TRANSPORTATION SCIENCE
Vol. 41, No. 3, August 2007, pp. 332–353
ISSN 0041-1655 | EISSN 1526-5447 | 07 | 4103 | 0332

DOI 10.1287/trsc.1060.0184
© 2007 INFORMS

Some Decomposition Methods for Revenue Management

William L. Cooper
Department of Mechanical Engineering, University of Minnesota, Minneapolis, Minnesota 55455, billcoop@me.umn.edu

Tito Homem-de-Mello
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60208, tito@northwestern.edu

Working within a Markov decision process (MDP) framework, we study revenue management policies that combine aspects of mathematical programming approaches and pure MDP methods by decomposing the problem by time, state, or both. The "time decomposition" policies employ heuristics early in the booking horizon and switch to a more-detailed decision rule closer to the time of departure. We present a family of formulations that yield such policies and discuss versions of the formulation that have appeared in the literature. Subsequently, we describe sampling-based stochastic optimization methods for solving a particular case of the formulation. Numerical results for two-leg problems suggest that the policies perform well. By viewing the MDP as a large stochastic program, we derive some structural properties of two-leg problems. We show that these properties cannot, in general, be extended to larger networks. For such larger networks we also present a "state-space decomposition" approach that partitions the network problem into two-leg subproblems, each of which is solved. The solutions of these subproblems are then recombined to obtain a booking policy for the network problem.

Key words: network revenue management; yield management; stochastic optimization; Markov decision processes
History: Received: October 2003; revision received: February 2006; accepted: October 2006.

1. Introduction

A wide number of industries face problems in which various products, comprised of combinations of resources, are sold to multiple classes of customers over a finite time horizon. Different classes of customers pay different amounts and consume different quantities of resources. Resources that are not sold prior to the end of the time horizon yield no revenue and consequently "perish." The collection of methods for dynamically controlling the availability of the different products to increase revenues is known as *revenue management*. These methods are particularly important in the airline industry, where the resources are seats on flight legs. Revenue management also plays an important role in the hotel, rental car, and broadcasting industries. For the sake of concreteness, we use the terminology of airlines throughout. See Talluri and van Ryzin (2004b) and Bitran and Caldentey (2003) for surveys of the revenue management literature.

Much of the recent research in revenue management has focused in two separate areas: Markov decision process (MDP) models for single-leg problems and mathematical programming formulations for multileg network problems. Lee and Hersh (1993) consider an MDP model for managing sales of airline seats to multiple customer classes. They prove a number of structural properties of the problem. Their work has subsequently been extended by numerous authors. Of note are Subramanian, Stidham, and Lautenbacher (1999), who consider cancellations and no-shows, and Talluri and van Ryzin (2004a), who model consumer choice behavior. Others have considered closely related continuous-time models; some work of this nature includes Kleywegt and Papastavrou (1998) and Feng and Xiao (2000). With a few exceptions mentioned later, most work within the MDP framework has dealt exclusively with single-leg problems. There appear to be two reasons for this: First, the so-called curse of dimensionality renders MDPs impractical for larger problems; second, it has proved difficult to obtain structural results for larger networks.

There is also a large body of literature that considers mathematical programming approaches to revenue management. Such work aims to capture the tradeoffs among the many different itineraries that can be built across a network of flights. However, such methods typically do not completely capture the dynamics or the randomness of the booking process. Nevertheless, such techniques do yield computationally tractable and practically implementable methods.

Williamson (1992) nicely summarizes many of these techniques. Other key work includes Bertsimas and Popescu (2003) and Bertsimas and de Boer (2005). For references to both MDP and math programming research, see Talluri and van Ryzin (2004b). We will provide more discussion of some math programming approaches in §§4 and 5.

In this paper, we present an MDP formulation of the network revenue management problem. One notable feature of our formulation is that we take care to ensure that standard allocation and booking-limit policies are feasible policies for the MDP. Interestingly, earlier MDP approaches that use remaining capacity as the state vector leave these well-known heuristics beyond the scope of the model. (Although some previous MDP formulations do employ state vectors that allow for these heuristics, the motivation is usually the inclusion of overbooking and cancellations in the model.) Of course, this is largely a mathematical issue, because allocation and booking-limit policies are, in fact, used in practice. Nevertheless, because we aim to analyze such policies within the framework of an MDP, it is important that the MDP be set up in a way that makes this possible.

One focus of this paper is a family of "time-decomposition" formulations that combine aspects of both the MDP and the mathematical programming approaches. The basic idea is, when far from the time of departure, to use simple decision rules and then to switch to the decision rules specified by an optimal policy when we are close enough to the end of the horizon. We present a formulation that yields such policies—notably, the formulation explicitly accounts for the fact that such a switch will take place. The "simple decision rules" mentioned above can typically be parameterized fairly easily, so that solving the formulation involves searching over the parameter space. (For instance, the parameter space could be the set of allocation vectors to be used prior to switch to the optimal policy.)

For moderate-sized problems, if the switching time is close enough to departure time, this type of policy will result in modest savings in terms of computation time and data storage, compared to using the MDP procedure for the entire horizon. Of course, there is a tradeoff, because the longer the optimal policy is in effect, the higher we expect revenues to be.

For larger problems, direct application of such a policy remains impractical. However, the formulation remains useful if it is possible to come up with approximations for what happens when an optimal policy is in use. In such cases, one could use such a formulation to make decisions "on the fly."

