{t+1,i}]}{\sum_{j \in N_l} u_{lj} v_{lj} + v_{l0}} \right) \\
& - \sum_i (V_{t,i} - V_{t+1,i}) x_i - \theta_t + \theta_{t+1} \\
& x_i \geq a_{ij} u_{lj} \quad \forall i, j \in N_l, l \in L \\
& x_i \in \{0, \ldots, c_i\} \quad \forall i \\
& u_l \in \{0,1\}^{|N_l|} \quad \forall l \in L.
\end{aligned}$$

If the integrality constraints are relaxed, the optimization problem above belongs to a class of optimization problems called *generalized linear-fractional programs* (see, e.g., Boyd and Vandenberghe 2004). A general approach to solve such a problem efficiently is not available. See Schaible and Shi (2003) for a recent review of literature on this subject.

In the following, we show that (**S-MNLD**) can be reduced to an equivalent linear integer program by exploiting the structure of the problem. In particular, we use the fact that the consideration sets are disjoint and that u_{lj} is binary.

Let

$$z_{lj} = \frac{u_{lj}}{\sum_{j \in N_l} v_{lj} u_{lj} + v_{l0}} \quad \forall\, j \in N_l,\ l \in L \tag{19}$$

$$\alpha_l = \frac{1}{\sum_{j \in N_l} v_{lj} u_{lj} + v_{l0}} \quad \forall\, l \in L. \tag{20}$$

It follows from the definition of z_{lj} and α_l that

$$\begin{aligned}
& \sum_{j \in N_l} v_{lj} z_{lj} + v_{l0} \alpha_l = 1 \quad \forall\, l \in L \\
& \alpha_l \geq 0 \quad \forall\, l.
\end{aligned} \tag{21}$$

Plugging (19)–(21) into (**S-MNLD**), we obtain

$$\begin{aligned}
(\textbf{S-MNLD1}) \quad \max_{x,z,\alpha} \ & \sum_{l \in L} \sum_{j \in N_l} \lambda_l v_{lj} \left[f_j - \sum_i a_{ij} V_{t+1,i} \right] z_{lj} \\
& - \sum_i (V_{t,i} - V_{t+1,i}) x_i - \theta_t + \theta_{t+1} \\
& x_i \geq \frac{a_{ij} z_{lj}}{\alpha_l} \quad \forall\, i, j \in N_l,\ l \in L \qquad (22) \\
& x_i \in \{0, \ldots, c_i\} \quad \forall\, i \\
& z_{lj} \in \{0, \alpha_l\} \quad \forall\, j \in N_l, l \in L \qquad (23) \\
& \sum_{j \in N_l} v_{lj} z_{lj} + v_{l0} \alpha_l = 1 \quad \forall\, l \in L \qquad (24) \\
& \alpha_l \geq 0 \quad \forall\, l. \qquad (25)
\end{aligned}$$

Lemma 1. (**S-MNLD**) *is equivalent to* (**S-MNLD1**); *i.e., both optimization problems have the same optimal objective value and an optimal solution to one can be obtained from an optimal solution of the other.*

Proof. Since (**S-MNLD1**) is obtained from (**S-MNLD**) through change of variables, it can be shown that an optimal solution to (**S-MNLD**) is a solution to (**S-MNLD1**) and both optimization problems have the same objective value at the solution.

Suppose $(\hat{x}, \hat{z}, \hat{\alpha})$ is an optimal solution to (**S-MNLD1**). From (23)–(24), we must have $\hat{\alpha}_l > 0$ for all l. Let $\hat{u}_{lj} = \hat{z}_{lj}/\hat{\alpha}_l \ \forall\, l, j$. Then $(\hat{x}, \hat{u})$ clearly satisfies the constraints in (**S-MNLD**). Furthermore, $\forall\, l, j$,

$$\begin{aligned}
\frac{\hat{u}_{lj}}{\sum_{j \in N_l} \hat{u}_{lj} v_{lj} + v_{l0}} &= \frac{\hat{u}_{lj} \hat{\alpha}_l}{\sum_{j \in N_l} \hat{u}_{lj} v_{lj} \hat{\alpha}_l + v_{l0} \hat{\alpha}_l} \\
&= \frac{\hat{z}_{lj}}{\sum_{j \in N_l} \hat{z}_{lj} v_{lj} + v_{l0} \hat{\alpha}_l} = \hat{z}_{lj},
\end{aligned}$$

where the last equation follows by using (24). It then follows that the two optimization problems have the same objective value at the given solution. This completes the proof. □

From Lemma 1, it suffices to solve (**S-MNLD1**). There is potentially one difficulty in solving the problem (**S-MNLD1**): the constraint (22) is nonlinear. However, we show that if $v_{l0} > 0 \ \forall\, l \in L$ and $a_{ij} \in \{0,1\} \ \forall\, i, j$, it can be replaced by an equivalent linear constraint.

Theorem 2. *Suppose* $v_{l0} > 0$ *for all* $l \in L$ *and* $a_{ij} \in \{0,1\}$. *We only need to solve the following linear integer program to find the maximum reduced profit for each* $t \geq 1$:

$$\begin{aligned}
(\textbf{S-MNLD2}) \quad \max_{x,z,\alpha} \ & \sum_{l \in L} \sum_{j \in N_l} \lambda_l v_{lj} \left[f_j - \sum_i a_{ij} V_{t+1,i} \right] z_{lj} \\
& - \sum_i (V_{t,i} - V_{t+1,i}) x_i - \theta_t + \theta_{t+1}
\end{aligned}$$

$$x_i \ge a_{ij} v_{l0} z_{lj} \quad \forall i, j \in N_l,\ l \in L \tag{26}$$

$$x_i \in \{0, \dots, c_i\} \quad \forall i$$

$$z_{lj} \in \{0, \alpha_l\} \quad \forall j \in N_l,\ l \in L \tag{27}$$

$$\sum_{j \in N_l} v_{lj} z_{lj} + v_{l0} \alpha_l = 1 \quad \forall l \in L$$

$$\alpha_l \ge 0 \quad \forall l.$$

Proof. We will show that the feasible region does not change before and after replacing the constraint (22) by (26). From (24),

$$\frac{1}{\sum_{j \in N_l} v_{lj} + v_{l0}} \le \alpha_l \le \frac{1}{v_{l0}} \quad \forall l \in L. \tag{28}$$