We present numerical results that suggest that time-decomposition policies using allocations in the first stage perform well for certain problems. In fact, in a range of one- and two-leg problems, our experiments have shown that despite the apparent crudeness of allocation policies, a time-decomposition policy with allocations performs nearly as well as an optimal policy. The time-decomposition policies are motivated in part by results that suggest relatively crude control methods are good when remaining capacity and expected demand are both large; see Gallego and van Ryzin (1994, 1997) and Cooper (2002). Additional motivation comes from the suggestion of Subramanian, Stidham, and Lautenbacher (1999, p. 163) that near the time of departure, when remaining demand and capacity are typically small, it is important to employ decision rules based on a detailed, accurate model. It is important to point out, however, that the most crucial and difficult decisions need not occur near the time of departure. As one referee remarked to us, properly rejecting a low-fare request far from departure may be as important as doing so at the very end of the horizon.

The time-decomposition formulation we propose falls into the general category of static stochastic optimization problems, where the goal is to maximize the expected value of a given random function. This class of problems has long been studied, and some algorithms have been proposed, notably the stochastic approximation method and its variants. We take a different approach and apply recently developed techniques that incorporate Monte Carlo sampling into a deterministic optimization method. Our choice is driven by successful applications of this type of method to other problems and by the fact that sampling is readily accomplished in our model, because underlying distributions are assumed to be known.

To deal with multileg problems, we propose a state-space decomposition heuristic that breaks a network problem into a collection of two-leg problems. The approach is inspired by methods that decompose network problems into single-leg problems, replacing fares with suitable displacement adjusted fares (e.g., Talluri and van Ryzin 2004b, Bertsimas and de Boer 2005). The individual two-leg problems are then solved using our time-decomposition approach (or another method, if so desired), and the resulting policies are reconstituted to obtain a policy for the original network problem. The method appears to be well suited for hub-and-spoke networks.

A number of authors have employed simulation-based optimization or related methods to solve revenue management problems. Robinson (1995) employs Monte Carlo integration to solve an expected marginal seat revenue (EMSR) formulation. van Ryzin and McGill (2000) describe a Robbins-Monro-like approach—a stochastic approximation (SA)-type method—for a similar formulation. Talluri and van

Ryzin (1999) use Monte Carlo estimates to obtain bid-price controls, and Karaesmen and van Ryzin (2004) employ a stochastic gradient method (another SA variant) to solve an overbooking problem. Zhang and Cooper (2005) describe a simulation-based search procedure for generating heuristics for problems where customers choose among flights.

Few papers have considered how to blend the MDP and math programming approaches. Bertsimas and de Boer (2005) describe a procedure designed to obtain a nested booking-limit policy. They rely on stochastic-gradient techniques as well as a carefully devised procedure to estimate derivatives by finite differences. To handle larger problems they discuss methods for approximating an MDP value function—we discuss this work in greater detail in §5. A continuous version of the Bertsimas-de Boer formulation has been studied by van Ryzin and Vulcano (2006a). Bertsimas and Popescu (2003) use math programming methods to approximate the opportunity cost of accepting a particular class. These procedures, in turn, yield heuristics for solving the MDP optimality equations. Günther (1998) employs related approximation methods, which also allow for a blend of math programming and MDP approaches. One of his key ideas is to use the solutions of math programs as a proxy for the MDP value function when making booking decisions.

We also derive some structural properties for the model under study. By recasting the MDP as a multistage stochastic linear integer program and employing results from integer programming, we are able to show that the value function for two-leg problems is *concave* in each component. You (1999) has proved a similar result; however, our proof technique is quite different and may therefore be of interest in its own right. In addition, our approach is relevant to problems with non-Markovian demand processes. We present an example that demonstrates that the value function need not be componentwise concave when there are three legs in the network. Also, we prove that the expected revenue from the time-decomposition policy is non-increasing as a function of the switch time. Thus, the blend of techniques from MDP theory, stochastic programming, Monte Carlo methods, and integer programming allows us to derive theoretical properties as well as a numerical method.

In summary, the primary contributions of this paper are (1) description and analysis of the general time-decomposition approach within a formal stochastic and dynamic framework, (2) adaptation of recently developed stochastic optimization techniques to solve a particular instance of the formulation, (3) derivation of some structural results on the MDP value function for multiple-leg problems, (4) development of a state-space decomposition approach that can be used in combination with any method that efficiently solves two-leg problems, and (5) demonstration of the use of both time and state-space decomposition approaches on numerical examples.

The remainder of the paper is organized as follows: Section 2 describes the basic network revenue management problem; §3 provides a formulation of the problem as an MDP; §4 reviews some mathematical programming approaches to the problem; §5 describes time-decomposition methods that contain aspects of the MDP and math programming techniques; §6 includes structural properties for two-leg problems obtained by recasting the MDP as a large stochastic program; §7 describes sampling-based optimization procedures for solving one variant of the time-decomposition method and presents the state-space decomposition approach; §8 contains numerical examples; and §9 presents a summary and conclusions.

2. Problem Framework

We model the revenue management problem as a finite-horizon discrete-time MDP with time units $t = 1, \dots, \tau$. We assume that time moves forward; $t = 1$ is the time at which sales commence and $t = \tau$ is the last period in which sales can occur. There are m resources, combinations of which can be used to create n possible products. Each arriving customer is assumed to belong to one of the n product classes, where a class j customer is defined to be someone who requests product j. We will use the terms *demand, request,* and *arrival* interchangeably. Nevertheless, the general approach that we describe could also be applied to models that do allow choice behavior.