It then follows from (22) that

$$x_i \ge \frac{a_{ij} z_{lj}}{\alpha_l} \ge a_{ij} v_{l0} z_{lj} \quad \forall i, j \in N_l,\ l \in L.$$

Hence, all feasible solutions to (**S-MNLD1**) satisfy (26). Next, we show that all feasible solutions to (**S-MNLD2**) satisfy (22). Let $(\hat{x}, \hat{z}, \hat{\alpha})$ be a feasible solution to (**S-MNLD2**). For fixed i, if $\hat{x}_i = 0$, then $\hat{z}_{lj} = 0$ from (26) for all j such that $a_{ij} > 0$; if $\hat{x}_i \ge 1$, then $\hat{x}_i \ge 1 \ge a_{ij} \hat{z}_{lj} / \hat{\alpha}_l$ since $\hat{z}_{lj} / \hat{\alpha}_l \in \{0, 1\}$ and $a_{ij} \in \{0, 1\}$; note that from (28) $\hat{\alpha}_l > 0$ for all l. This completes the proof. □

Subsequently, we will refer to (**S-MNLD2**) as the column generation subproblem. Note that some existing linear programming solvers, such as CPLEX, provide ways to handle (27) directly. In our numerical study, we assume A is binary. However, our theoretical results that follow do not depend on this assumption.

4. Policies from Functional Approximation

In this section, we show how to use the value function approximations to construct control policies.

4.1. Direct Use of Dynamic Bid-Prices

Let (V^*, θ^*) be the optimal solution for (**D1**). Using the approximation

$$v_t(x) - v_t(x - A^j) \approx \sum_i a_{ij} V^*_{t,i},$$

a control policy in period t and state x can be computed by solving for each l

$$\max_{u_{lj} \in \{0,\, 1\{x \ge A^j\}\} \forall\, j \in N_l} \frac{\sum_{j \in N_l} v_{lj} u_{lj} [f_j - \sum_i a_{ij} V^*_{t+1,i}]}{\sum_{j \in N_l} v_{lj} u_{lj} + v_{l0}}. \tag{29}$$

The constraint $u_{lj} \in \{0, 1_{\{x \ge A^j\}}\}$ in (29) incorporates the constraint on capacity. The heuristic is motivated by the dynamic programming recursion (1). The maximization in (29) can be solved efficiently using a simple ranking procedure (see Liu and van Ryzin 2008; Gallego et al. 2004). The resulting policy is called approximate dynamic programming (ADP) in §5.

4.2. Decomposition Based on Solutions to (D1)

We can also use an optimal solution (V^*, θ^*) to (**D1**) in a decomposition approach. Given i, we use the following approximation:

$$v_t(x) \approx v^i_t(x_i) + \sum_{k \ne i} V^*_{t,k} x_k \quad \forall t, x, \tag{30}$$

where $v^i_t(\cdot)$ is the value function for the leg-i problem.

Substituting (30) into (**D0**), we obtain

$$\begin{aligned}(\mathbf{LP}_i) \quad \min_{v^i_t(\cdot)} \ & v^i_1(c_i) + \sum V^*_{1,k} c_k \\ & v^i_t(x_i) + \sum_{k \ne i} V^*_{t,k} x_k \ge \sum_{j \in S} \lambda P_j(S) \\ & \quad \cdot \left[f_j + v^i_{t+1}(x_i - a_{ij}) + \sum_{k \ne i} (x_k - a_{kj}) V^*_{t+1,k} \right] \\ & \quad + (\lambda P_0(S) + 1 - \lambda) \left[v^i_{t+1}(x_i) + \sum_{k \ne i} x_k V^*_{t+1,k} \right] \\ & \quad \forall t, x, S \subseteq N(x). \end{aligned} \tag{31}$$

Proposition 2 relates the objective value from $(\mathbf{LP}_i)$ to z_{LP}, z_{P1}, and the MDP value.

Proposition 2. *For each i, let $\{v^i_t(\cdot)\}$ and $\{v^{*i}_t(\cdot)\}$ be a feasible solution and an optimal solution to* $(\mathbf{LP}_i)$, *respectively. We have*

(i) $\min_i \{v^i_t(x_i) + \sum_{k \ne i} V^*_{t,k} x_k\} \ge v_t(x)\ \forall t, x$;

(ii) $z_{\mathrm{LP}} \ge z_{\mathrm{P1}} \ge \max_i \{v^{*i}_1(c_i) + \sum_{k \ne i} V^*_{1,k} c_k\} \ge \min_i \{v^{*i}_1(c_i) + \sum_{k \ne i} V^*_{1,k} c_k\} \ge v_1(c)$.

Proof. (i) We only need to prove $v^i_t(x_i) + \sum_{k \ne i} V^*_{t,k} x_k \ge v_t(x)$ for all i, t, x. We prove the result by induction. For $t = \tau$, we note from (31) that

$$v^i_\tau(x_i) + \sum_{k \ne i} V^*_{\tau,k} x_k \ge \sum_{j \in S} \lambda P_j(S) f_j \quad \forall x, S \subseteq N(x).$$

It follows that

$$v^i_\tau(x_i) + \sum_{k \ne i} V^*_{\tau,k} x_k \ge \max_{S \subseteq N(x)} \sum_{j \in S} \lambda P_j(S) f_j = v_\tau(x).$$

This shows that the result holds for $t = \tau$. Next, assume the result holds for $t + 1$. Then, by (31) and the inductive assumption for all x, S, we have

$$\begin{aligned} v^i_t(x_i) + \sum_{k \ne i} V^*_{t,k} x_k \ge & \sum_{j \in S} \lambda P_j(S)(f_j + v_{t+1}(x - A^j)) \\ & + (\lambda P_0(S) + 1 - \lambda) v_{t+1}(x). \end{aligned}$$

From the optimality equation for period t, we obtain

$$v^i_t(x_i) + \sum_{k \ne i} V^*_{t,k} x_k \ge v_t(x).$$

(ii) The first inequality is established in Proposition 1, and the third inequality is immediate. The last inequality is implied by part (i). So, we will only need to show the second inequality. It suffices to show $z_{\mathrm{P1}} \geq v_1^i(c_i) + \sum_{k\neq i} V^*_{1,k} c_k \ \forall i$. Let (V^*, θ^*) be an optimal solution to (**D1**). From the feasibility of (V^*, θ^*) to (**D1**), $v_t^i(x_i) = \theta^*_t + V^*_{t,i} x_i \ \forall t, x_i$ is a feasible solution to ($\mathbf{LP}_i$). Because ($\mathbf{LP}_i$) is a minimization problem, this establishes the inequality. □

Part (ii) in Proposition 2 shows that $\min_i\{v_1^{*i}(c_i) + \sum_{k\neq i} V^*_{t,1} c_k\}$ is a tighter bound of $v_1(c)$ than z_{P1}. Such a bound is useful because it provides a better benchmark in numerical studies.

($\mathbf{LP}_i$) has the same number of constraints as (**D1**) but more decision variables. So solving ($\mathbf{LP}_i$) is potentially even harder than solving (**D1**), although it is possible to solve the program via column generation. Instead, we consider the following dynamic program:

$$\begin{aligned}\hat{v}_t^i(x_i) &= \max_{x_{-i},\, S\subseteq N(x)} \left\{ \sum_{j\in S} \lambda P(S) \left[f_j - \hat{v}^i_{t+1}(x_i) + \hat{v}^i_{t+1}(x_i - a_{ij}) \right.\right. \\ &\qquad \left.\left. - \sum_{k\neq i} V^*_{t+1,k} a_{kj} \right] - \sum_{k\neq i} (V^*_{t,k} - V^*_{t+1,k}) x_k \right\} + \hat{v}^i_{t+1}(x_i) \quad (32)\end{aligned}$$

with boundary conditions $v^i_{\tau+1}(x_i) = 0 \ \forall x_i, i$. In the above, x_{-i} denotes the vector x without the i-th component. We can use the backward recursion algorithm for dynamic programs to solve for $\hat{v}_t^i(\cdot)$. In each iteration, the maximization problem in (32) is almost the same as the column generation subproblem (**S0**). The only difference is that here in each iteration, the value of x_i is fixed. As discussed in §3, for MNLD choice model, we only need to solve a linear mixed-integer program. Proposition 3 shows that $\hat{v}_1^i(c_i)$ from (32) is equal to $v_1^i(c_i)$ from ($\mathbf{LP}_i$).

Proposition 3. $v_1^{*i}(c_i) = \hat{v}_1^i(c_i) \ \forall i$.

Proof. We first show that $v_t^{*i}(x_i) \geq \hat{v}_t^i(x_i)$. The proof is by induction and is similar to the proof of part (i) in Proposition 2. It then follows that $v_1^{*i}(c_i) \geq \hat{v}_1^i(c_i)$. Furthermore, by (32), $\hat{v}_t^i(\cdot)$ satisfies the constraints in ($\mathbf{LP}_i$), and is therefore feasible for ($\mathbf{LP}_i$). Hence $\hat{v}_1^i(c_i) \geq v_1^{*i}(c_i)$. This completes the proof. □

After $\hat{v}_t^i(\cdot)$ is determined for each i, we can use the approximation

$$v_t(x) - v_t(x - A^j) \approx \sum_{i=1}^m (\hat{v}_t^i(x_i) - \hat{v}_t^i(x_i - a_{ij}))$$

to compute a heuristic policy for (1) by solving

$$\max_{S\subseteq N(x)} \left\{ \sum_{j\in S} \lambda P_j(S) \left[f_j - \sum_{i=1}^m (\hat{v}_t^i(x_i) - \hat{v}_t^i(x_i - a_{ij})) \right] \right\}.$$

This optimization again can be done effectively for MNLD choice model. The resulting policy is called ADPD in §5.

4.3. Linear Programming Decomposition

Let π^* be the vector of dual values of the resource constraints in (**LP**). The vector π^* can also be used in a decomposition approach similar to the one in §4.2. Given i, we use the following approximation:

$$v_t(x) \approx \tilde{v}_t^i(x_i) + \sum_{k\neq i} \pi^* x_k \quad \forall t, x, \qquad (33)$$

where $\tilde{v}_t^i(\cdot)$ is the value function for the leg-i problem that is obtained when (33) is substituted into (**D0**); for more details, see Liu and van Ryzin (2008). This policy is called LPD in §5. We finish this section by showing the following result.

Proposition 4. $\tilde{v}_t^i(x_i) + \sum_{k\neq i} \pi^*_k x_k \geq v_t(x) \ \forall i$.

Proof. Note that $\tilde{v}_t^i(\cdot)$ is the solution to the linear program (**D0**) with $v_t(x)$ replaced by $\tilde{v}_t^i(x_i) + \sum_{k\neq i} \pi^*_k x_k$. The proof is similar to the proof for part (i) in Proposition 2 and is omitted. □

Proposition 4 shows that linear programming decomposition produces a bound for the value function of the problem. Although decomposition heuristic is studied in much of existing literature (see, e.g., Talluri and van Ryzin 2004b), the bound in Proposition 4 is new.

5. Numerical Experiments

We conducted numerical experiments to study the performance of the proposed solution approach for (**P1**), the relationships among the different bounds, and the performance of the proposed policy approaches.

In the no-choice (independent demand) setting, Adelman (2007) shows that the performance of policies based on dynamic bid-prices depends heavily on the (nominal) load factor. Because demand depends on the offer set, the notion of load factor is not immediately clear for network choice problems. Given choice probability P, let

$$S^* \in \arg\max_{S\subseteq N} \sum_{j\in S} P_j(S) f_j.$$

Note that S^* is a revenue-maximizing set of open products when there is ample capacity of each resource. We call

$$\rho = \frac{\lambda \sum_{t=1}^{\tau} \sum_{j\in S^*} \sum_{i=1}^m a_{ij} P_j(S^*)}{\sum_{i=1}^m c_i} \qquad (34)$$

the (nominal) load factor. In the no-choice setting $S^* = N$ and the load factor defined in (34) is the same as the one commonly used in the literature (see, e.g., Adelman 2007).