Let $p_{t,j}$ be the probability that a class j arrival occurs at time t and $D_{t,j}$ be the random demand for class j at time t. So $P(D_{t,j} = 1) = p_{t,j}$, and $P(D_{t,j} = 0) = 1 - p_{t,j}$. As is customary in the revenue management literature, we assume there is at most one unit of demand per time period, so that $P(\exists j_1, j_2\colon j_1 \neq j_2, D_{t,j_1} = D_{t,j_2} = 1) = 0$. The formulation does not allow multiple requests in a single period, so we do not need to worry about situations in which there are two simultaneous requests and acceptance of both of them would lead to violation of the capacity constraint. The arrivals during different time units are assumed to be independent, and the arrival process is not affected by the acceptance-rejection policy (of course, the policy does affect which customers are accepted and which are rejected). For each t, we use D_t to denote the n vector with entries $D_{t,1}, \dots, D_{t,n}$. We use a similar convention throughout for vectors; we simply drop the subscript to denote the vector of appropriate dimension. For example, p_t is the n-vector $p_t \equiv (p_{t,j})$. Let $p_{t,0} = P(D_t = 0)$.

There is an m-vector of resources $c \equiv (c_i)$, and an $(m \times n)$ matrix $A \equiv (a_{ij})$. The entry $c_i > 0$ represents the (integer) amount of resource i present at the beginning of the time horizon (just prior to $t = 1$), and the entry $a_{ij} \in \{0, 1\}$ is the amount of resource i required by a class j customer. In the airline setting, this model can represent demand for a network of flights that depart on a particular day. In this case, the resources are leg-cabin combinations (so vector c represents the number of seats in leg-cabins throughout the network), and the products are itinerary-fare class combinations. Row i of A has zeroes in the columns corresponding to fare class-itineraries that do not use leg-cabin i and ones in the columns corresponding to itinerary-fare classes that do use leg-cabin i.

Whenever a class j product is sold, we say that we have accepted a class j customer. In this case, the customer pays the fare f_j and consumes the resource requirement, and thereafter fewer resources are available to sell to future customers (recall that we are not considering overbookings in our model). In contrast, we say that a customer's request is rejected if he or she is not sold the requested product. In this case, no revenue is realized and no resources are consumed.

3. MDP Methods

In this section, we describe an MDP formulation of the revenue management problem, taking care to ensure that various well-known heuristics are indeed valid control policies within the MDP framework. Loosely speaking, in each time period we must specify which classes we are willing to accept and which we would reject, based on what has occurred so far. Somewhat more formally, we must specify an *action* by choosing an element from the set $\{0, 1\}^n$, where a "1" in the jth position indicates that we are willing to accept a class j customer in period t, should one arrive. An entry "0" indicates that a class j customer will be rejected. See Puterman (1994) for further discussion of the MDP terminology used throughout this section.

The *state* at time t is the n-vector whose entries are the cumulative number of accepted bookings in each class prior to the arrival in period t. Although this is not the usual definition of state, certain natural history-dependent decision rules (e.g., allocations as described in §§4 and 5) require this information. If we instead defined the state to be the vector of the number of seats remaining on each leg, then several types of booking limit policies would not be "admissible"—we will return to this issue shortly. We denote the random sequence of states by $(S_1, \ldots, S_{\tau+1})$ and the random sequence of actions by $(U_1, \ldots, U_\tau)$; recall that each S_t and U_t is an n-dimensional vector. Given state and action for time t, the state in time $t+1$ is conditionally independent of past states and actions and has mass function given by $P(S_{t+1} = s' \mid S_t = s, U_t = u, S_1, \ldots, S_{t-1}, U_1, \ldots, U_{t-1}) = P(s + uD_t = s')$, where uD_t is the n-dimensional vector with entries $u_j D_{t,j}$.

If action u is selected in state s at time t, a *single-period* reward of $g_t(s, u) = E[f \cdot (uD_t)]$ is received. In period $\tau + 1$ there is no action, and $g_{\tau+1}(s) = 0$ for all s. Note that this formulation has a single-period reward function that is the *expected revenue* in the period. Because our ultimate objective is to maximize expected revenue (see Equation (1) below), taking this "extra" expectation prior to any time t arrival does not alter our results.

A *history up to time t* is a sequence of actions and states prior to time t; more precisely, a history up to t is a sequence of the form $h_t = (s_1, u_1, s_2, u_2, \ldots, s_t)$ where $s_k = (s_{k,1}, \ldots, s_{k,n})$ is the state at time k and $u_k = (u_{k,1}, \ldots, u_{k,n})$ is the action taken at time k. The history up to time t does not include the possible arrival in period t. We denote the set of all possible histories up to time t by $\mathcal{H}_t$. A *decision rule* for time t maps a history up to time t to an action—that is, a decision rule for time t specifies who we are willing to accept as a function of what has happened already. For decision rule π_t and history h_t for time t, the statement $\pi_t(h_t) = u \in \{0, 1\}^n$ means that based on the occurrence of h_t, we select action u.

We also employ the notation $\pi_{t,j}(h_t)$ to indicate the accept/reject rule for a potential class j arrival at time t, as specified by π_t. For a particular history h_t prior to time t, if we are willing to accept class j at time t, then $\pi_{t,j}(h_t) = 1$. On the other hand, if we are not willing to accept class j at time t, then $\pi_{t,j}(h_t) = 0$. Observe that the above definitions of state and action enable us to specify any accept/reject rule that is a function of remaining capacity; that is, decision rules of the form "select action u when the remaining capacity vector is r at time t" can be written as $\pi_t(h_t) = u$ for all $h_t = (s_1, u_1, s_2, u_2, \ldots, s_t)$ for which $c - As_t = r$.

A *policy* specifies what decision rule is in use each period. We will use $\pi = (\pi_1, \ldots, \pi_\tau)$ to denote a generic policy, where π_t is the decision rule for period t. For a particular policy, note that the actual action implemented by a decision rule in period t is random, because the history up to that point is random—that is, the sequence of prior states (and therefore prior actions) depends on the realized values of $(D_1, \ldots, D_{t-1})$.