5.1. Computational and Bound Performance

We study the performance of the proposed algorithm to solve (**P1**) on randomly generated hub-and-spoke

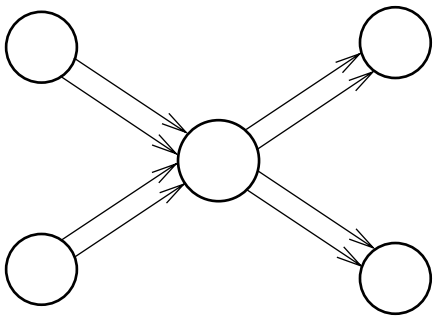

Figure 1 Network Structure for HS1 Instances with $K = 4$

network instances. We implemented the solution methods using C++ and CPLEX on an Intel Xeon 3.6 GZ workstation. In the column generation algorithm for (**P1**), the objective value of the restricted problem plus the sum over time of the maximum reduced profit in each period serves as an upper bound for the optimal objective value (see Adelman 2007). In this subsection, we use a 5% optimality tolerance; that is the column generation procedure terminates when the sum of the maximum reduced profit in each period over time is within 5% of the objective value of the restricted problem. In our implementation, we added the monotonicity constraints suggested by Theorem 1, which often speed up the solution considerably. The same phenomenon was observed in Adelman (2007).

We consider a set of hub-and-spoke network instances with $K \in \{2, 4, 8, 12\}$ nonhub locations. This set of instances is called HS1, subsequently. Figure 1 shows the network structure when $K = 4$, where each arc in the network represents a flight leg. Half of the nonhub locations each have two parallel flights to the hub, and the other half each have two parallel flights from the hub. There are $2K$ flights in total. Table 1 shows key statistics for HS1 instances. For $K = 12$, there are 48 customer segments and 336 products. We assume that all products with the same origin and destination belong to the same segment, so the number of segments is the same as the number of origin-destination (O-D) pairs. Two classes, a high-fare class and low-fare class, are offered for each possible itinerary. The high-class fares of local itineraries to the hub are drawn from the Poisson distribution with mean 30; the corresponding low fares are drawn from the Poisson distribution with mean 10. All other high fares are drawn from the Poisson distribution with mean 300, and all other low fares are drawn from the Poisson distribution with mean 100. Choice parameters for high- and low-fare products are drawn from the Poisson distribution with mean 50 and 200, respectively. The no-purchase weight for each segment is drawn from the Poisson distribution with mean 10. To randomly generate the arrival rate for each customer segment, we draw E_l from the Poisson distribution with mean 20 for each l and set $\lambda_l = 0.9E_l / \sum_{l'} E_{l'}$. Note that the total arrival rate in each period is 0.9. We generated problem instances with $\tau \in \{50, 100, 200, 400, 800\}$.

Table 1 Statistics of Hub-and-Spoke Test Instances HS1 and HS2

Case	HS1	HS2
No. of nonhub locations	$K \in \{2, 4, 8, 12\}$	$K \in \{2, 4, 8, 16\}$
No. of O-D pairs (segments)	$K + K^2/4$	$2K + K(K-1)$
No. of Resources	$2K$	$2K$
No. of Itineraries	$2K + K^2$	$2K + K(K-1)$
No. of Products	$4K + 2K^2$	$4K + 2K(K-1)$

Table 2 shows the load factor and capacity per leg for the problem instances. The capacity across legs is taken to be the same for each instance. The CPU seconds to solve (**P1**) is also reported. The biggest instance with 12 nonhub locations and 800 periods takes about 24 minutes to solve, which is practical in real application. Also the CPU seconds increase as τ and K increase. This observation is not surprising, because the size of (**P1**) increases with both τ and K.

Table 3 reports the CPU seconds for HS1 instances except for half the load factor. The load was cut in half by cutting the arrival rate in each period in half. The table shows that it is 3.63 times faster on average to solve these problems with a range of 1.18–7.88. This speedup can be explained by the fact that as the load factor decreases, the combination of a few offer

Table 2 Load Factor, Capacity, and CPU Seconds for HS1

	Number of nonhub locations, resources, and products											
	2, 4, 16			4, 8, 48			8, 16, 160			12, 24, 336		
τ	Load factor	Capacity per leg	CPU seconds	Load factor	Capacity per leg	CPU seconds	Load factor	Capacity per leg	CPU seconds	Load factor	Capacity per leg	CPU seconds
50	1.37	10	0.33	1.31	6	1.98	1.42	3	18.74	1.54	2	332.77
100	1.35	21	0.85	1.39	11	2.66	1.47	6	35.15	1.53	4	150.42
200	1.36	38	1.24	1.39	22	5.87	1.46	12	55.04	1.37	9	270.63
400	1.35	79	3.59	1.42	45	12.31	1.43	25	164.98	1.43	17	557.12
800	1.40	156	6.73	1.38	88	24.93	1.42	49	239.12	1.43	34	1,432.72

Table 3 CPU Seconds to Solve (P1) for HS1 with Half the Load Factor

	Locations			
τ	2	4	8	12
50	0.24	0.33	6.15	104.42
100	0.42	0.99	5.09	29.31
200	0.86	2.14	22.16	34.36
400	1.18	3.83	42.19	100.14
800	5.72	7.46	50.35	525.17

sets provides near-optimal performance. In particular, when there is enough capacity to accept all customer requests, the policy that only offers the revenue maximization set S^* is optimal. As a result, fewer columns need to be added when solving (**P1**).

Table 4 reports the approximate relative difference $z_{\mathrm{LP}}/z_{\mathrm{P1}}$ for HS1. Because (**P1**) is not solved to optimality, the relative difference presented in the Table is optimistic. As observed in Adelman (2007), we expect the objective value to be close to z_{P1}, and therefore the results to be representative. For this set of instances, the difference is relatively small, with the largest difference about 8% for the case with $\tau = 50$ and $K = 12$.

However, we use another set of instances to show that the relative difference $z_{\mathrm{LP}}/z_{\mathrm{P1}}$ can be huge. We consider hub-and-spoke networks with $K \in \{2, 4, 8, 16\}$ nonhub locations. This set of instances is called HS2, subsequently. There are flights to and from the hub for each nonhub location. Figure 2 shows the network structure when $K = 4$. One product is offered for each itinerary. The fare is generated from a uniform distribution with range $[75, 250]$, which is denoted by $\mathcal{U}[75, 250]$. The choice parameter value for each product is three times a random number drawn from the Poisson distribution with mean 20. The no-purchase value for each segment is set to 1. We draw E_l from the uniform distribution with range 0.1–0.7 for each l and set $\lambda_l = 0.8E_l / \sum_{l'} E_{l'}$. We generated problem instances with $\tau = \{20, 50, 100, 200, 400, 800\}$. Capacity is taken to be the same across legs in each instance. Table 5 reports load factor and capacity for the instances. Table 6 shows the approximate relative difference $z_{\mathrm{LP}}/z_{\mathrm{P1}}$ for HS2. For $\tau = 20$ and $K = 16$, the relative difference is about 53%. This shows that

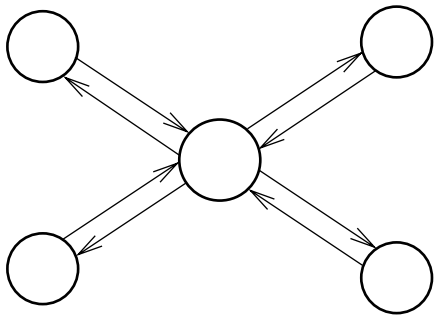

Figure 2 Network Structure for HS2 Instances with $K = 4$

z_{P1} can serve as a much better revenue bound. The magnitude of difference reported here is similar to that reported in Adelman (2007).

In both Tables 4 and 6, the approximate relative difference increases as the horizon length decreases. This behavior can be explained by the result of Liu and van Ryzin (2008), which shows that LP is asymptotically optimal as the problem scales up linearly in capacity and time. The difference also appears to be bigger for problems with more complex network structure.

5.2. Description of Simulated Instances

We studied the relative performance of four different heuristic policies. ADP was introduced in §4.1, and ADPD was introduced in §4.2. LP is the same as ADP where the dual values from **LP** are used instead of dynamic bid-prices. Linear program decomposition (LPD) was introduced in §4.3. In addition, we also tested each heuristic where (**LP**) and (**P1**) are resolved five times for equally spaced time intervals. Each heuristic is simulated 100 times using the same arrival streams. We tested the heuristics on four sets of instances. Table 7 reports the capacity configuration and load factor of the test instances, and Table 8 summarizes key statistics of the simulated instances. These test instances cover a wide variety of cases with different network structure and randomly generated choice parameters. For each set of instances, we consider $\tau \in \{20, 50, 100, 200, 400\}$.

PF1 (Four Parallel Flights). Parallel flights are scheduled flights on the same origin and destination

Table 4 Approximate Upper Bounds and Relative Difference for HS1

	Locations											
	2			4			8			12		
τ	z_{LP}	z_{P1}	$z_{\mathrm{LP}}/z_{\mathrm{P1}}$	z_{LP}	z_{P1}	$z_{\mathrm{LP}}/z_{\mathrm{P1}}$	z_{LP}	z_{P1}	$z_{\mathrm{LP}}/z_{\mathrm{P1}}$	z_{LP}	z_{P1}	$z_{\mathrm{LP}}/z_{\mathrm{P1}}$
50	4,769.99	4,685.68	1.02	7,829.55	7,681.12	1.02	8,865.23	8,400.01	1.06	8,756.77	8,125.07	1.08
100	10,400.20	10,286.70	1.01	14,512.40	14,304.70	1.01	17,906.80	17,343.90	1.03	17,266.10	16,648.40	1.04
200	16,996.20	16,748.40	1.01	28,820.00	28,540.00	1.01	34,230.40	33,448.50	1.02	38,154.10	37,198.00	1.03
400	39,471.10	39,334.40	1.00	59,009.30	58,415.30	1.01	70,714.00	69,336.90	1.02	73,638.30	72,106.40	1.02
800	69,810.90	69,079.90	1.01	114,076.00	112,267.00	1.02	137,934.00	135,953.00	1.01	144,887.00	142,255.00	1.02

Table 5 Load Factor and Capacity for HS2

	Number of nonhub locations, resources, and products							
	2, 4, 12		4, 8, 40		8, 16, 144		16, 32, 544	
τ	Load factor	Capacity per leg	Load factor	Capacity per leg	Load factor	Capacity per leg	Load factor	Capacity per leg
20	1.81205	3	1.57753	2	1.79335	1	0.92483	1
50	1.57180	8	1.54017	5	1.47362	3	2.29805	1
100	1.59299	19	1.60150	10	1.46147	6	1.53856	3
200	1.58430	34	1.58113	20	1.59850	11	1.54098	6
400	1.58434	68	1.60127	40	1.61746	22	1.66522	11
800	1.58774	126	1.58524	81	1.57865	43	1.62501	23

Table 6 Approximate Relative Difference for HS2

	Locations			
τ	2	4	8	16
20	1.05	1.17	1.41	1.53
50	1.02	1.05	1.12	1.35
100	1.01	1.03	1.07	1.13
200	1.01	1.02	1.04	1.07
400	1.01	1.02	1.03	1.05
800	1.00	1.02	1.03	1.04

on the same day. In practice, substitution among parallel flights is widely observed. We randomly generated instances with four parallel flights. A single class is offered on each flight. All the products belong to the same segment, and the arrival probability λ is 0.9 in each period. The fare is generated from $\mathcal{U}[10, 100]$. The MNL choice parameter for product j, v_j, is generated from the Poisson distribution with mean 100. The no-purchase weight, v_0, equals $0.5\sum_{j=1}^{n} v_j$; hence when all products are open, the no-purchase probability is 1/3.

PF2 (Eight Parallel Flights). This set of instances is the same as PF1 except there are eight instead of four parallel flights.

HS3 (Hub-and-Spoke Network with Two Nonhub Locations). There are four parallel flights from location 1 to the hub, and four parallel flights from the hub to location 2. The fare from location 1 to the hub is generated from $\mathcal{U}[1, 10]$, and the fare from the hub to location 2 is generated from $\mathcal{U}[10, 100]$. Each through itinerary fare is 0.95 times the corresponding sum of local itinerary fares. There are three disjoint product segments, one for location 1 to the hub, one for the hub to location 2, and one for the through itinerary (location 1–location 2). The arrival rates are 0.45, 0.225, and 0.225, respectively. The MNL choice parameter for product j, v_{lj}, is generated from the Poisson distribution with mean 100. The no-purchase weight for segment l, v_{l0}, equals $0.5\sum_{j\in N_l} v_{lj}$.