The objective is, starting from no seats sold at time $t = 1$, to maximize the expected revenue

$$\max_{\pi \in \Pi} \mathbb{E} \sum_{t=1}^{\tau} g(S_t^\pi, \pi_t(H_t^\pi)) = \max_{\pi \in \Pi} \mathbb{E} f \cdot S_{\tau+1}^\pi = \max_{\pi \in \Pi} \mathbb{E} \sum_{t=1}^{\tau} f \cdot N_t^\pi, \quad (1)$$

where N_t^π has entries $N_{t,j}^\pi = D_{t,j}\pi_{t,j}(H_t^\pi)$ and H_t^π is the random history prior to time t. The superscript π indicates the dependence of the random quantities on the choice of policy. Finally, Π is the set of all possible policies.

Before proceeding, let us briefly return to the issue of our choice of state vector. Consider an example with two classes and one leg with scalar capacity c, and suppose we instead used the common setup wherein the state is the number of seats remaining. Suppose now that we want to consider a policy that accepts at most one booking in class 2, and we do not want to limit class 1 bookings at all (except, of course, that we cannot exceed capacity). Because such policies are widespread in practice, it is important that our formulation be able to handle them. Imagine that in time unit 1 we institute the decision rule $(1, 1)$, which says we will accept either class of customer. Let Scenario 1 be the case where there is a class 1 arrival in period 1 and Scenario 2 be the case where there is a class 2 arrival in period 1. In each scenario, if we were to make the state equal to remaining capacity, then the information available at the beginning of period 2 (that is, the history up to $t = 2$) would be $h_t = (c, (1, 1), c - 1)$. However, from h_t we have no way of determining whether the customer we booked in period 1 was from class 1 or class 2. Hence, we have no way to institute the policy "accept at most one from class 2."

Under the preceding assumptions, there exists a Markovian optimal policy (see, e.g., Puterman 1994). This means that there is a policy that maximizes (1) and for each t is a function solely of the current state. Let s denote a generic state. Then one can compute an optimal policy and the corresponding maximum in (1) via the recursive optimality equations

$$w_t(s) = \max_{\substack{u\in\{0,1\}^n \\ P(A(s+uD_t)\le c)=1}} \mathbb{E}[f\cdot(uD_t) + w_{t+1}(s+uD_t)] \quad (2)$$

and $w_{\tau+1}(\cdot) = 0$. For each $t \in \{1, \ldots, \tau+1\}$, $w_t(s)$ represents the maximum expected revenue that can be generated in periods $t, \ldots, \tau$, provided that s is the state in period t. Moreover, $w_1(0)$ is the maximum in (1). If $\pi^* = (\pi_1^*, \ldots, \pi_\tau^*)$ is such that $\pi_t^*(s)$ is a maximizer of the right-hand side of (2), then π^* is an optimal policy—note that the argument of $\pi_t^*(\cdot)$ is the current state.

As the following result shows, there exist Markovian optimal policies whose decisions depend only on the remaining capacity vector $c - As$. Thus, to search for an *optimal* policy, we can restrict ourselves to the smaller state space obtained by collapsing any two states $s, \bar{s}$ such that $c - As = c - A\bar{s}$ into a single state. This illustrates why in many papers it is assumed that the state vector is the remaining capacity vector. However, as pointed out earlier, such a selection would render standard partitioned-allocation and booking-limit policies unmeasurable with respect to the history. Since in our case we deal with the latter policies, it is important to consider the framework with the larger state space.

Proposition 1. *If states s and $\bar{s}$ are such that $As = A\bar{s}$, then $w_t(s) = w_t(\bar{s})$. Hence, there exists a Markovian optimal policy so that $\pi_t^*(s) = \pi_t^*(\bar{s})$ whenever $As = A\bar{s}$.*

Proof. The proof of the first statement is by backward induction on t. In view of the boundary condition $w_{\tau+1}(\cdot) = 0$, it follows that the right side of (2) is identical for s and $\bar{s}$ with $As = A\bar{s}$. Therefore, $w_\tau(s) = w_\tau(\bar{s})$, which completes the base case of the induction. The argument for the inductive step is identical, because $A(s + uD_t) = A(\bar{s} + uD_t)$. The second assertion above follows immediately from (2) and the first assertion. □

In light of the above, we can, without loss of optimality, reformulate (2) in the standard form where the argument of the value function is the remaining capacity vector—under this reformulation, we denote the value function by $v_t(\cdot)$. So we have $v_t(c - As) = w_t(s)$ for all s and t. In the following, r is an m-vector of remaining capacities. The reformulation value function $v_t(\cdot)$ is determined by

$$v_t(r) = \max_{\substack{u\in\{0,1\}^n \\ P(r-A(uD_t)\ge 0)=1}} \mathbb{E}[f\cdot(uD_t) + v_{t+1}(r - A(uD_t))] \quad (3)$$

and $v_{\tau+1}(r) = 0$. Typically, the number n of classes is considerably larger than the number m of legs, so such a reformulation is desirable insomuch as it greatly reduces the amount of data that must be stored. On the other hand, introducing cancellations into the model would necessitate using the original, larger formulation—see, for instance, Subramanian, Stidham, and Lautenbacher (1999) for an example in the single-leg case.

Next, let $\mathbb{I}\{\cdot\}$ be the indicator function, e_j be the n-dimensional vector with a one in the j position and zeroes elsewhere, and A^j be the jth column of the matrix A. For each t, j, and r define

$$\begin{aligned}\pi_{t,j}^*(s) &= \mathbb{I}\{f_j \ge w_{t+1}(s) - w_{t+1}(s+e_j)\}\,\mathbb{I}\{A(s+e_j)\le c\} \\ &= \mathbb{I}\{f_j \ge v_{t+1}(c - As) - v_{t+1}(c - As - A^j)\} \\ &\quad \cdot \mathbb{I}\{A^j \le c - As\}. \end{aligned} \quad (4)$$

Then π^* is a Markovian optimal policy for (1). We omit proof of this fact, because results of this type have been obtained elsewhere (e.g., Lautenbacher and Stidham 1999; Talluri and van Ryzin 1998). Intuitively, (4) states that we should accept a booking request if and only if there is enough capacity and the corresponding fare exceeds the loss in future expected revenue that would be caused by accepting the request.