HS4 (Hub-and-Spoke Network with Four Nonhub Locations). This set of instances is generated the same way as HS1 except for the arrival rates. We assume the locations are marked 1–4. Locations 1 and 2 each have two parallel flights to the hub, and locations 3 and 4 each have two parallel flights from the hub. All products with the same origin and destination belong to the same segment, so there are 8 segments. Segments 1–4 correspond with local itineraries for locations 1–4, respectively. Segments 5–8 contain through itinerary products. The arrival rates for segments 1–8 are 0.2, 0.2, 0.05, 0.05, 0.1, 0.1, 0.1, and 0.1, respectively.

5.3. Policy Results

The bounds and simulated averages for the test instances are reported in Tables 9–12. To make it easy to compare the bounds and performance, we also report the relative difference of bounds and policy performance in Tables 13–14.

We report four different bounds. (**LP**) is solved to optimality in all the examples, and the objective value z_{LP} is reported in the LP-bound column. The value $\min_i\{\tilde{v}_1^i(c_i) + \sum_{k\neq i}\pi_k^* c_k\}$ is reported in the LPD-bound column. The objective value from (**P1**) is reported in P1-bound column. (**P1**) is solved to 0.5% of optimality. The value $\min_i\{v_1^i(c_i) + \sum_{k\neq i} V_{1,k}^* c_k\}$ is reported in the ADPD-bound column. Note that because (**P1**) is not solved to optimality, the ADPD bound inherits errors from the solution of (**P1**). Consequently, the reported ADPD bound is not always tighter than P1 bound, as suggested by Proposition 2.

Table 7 Load Factor and Capacities for Simulated Instances

	PF1		PF2		HS3		HS4	
τ	Capacity	Load factor	Capacity	Load factor	Capacity	Load factor	Capacity	Load factor
20	1 1 3 3	1.11	1 1 1 1 1 1 1 1	1.26	1 1 1 1 1 1 1 1	1.56	2 2 2 2 2 2 2 2	1.52
50	4 4 8 8	1.12	2 2 2 2 4 4 4 4	1.10	2 2 4 4 2 2 4 4	1.41	5 5 5 5 5 5 5 5	1.48
100	8 8 16 16	0.59	4 4 4 4 8 8 8 8	1.04	4 4 8 8 4 4 8 8	1.25	11 11 11 11 11 11 11 11	1.35
200	20 20 30 30	1.09	9 9 9 9 16 16 16 16	1.09	9 9 15 15 9 9 15 15	1.48	21 21 21 21 21 21 21 21	1.42
400	40 40 60 60	1.20	20 20 20 20 30 30 30 30	1.09	17 17 31 31 17 17 31 31	1.48	43 43 43 43 43 43 43 43	1.36

Table 8 Statistics of Simulated Test Instances

Case	PF1	PF2	HS3	HS4
No. of nonhub locations	—	—	2	4
No. of O-D pairs (segments)	1	1	3	8
No. of resources	4	8	8	8
No. of itineraries	4	8	24	24
No. of products	4	8	24	48

Our numerical results indicate that ADP can outperform LP with a revenue difference of up to 440%. Even with re-solving, ADP beats LP by up to 8%. Such a difference in revenue performance is quite significant for revenue management applications. We postulate that the difference in performance is because of the fact that (**P1**)–(**D1**) gives better bid-prices, which are time dependent. We also note that the bid-price of one resource affects the set of products offered; even products that do not use this particular resource can be affected because of customer choice among products. This should be contrasted with the bid-price controls for the independent demand case, where the bid-price of a particular resource only affects the controls of products that use this resource. We also note that the performance difference between LP and ADP tends to, although not always, be smaller for problems with longer time horizon. This behavior can be attributed to the asymptotic result of Liu and van Ryzin (2008).

5.4. Decomposition

LPD and ADPD are both based on decomposition of the network problem into many leg-based problems. Liu and van Ryzin (2008) report that LPD performs consistently better than LP. It is also our experience that decomposition-based approaches, including LPD and ADPD, usually perform better than naive bid-price controls.

We are, however, more interested in the relative performance of LPD and ADPD. Tables 13–14 show that without re-solving ADPD, it can perform up to 9% better than LPD. With re-solving, the difference can still be as high as 6%. Of course, to use ADPD, we need to solve (**P1**) and the harder dynamic programming recursion outlined in §4.2.

The relative performance of LPD and ADPD is affected by several factors. First, it depends on the *decomposability* of a particular problem. The basic decomposition idea is to decompose the dynamic programming value function $v_t(x)$ so that

$$v_t(x) \approx \sum_{i=1}^{m} v_t^i(x_i) \quad \forall\, t, x \tag{35}$$

for a set of functions $\{v_t^i(\cdot)\}_{\forall t,\, i}$. We say a problem is fully decomposable if the approximation in (35) is exact. To gain insights into the performance of decomposition heuristics, consider the following two problems:

(1) One flight with multiple fare classes;

(2) A parallel flight problem with independent demand for each flight.

The first problem is a single-leg problem and the second problem can be trivially decomposed into single-leg problems. Consequently, the decomposition heuristics actually give an optimal policy for both problems. In fact, the bid-prices from (**LP**) and (**P1**) are not needed for the decomposition.

Now, consider two modified problems:

(3) A hub-and-spoke network problem with one flight for each local itinerary and very low arrival rates for through itineraries;

(4) A parallel flight problem where only a small portion of customers switch flights if their most preferred flight is not offered.

Both problems listed above are clearly not fully decomposable because of a network effect and a customer choice effect, in contrast to problems 1–2, respectively. However, we expect that decomposition-based approaches work well for these problems, because the network and customer choice effects are weak. On the other hand, we observed from our experience that the relative performance difference between LPD and ADPD is bigger for problems with a stronger network effect or customer choice effect.

Table 9 Bounds and Simulated Average Revenue for PF1

Re-solving	τ	LP bound	LPD bound	LP	Stdev	LPD	Stdev	P1 bound	ADPD bound	ADP	Stdev	ADPD	Stdev
1	20	305.499	304.261	128.56	(6.41)	286.95	(36.93)	301.918	300.085	295.74	(19.72)	289.28	(31.14)
	50	1,734.610	1,733.540	378.69	0.00	1,589.82	(153.29)	1,728	1,721.080	1,666.95	(123.85)	1,642.89	(117.78)
	100	1,609.690	1,609.630	1,422.97	(194.74)	1,486.29	(110.67)	1,608.83	1,608.620	1,462.21	(163.93)	1,551.98	(83.33)
	200	3,550.270	3,550.270	3,075.73	(266.17)	3,341.04	(173.92)	3,550.24	3,550.350	3,526.60	(56.33)	3,460.37	(137.83)
	400	7,359.560	7,359.560	3,393.99	0.00	7,175.14	(232.30)	7,359.56	7,377.720	7,351.05	(42.54)	7,295.36	(144.87)
5	20			281.49	(34.32)	290.44	(28.18)			296.18	(22.04)	296.11	(23.19)
	50			1,566.05	(155.77)	1,636.20	(118.83)			1,670.61	(114.00)	1,666.27	(102.39)
	100			1,546.57	(77.32)	1,531.20	(76.06)			1,564.52	(74.23)	1,575.57	(50.46)
	200			3,385.39	(132.20)	3,433.37	(115.59)			3,525.36	(55.95)	3,518.41	(70.71)
	400			6,932.41	(283.70)	7,232.18	(152.03)			7,318.45	(128.24)	7,338.04	(67.18)

Table 10 Bounds and Simulated Average Revenue for PF2

Re-solving	τ	LP bound	LPD bound	LP	Stdev	LPD	Stdev	P1 bound	ADPD bound	ADP	Stdev	ADPD	Stdev
1	20	424.272	422.495	166.18	(14.04)	390.77	(40.86)	399.752	399.925	395.81	(36.65)	395.04	(38.32)
	50	1,177.180	1,172.820	1,071.83	(107.29)	1,001.83	(125.58)	1,149.930	1,135.450	1,106.87	(87.33)	1,094.14	(99.16)
	100	2,498.590	2,498.590	1,573.23	(21.80)	2,249.15	(174.79)	2,493.980	2,496.900	2,443.76	(78.10)	2,397.51	(119.18)
	200	6,263.010	6,262.970	4,596.93	(520.95)	5,294.28	(400.87)	6,250.530	6,260.750	6,040.58	(213.50)	5,998.74	(297.17)
	400	12,620.300	12,620.300	11,500.01	(454.43)	11,427.90	(651.68)	12,618.300	12,663.700	12,543.88	(161.25)	12,354.94	(370.28)
5	20			360.35	(55.75)	389.41	(39.64)			391.99	(41.23)	393.57	(38.61)
	50			1,047.74	(96.35)	1,069.40	(90.52)			1,105.67	(91.97)	1,114.75	(82.59)
	100			2,276.78	(135.91)	2,346.49	(128.18)			2,449.23	(71.97)	2,445.00	(70.29)
	200			5,867.58	(238.17)	5,899.84	(251.61)			6,145.10	(136.46)	6,116.54	(182.68)
	400			12,029.51	(393.85)	12,131.83	(336.61)			12,496.28	(182.70)	12,520.95	(163.59)

Table 11 Bounds and Simulated Average Revenue for HS3

Re-solving	τ	LP bound	LPD bound	LP	Stdev	LPD	Stdev	P1 bound	ADPD bound	ADP	Stdev	ADPD	Stdev
1	20	282.75	276.20	198.47	(69.10)	215.59	(61.70)	250.24	250.49	235.58	(51.41)	235.58	(51.23)
	50	885.26	873.80	708.21	(133.30)	728.02	(131.93)	854.10	843.17	772.03	(109.20)	779.50	(113.09)
	100	936.57	931.79	828.25	(107.30)	849.25	(96.11)	923.70	920.54	847.81	(72.71)	881.78	(70.62)
	200	3,322.46	3,316.06	2,787.66	(277.20)	3,110.89	(242.15)	3,307.20	3,314.41	2,894.62	(275.36)	3,134.01	(220.60)
	400	6,680.41	6,674.38	6,123.25	(403.73)	6,393.09	(319.41)	6,664.83	6,681.96	6,226.28	(305.55)	6,480.61	(275.52)
5	20			219.84	(61.51)	223.02	(60.54)			237.25	(48.66)	237.17	(48.05)
	50			748.63	(113.22)	768.14	(99.57)			785.59	(98.27)	791.82	(93.28)
	100			844.74	(81.24)	875.66	(74.21)			877.89	(73.81)	885.71	(62.41)
	200			3,088.68	(184.74)	3,169.09	(167.84)			3,166.23	(176.80)	3,229.29	(122.21)
	400			6,368.18	(242.72)	6,486.49	(222.64)			6,432.14	(278.87)	6,522.37	(214.97)

Table 12 Bounds and Simulated Average Revenue for HS4

Re-solving	τ	LP bound	LPD bound	LP	Stdev	LPD	Stdev	P1 bound	ADPD bound	ADP	Stdev	ADPD	Stdev
1	20	2,534.41	2,501.63	1,790.82	(386.52)	1,975.15	(367.98)	2,371.81	2,369.42	2,053.02	(328.91)	2,095.84	(338.19)
	50	6,418.83	6,388.56	3,965.17	(753.34)	5,478.94	(579.81)	6,300.67	6,292.95	5,626.97	(558.80)	5,783.79	(462.26)
	100	13,889.00	13,877.90	11,110.40	(1,115.40)	12,933.26	(728.68)	13,660.20	13,671.60	12,848.76	(716.39)	12,912.86	(736.21)
	200	26,454.40	26,422.00	22,504.33	(1,341.46)	24,437.91	(1,243.38)	26,347.20	26,413.10	24,319.58	(1,261.57)	25,081.66	(1,095.66)
	400	55,660.10	55,611.30	51,122.14	(2,409.41)	53,684.53	(1,666.95)	55,409.80	55,541.50	51,638.53	(2,300.02)	53,712.35	(1,654.14)
5	20			1,963.30	(379.91)	2,054.06	(332.89)			2,095.46	(312.13)	2,082.81	(313.85)
	50			5,398.34	(537.58)	5,642.24	(460.03)			5,769.49	(486.50)	5,793.13	(459.11)
	100			12,279.58	(862.97)	12,929.16	(687.85)			12,805.56	(796.65)	12,980.10	(690.57)
	200			24,357.47	(1,169.01)	24,901.04	(947.68)			24,783.93	(1,060.12)	25,117.72	(946.74)
	400			51,863.12	(1,961.96)	53,722.25	(1,315.53)			52,225.56	(1,864.89)	53,855.08	(1,369.68)