Recall that demand $D_{t,j}$ is a Bernoulli random variable with $P(D_{t,j}=1)=p_{t,j}$. Moreover, $P(D_t=e_j)=p_{t,j}$ and $\sum_j P(D_t=e_j)+P(D_t=0)=1$, so $\mathbb{E}D_t=p_t$. Hence, the constraint set in (3) can be written as

$$\begin{aligned}\mathcal{U} &\equiv \{u\in\{0,1\}^n\colon\ r-A(uD_t)\geq 0 \text{ w.p.1}\}\\ &= \{u\in\{0,1\}^n\colon\ r-A^j u_j\geq 0,\ j=1,\dots,n\}.\end{aligned} \tag{5}$$

So, problem (3) can be rewritten as

$$v_t(r)=\max_{u\in\mathcal{U}}\mathbb{E}[f\cdot(uD_t)+v_{t+1}(r-A(uD_t))]. \tag{6}$$

Formulation (3)—or, more precisely, its rewritten form (6)—can be interpreted as a *multistage* $\{0,1\}$ *stochastic linear program*. The objective function in (6) consists of a linear term plus the expected value of the optimal value function of the next stage; also, the constraint set $\mathcal{U}$ is defined by linear inequalities plus the 0-1 constraints. To make the point more precise, let us reformulate the value function in (6). For each $t=1,\dots,\tau$, let $D^t\equiv(D_1,\dots,D_t)$ and $u^t\equiv(u_1,\dots,u_t)$. Also, define $f_t\equiv(f\mathbb{E}D_t)=(fp_t)$. For any fixed r, define

$$\begin{aligned}&V_t(u^{t-1},D^{t-1})\\ &\quad\equiv \max_{u_t\in\{0,1\}^n}\ f_t\cdot u_t+\mathbb{E}[V_{t+1}(u^t,D^t)\mid D^{t-1}]\\ &\qquad\text{s. t. } A^j u_{t,j}\leq r-A(u_1D_1)-\cdots-A(u_{t-1}D_{t-1}),\\ &\qquad\qquad j=1,\dots,n,\end{aligned} \tag{7}$$

where $V_{\tau+1}\equiv 0$. It can be shown by backward induction that $V_t(u^{t-1},D^{t-1})=v_t(r-\sum_{s=1}^{t-1}A(u_sD_s))$ w.p.1. In particular, we have $v_t(r)=V_t(0_{t-1},D^{t-1})$, where 0_{t-1} is an $n\times(t-1)$ matrix of zeroes. Problem (7) is in the standard form of multistage stochastic linear (integer) programs.

One important property of stochastic linear programs is that, when all underlying random variables have finite support, the model can be written as a (typically large) single linear programming problem (LP). This LP has a block-angular structure that is amenable to decomposition methods such as Benders'. This has led to the development of a class of methods, typically called *L-shaped*. For a discussion of multistage methods and models, see Birge and Louveaux (1997), Prékopa (1995), and references therein. Observe that this is exactly the situation in the present case—indeed, each D_t takes on at most $n+1$ values. Of course, we must add the 0-1 constraints to the resulting LP. In §6 we will use the single "large" formulation to obtain structural characteristics of the value function.

4. Mathematical Programming Methods

The recursions in §3 provide *optimal* solutions to the revenue management problem under study. As mentioned above, however, the MDP approach is limited to somewhat small problems. Thus, for larger problems, heuristic methods are needed. In this section, we discuss some standard techniques based on mathematical programming. To this end, we need to introduce a bit more notation. For each $t\in\{0,\dots,\tau\}$, we define the random variables $D_{(t),j}=\sum_{s=1}^t D_{s,j}$, which represent the cumulative demand for class j up to time t. We shall use the convention that an empty sum equals zero.

We start with a heuristic based on linear programs, which has been well studied in the literature. In the following, μ is an n-vector where the jth entry is the expected class j demand over $(0,\tau]$; $\mu_j\equiv\mathbb{E}D_{(\tau),j}$. The linear program is given by

$$\max_x\{f\cdot x\colon\ Ax\leq c,\ 0\leq x\leq\mu\}. \tag{8}$$

We will denote by x^{LP} an optimal solution of the linear program. If the state at time t is $s=(s_1,\dots,s_n)$, the policy π^{LP} specifies $\pi_{t,j}^{\text{LP}}(s)=\mathbb{I}\{s_j\leq x_j^{\text{LP}}-1\}$. Notice that there are no integrality constraints in (8): The definition of the policy π^{LP} does not require that. Observe also that the optimal value ν^{LP} of (8) is *not* the revenue given by π^{LP}, because it does not take into account the actual demand. In fact, ν^{LP} provides an upper bound on the optimal value $v_1(c)$; this follows from Jensen's inequality because the value of the LP is concave in the right-hand side of the constraint $x\leq\mu$ (see Chen, Günther, and Johnson 1998 for details).

The policy π^{LP} requires solving an LP just once, at the beginning of the booking process, and then using the resulting allocation throughout the entire booking horizon. In practice, it is common to solve such an LP at various points in time during the booking process, modifying the LP each time to account for the fact that mean future demand and remaining capacity have changed. The derived policy may then be followed only up to the next re-solve point.

The policy π^{LP} allocates space to the various classes, so policies of this type are also often called *partitioned allocation policies*. Note that optimal policies are generally not of this form. For certain multiclass, single-leg models, it has been shown that there is an optimal "nested" policy. This means that seats that are available to lower-class demand are always available to higher-class demand. It is also common to obtain other types of policies from (8). One such method is to use leg dual values as "bid prices" rather than to create allocations. Key references include Williamson (1992) and Talluri and van Ryzin (1998, 1999). In this paper we do not study nested or bid-price policies.

The LP-based method described above, while simple, has a major drawback: its inability to take randomness into account. The allocations are based solely on the *expected* demand. It is well known that, in general, the introduction of information about the distributions of the underlying random variables leads to richer models—although there are exceptions to this rule (see, e.g., Williamson 1992). This suggests that, to capture the randomness inherent to the problem, one should replace the LP by something that explicitly models randomness. One such formulation is

$$\max_{x\in\mathbb{Z}_+^n} \mathbb{E}[f\cdot\min\{x, D_{(\tau)}\}] \quad \text{subject to} \quad Ax\le c. \tag{9}$$

The above is sometimes called the *probabilistic nonlinear program*. We can use an optimal solution x^{SP} of (9) to obtain a feasible policy for the MDP—simply define $\pi^{\mathrm{SP}}_{t,j}(s)=\mathbb{I}\{s_j\le x^{\mathrm{SP}}_j-1\}$. Observe that the optimal objective value ν^{SP} of (9) is indeed the expected revenue from following π^{SP} (notice the importance of including integrality constraints in (9) for such property). Moreover, because π^{SP} yields a feasible policy for the MDP, it follows that ν^{SP} is a lower bound on $v_1(c)$. It also can be seen that the expected revenue resulting from using the optimal solution x^{LP} of (8) cannot be higher than ν^{SP}.

Problem (9) is a special case of a two-stage stochastic program (SP), falling into the category of *simple integer recourse* models. The advantage of such models is that the resulting problem is separable and therefore can be easily solved using standard methods from stochastic programming (see Chen and Homem-de-Mello 2006 for a detailed discussion); a similar formulation is discussed in de Boer, Freling, and Piersma (2002).

5. Time-Decomposition Methods

We now describe a family of time-decomposition procedures that combine the MDP and the stochastic programming approaches. The basic idea is to split the time horizon into two portions, the first of which corresponds to using heuristic decision rules and the second of which corresponds to decision rules from an optimal policy. The first step in the construction of our solution procedure is to specify the "switch time" $\sigma\in\{1,\dots,\tau+1\}$ that divides the horizon in two.

We then consider policies described by some generic control parameter $\theta\in\Theta_{\sigma-1}$, where $\Theta_{\sigma-1}$ is the set of allowable parameter choices. Below, we will provide specific examples of θ that correspond to specific heuristics. In each period $t\le\sigma-1$, we use decision rule $\delta^\theta_t=(\delta^\theta_{t,1},\dots,\delta^\theta_{t,n})$, where $\delta^\theta_{t,j}$ maps the history of the booking process prior to time t to a decision as to whether we are willing to accept a class j request at time t. The superscript θ reflects that the decision rule depends on θ; the details of this dependence themselves depend on the meaning of the "generic control parameter." We denote the random sales vector in the first time span (periods $1,\dots,\sigma-1$) by $X_{\sigma-1}(\theta)$, where the parenthetical θ indicates the dependence of sales on the choice of θ that determines $(\delta^\theta_1,\dots,\delta^\theta_{\sigma-1})$. This yields the two-stage stochastic optimization problem,

$$\max_{\theta\in\Theta_\sigma}\mathbb{E}[f\cdot X_{\sigma-1}(\theta)+v_\sigma(c-AX_{\sigma-1}(\theta))], \tag{10}$$

where Θ_σ is the set of possible choices of θ. For future developments, let $R_\sigma(\theta)=f\cdot X_{\sigma-1}(\theta)+v_\sigma(c-AX_{\sigma-1}(\theta))$.

For a fixed value of θ, the objective function of (10) is the expected revenue from following the policy

$$\pi^{[\theta,\sigma]}_t=\begin{cases}\delta^\theta_t & \text{if } t<\sigma\\ \pi^*_t & \text{if } t\ge\sigma.\end{cases} \tag{11}$$

Let $\theta^*=\theta^*(\sigma)$ denote a maximizer of (10), and define $\pi^{\mathrm{T}(\sigma)}_t=\pi^{[\theta^*,\sigma]}_t$ so that $\pi^{\mathrm{T}(\sigma)}$ is the policy that yields the optimal objective function value in (10). Let $R^{\mathrm{T}(\sigma)}=R_\sigma(\theta^*)$. At this point, we are ready to describe some particular examples of θ and (10).

5.1. Allocations

Perhaps the simplest example of the formulation involves using allocations prior to σ. Here we have that $\delta^\theta_{t,j}(s)=\mathbb{I}\{s_j\le\theta_j-1\}$, where $\theta=(\theta_1,\dots,\theta_n)\in\Theta_\sigma:=\{\theta\in\mathbb{Z}^n_+\colon A\theta\le c\}$. Here, a choice of θ_j represents the space allocated to class j for the time span $1,\dots,\sigma-1$. Using induction, it follows easily that $X_{\sigma-1}(\theta)=\min\{\theta,D_{(\sigma-1)}\}$, and so formulation (10) can be written succinctly as

$$\begin{aligned}\max_{\theta\in\mathbb{Z}^n_+}\ &\mathbb{E}[f\cdot\min\{\theta,D_{(\sigma-1)}\}\\ &\qquad+v_\sigma(c-A\min\{\theta,D_{(\sigma-1)}\})]\\ &\text{subject to}\quad A\theta\le c.\end{aligned} \tag{12}$$

Owing to the explicit form of (12) we are able to solve it efficiently using variable-sample stochastic optimization techniques as described in §§7 and 8. Furthermore, for the case of the two-leg problems considered in §8, the resulting policies appear to perform well. Finally, observe that if we take $\sigma=\tau+1$, then (12) reduces to (9).