Table 13 Relative Difference in Bounds and Policy Performance for Parallel Flights Instances PF1 and PF2

		PF1 (%)				PF2 (%)			
Re-solving	τ	LP bound/ P1 bound	LPD bound/ ADPD bound	ADP/LP	ADPD/LPD	LP bound/ P1 bound	LPD bound/ ADPD bound	ADP/LP	ADPD/LPD
1	20	101.19	101.39	230.04	100.81	106.13	105.64	238.18	101.09
	50	100.38	100.72	440.19	103.34	102.37	103.29	103.27	109.21
	100	100.05	100.06	102.76	104.42	100.18	100.07	155.33	106.60
	200	100.00	100.00	114.66	103.57	100.20	100.04	131.40	113.31
	400	100.00	99.75	216.59	101.68	100.02	99.66	109.08	108.11
5	20			105.22	101.95			108.78	101.07
	50			106.68	101.84			105.53	104.24
	100			101.16	102.90			107.57	104.20
	200			104.13	102.48			104.73	103.67
	400			105.57	101.46			103.88	103.21

Table 14 Relative Difference in Bounds and Policy Performance for Hub-and-Spoke Network Instances HS3 and HS4

		HS3 (%)				HS4 (%)			
Re-solving	τ	LP bound/ P1 bound	LPD bound/ ADPD bound	ADP/LP	ADPD/LPD	LP bound/ P1 bound	LPD bound/ ADPD bound	ADP/LP	ADPD/LPD
1	20	112.99	110.27	118.70	109.27	106.86	105.58	114.64	106.11
	50	103.65	103.63	109.01	107.07	101.88	101.52	141.91	105.56
	100	101.39	101.22	102.36	103.83	101.67	101.51	115.65	99.84
	200	100.46	100.05	103.84	100.74	100.41	100.03	108.07	102.63
	400	100.23	99.89	101.68	101.37	100.45	100.13	101.01	100.05
5	20			107.92	106.34			106.73	101.40
	50			104.94	103.08			106.88	102.67
	100			103.92	101.15			104.28	100.39
	200			102.51	101.90			101.75	100.87
	400			101.00	100.55			100.70	100.25

The impact of the customer choice effect is demonstrated by the results of PF1–PF2. PF2 has more pronounced customer choice effect because more flights are involved. Table 13 shows the difference between LPD and ADPD is much larger for PF2 than PF1 cases. This suggests that the benefit of using dynamic over static bid-prices increases as the problem becomes less decomposable. We suspect that this is because time-dependent interactions among resources become more complex.

Other factors that affect the relative performance between LPD and ADPD include load factor and capacity asymmetry. When load factor is very high or very low, the performance difference is usually small. This is not surprising because effective revenue management control is most needed when the load factor is in an intermediate range. We also observe that when flights differ significantly in their capacity, the performance difference is usually bigger. When flight capacity is asymmetric, it is more likely that some resources are more critical than others. Hence, it is more important to use accurate bid-prices in such cases. Our observation is derived from our experience when designing and conducting the numerical experiments. The observation is useful because it can give some guidelines on experimental design if it is of interest to compare our approach to other related approaches.

References

Adelman, D. 2007. Dynamic bid-prices in revenue management. *Oper. Res.* **55**(4) 647–661.

Belobaba, P. P., L. R. Weatherford. 1996. Comparing decision rules that incorporate customer diversion in perishable asset revenue management situations. *Decision Sci.* **27**(2) 343–363.

Bertsekas, D. P., J. N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

Boyd, S., L. Vandenberghe. 2004. *Convex Programming*. Cambridge University Press, Cambridge, UK.

Bront, J. M., I. Mendez-Diaz, G. Vulcano. 2007. A column generation algorithm for choice-based network revenue management. *Oper. Res.* Forthcoming.

Brumelle, S. L., J. I. McGill, T. H. Oum, K. Sawaki, M. W. Tretheway. 1990. Allocation of airline seats between stochastically dependent demands. *Transportation Sci.* **24**(3) 183–192.

de Farias, D., B. Van Roy. 2003. The linear programming approach to approximate dynamic programming. *Oper. Res.* **51**(6) 850–865.

Gallego, G., G. Iyengar, R. Phillips, A. Dubey. 2004. Managing flexible products on a network. CORC Technical Report Tr-2004-01, IEOR Department, Columbia University, New York.

Jiang, H., G. Miglionico. 2006. Airline network revenue management with buy-up. Working Papers 11/2006, Judge Business School, University of Cambridge, Cambridge, UK.

Kunnumkal, S., H. Topaloglu. 2008. A refined deterministic linear program for the network revenue management problem with customer choice behavior. *Naval Res. Logist.* **55**(6) 563–580.

Liu, Q., G. van Ryzin. 2008. On the choice-based linear programming model for network revenue management. *Manufacturing Service Oper. Management* **10**(2) 288–310.

Powell, W. 2007. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, New York.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York.

Schaible, S., J. Shi. 2003. Fractional programming: The sum-of-ratios case. *Optim. Methods Software* **18**(2) 219–229.

Schweitzer, P., A. Seidmann. 1985. Generalized polynomial approximations in Markovian decision processes. *J. Math. Anal. Appl.* **110**(2) 568–582.

Talluri, K., G. van Ryzin. 2004a. Revenue management under a general discrete choice model of consumer behavior. *Management Sci.* **50**(1) 15–33.

Talluri, K., G. van Ryzin. 2004b. *The Theory and Practice of Revenue Management*. Kluwer Academic Publishers, Boston.

Van Ryzin, G., G. Vulcano. 2004. Simulation-based optimization of virtual nesting controls under consumer choice behavior. Working paper, Columbia Graduate School of Business, New York.

Zhang, D., W. L. Cooper. 2005. Revenue management for parallel flights with customer-choice behavior. *Oper. Res.* **53**(3) 415–431.

Zhang, D., W. L. Cooper. 2009. Pricing substitutable flights in airline revenue management. *Eur. J. Oper. Res.* **197** 848–861.

Zhao, W., Y.-S. Zheng. 2001. A dynamic model for airline seat allocation with passenger diversion and no-shows. *Transportation Sci.* **35**(1) 80–98.