5.2. Bid Prices

To use bid-price controls during the first time span, we consider bid-price vectors of the form $\theta=(\theta_1,\dots,\theta_m)\in\Theta_\sigma:=\mathbb{R}^m$, where θ_i is the bid price of leg i. In this case, we have $\delta^\theta_{t,j}(s)=\mathbb{I}\{f_j\ge A^j\cdot\theta\}\mathbb{I}\{As+A^j\le c\}$ where A^j is the jth column of A. The first indicator function specifies that we should accept a request for product j if its fare f_j exceeds the sum of the bid prices $\sum_{i=1}^{m} a_{ij}\theta_i$ on the legs that j consumes. The second indicator function prevents us from selling beyond capacity.

5.3. Approximate Dynamic Programming

For larger networks, direct application of the general formulation (10) suffers from some of the same shortcomings as the pure MDP approach. In particular, it depends on computation of v_σ, which for larger networks can be essentially impossible even for σ close to the time departure. Computation of v_σ still requires using recursions (3) for the time span after the switch. One way around this is to use some computable approximation for v_σ and then to employ the formulation in repeated-resolve mode described earlier. Namely, periodically resolve the formulation throughout the horizon in order to update the policy in use. This is precisely the approach taken by Bertsimas and de Boer (2005), who use a variant of virtual nesting (see Smith, Leimkuhler, and Darrow 1992) together with a detailed value function approximation procedure. van Ryzin and Vulcano (2006a) have studied properties of a continuous version of the Bertsimas-de Boer formulation.

We conclude this section with the following result, which provides a quantitative relationship among the time-decomposition policies by describing how the expected revenue behaves as a function of the switch time. The result is intuitive—the longer the optimal MDP policy is in effect, the higher expected revenue is. It is evident that the ensuing proposition can be applied for both the allocations and bid prices as described above.

Proposition 2. *Suppose that if $\sigma_1 \le \sigma_2$, then for each $\theta \in \Theta_{\sigma_2-1}$ there exists $\phi \in \Theta_{\sigma_1-1}$ so that $\delta_t^\theta = \delta_t^\phi$ for $t = 1, \ldots, \sigma_1 - 1$. Then the expected revenue $\mathbb{E}[R^{\mathrm{T}(\sigma)}]$ is nonincreasing as a function of switch time σ.*

Proof. Suppose that $\sigma_1 \le \sigma_2$, and let θ^{**} be a maximizer of (10) for $\sigma = \sigma_2$. To simplify notation, let $\bar{\pi} = \pi^{\mathrm{T}(\sigma_2)}$. Let $\phi \in \Theta_{\sigma_1-1}$ satisfy the condition in the supposition (with respect to θ^{**}). Consider the policy $\tilde{\pi} = (\tilde{\pi}_1, \ldots, \tilde{\pi}_\tau)$ defined by

$$\tilde{\pi}_t = \begin{cases} \delta_t^\phi & \text{if } t < \sigma_1 \\ \pi_t^* & \text{if } t \ge \sigma_1. \end{cases}$$

It now follows that $\mathbb{E}R^{\mathrm{T}(\sigma_1)} \ge \mathbb{E}R_{\sigma_1}(\phi)$. Therefore, it suffices to prove that

$$\mathbb{E}R_{\sigma_1}(\phi) \ge \mathbb{E}R^{\mathrm{T}(\sigma_2)} = \mathbb{E}R_{\sigma_2}(\theta^{**}).$$

Recall that $\mathbb{E}R_{\sigma_2}(\theta^{**})$ is the expected revenue from the policy $\bar{\pi} = (\delta_1^{\theta^{**}}, \ldots, \delta_{\sigma_2-1}^{\theta^{**}}, \pi_{\sigma_2}^*, \ldots, \pi_\tau^*)$. By construction $\tilde{\pi}$ and $\bar{\pi}$ are identical over periods $t = 1, \ldots, \sigma_1 - 1$. Hence, we need only show that the expected revenue accrued by $\tilde{\pi}$ during periods $\sigma_1, \ldots, \tau$ is at least as great as that accrued by $\bar{\pi}$ over the same span. For any policy π the expected revenue over periods $\sigma_1, \ldots, \tau$ can be written as

$$\sum_{h \in \mathscr{H}_{\sigma_1}} \mathbb{E}\left[\sum_{t=\sigma_1}^{\tau} g(S_t^\pi, \pi_t(H_t^\pi)) \,\Big|\, H_{\sigma_1}^\pi = h\right] P(H_{\sigma_1}^\pi = h).$$

Because $\tilde{\pi}$ and $\bar{\pi}$ are identical over periods $t = 1, \ldots, \sigma_1 - 1$, we have $P(H_{\sigma_1}^{\tilde{\pi}} = h) = P(H_{\sigma_1}^{\bar{\pi}} = h)$. Moreover, because $\tilde{\pi}$ coincides with π^* over periods $\sigma_1, \ldots, \tau$, it follows that $\tilde{\pi}$ maximizes the quantity $\mathbb{E}[\sum_{t=\sigma_1}^{\tau} g(S_t^\pi, \pi_t(H_t^\pi)) \mid H_{\sigma_1}^\pi = h]$ over all allowable policies. This completes the proof. □

6. Structural Properties

In this section, we return to the MDP formulation of §3. By viewing the problem as a large stochastic program, we prove some structural properties of the value function $v_t(r)$ defined in (6).

For each $t = 1, \ldots, \tau$, let $\omega_{k_1,\ldots,k_t}^t$ be the specific scenario defined by the k_1th outcome at stage 1, the k_2th outcome at stage 2, and so on. The "0th" outcome at stage t means $D_t = 0$, whereas the "jth" outcome at stage t means $D_t = e_j$. Let $u_t(\omega_{k_1,\ldots,k_{t-1}}^{t-1})$ be the vector of decision variables of the optimization problem defined by $V_t(u^{t-1}, D^{t-1})$ in (7), when the realization of demands up to time $t-1$ is given by scenario $\omega_{k_1,\ldots,k_{t-1}}^{t-1}$. By convention, $u_{t,0} \equiv 0$, $A^0 \equiv 0$, and $u_1(\omega^0) \equiv u_1$ (i.e., u_1 does not depend on any particular scenario). Below, all summations are from 0 to n. We can now rewrite (7) for $t = 1$ in a single binary integer program (IP):

$$\begin{aligned}\text{(MSSIP)} \quad \max \quad & (fp_1) \cdot u_1 + \sum_{k_1} p_{1,k_1}[(fp_2) \cdot u_2(\omega_{k_1}^1)] \\ & + \sum_{k_1} p_{1,k_1} \sum_{k_2} p_{2,k_2}[(fp_3) \cdot u_3(\omega_{k_1,k_2}^2)] \\ & + \cdots + \sum_{k_1} p_{1,k_1} \sum_{k_2} p_{2,k_2} \cdots \sum_{k_{\tau-1}} p_{\tau-1,k_{\tau-1}} \\ & \cdot [(fp_\tau) \cdot u_\tau(\omega_{k_1,k_2,\ldots,k_{\tau-1}}^{\tau-1})]\end{aligned}$$

s.t.

$$\begin{aligned} & A^{k_1} u_{1,k_1} \le r, \quad k_1 = 0, \ldots, n \\ & A^{k_1} u_{1,k_1} + A^{k_2} u_{2,k_2}(\omega_{k_1}^1) \le r, \quad k_1, k_2 = 0, \ldots, n \\ & \vdots \\ & A^{k_1} u_{1,k_1} + A^{k_2} u_{2,k_2}(\omega_{k_1}^1) \\ & \quad + \cdots + A^{k_\tau} u_{\tau,k_\tau}(\omega_{k_1,\ldots,k_{\tau-1}}^{\tau-1}) \le r, \\ & \qquad k_1, \ldots, k_\tau = 0, \ldots, n \qquad (13) \\ & u_t(\omega_{k_1,\ldots,k_{t-1}}^{t-1}) \in \{0,1\}^n, \\ & \qquad t = 1, \ldots, \tau,\ k_1, \ldots, k_{t-1} = 0, \ldots, n. \qquad (14)\end{aligned}$$

Notice that each block of inequalities is successively contained in the next; therefore, we can eliminate all but (13). Notice also that the above formulation corresponds to $v_1(r)$. For a general $v_t(r)$, $t \le \tau$, we simply add the constraints $u_s(\omega^{s-1}_{k_1,\ldots,k_{s-1}}) = 0$, $s = 1, \ldots, t-1$; recall that, as discussed earlier, $V_t(0_{t-1}, D^{t-1}) = v_t(r)$.

Although the possibility of reducing the problem to a 0-1 integer linear program seems appealing, in the present case it does not appear to help in terms of yielding an alternative to the iterative solution of optimality equations. The reason is that the number of possible *scenarios* is huge; because the demands in different periods are assumed to be independent, there are $(n+1)^\tau$ overall possibilities. From the above formulation, we can see that we have $(n+1)^\tau - 1$ decision variables in the IP. Moreover, there are $m(n+1)^\tau$ inequalities, plus the 0-1 constraints. So we can see that even with moderate values of n and τ it is impractical to solve the IP exactly, even with the aid of decomposition methods.

In contrast, with the MDP approach, the optimal policy can be found in $O(mn\tau)$ time, although it requires the storage of the optimal policy at every stage for each value of r (which takes $O(n\tau \prod_{i=1}^m c_i)$ space). Nevertheless, the above multistage stochastic integer programming (MSSIP) formulation will allow us to derive some properties of the original problem. Moreover, formulation (MSSIP) can easily be modified to allow dependence of demand across stages. That is, one could, in principle, start from (MSSIP) and model an arbitrary correlation structure. (Dependence can also be incorporated into an MDP by enlarging the state space at the cost of a potential loss of computational tractability.)

Consider the relaxed value function defined by

$$v^{\mathrm{R}}_t(r) = \max_{u \in [0,1]^n} \{\mathbb{E}[f \cdot (uD_t) + v^{\mathrm{R}}_{t+1}(r - A(uD_t))]:\\ P(r - A(uD_t) \ge 0) = 1\}, \qquad (15)$$

with $v^{\mathrm{R}}_{\tau+1}(\cdot) \equiv 0$ (notice that u is no longer required to be integer). It can be seen that $v^{\mathrm{R}}_1(r)$ coincides with the optimal value of (MSSIP) when the integrality constraints in (14) are relaxed to $u_t(\omega^{t-1}_{k_1,\ldots,k_{t-1}}) \in [0,1]^n$. As we did for $v_t(r)$, we set $u_s(\omega^{s-1}_{k_1,\ldots,k_{s-1}}) = 0$, $s = 1, \ldots, t-1$ in the relaxed MSSIP to obtain $v^{\mathrm{R}}_t(r)$. Note that $v^{\mathrm{R}}_t(r)$—and $v_t(r)$—will typically be of interest only when we have independence of demand across periods as in the MDP; otherwise, the conditional expected revenue from time t onward will depend on the entire prior history, not just the remaining capacity at time t (cf. Equation (7)). In preparation for the next result, we say a real-valued function g is nondecreasing if $g(r_1) \ge g(r_2)$ for $r_1 \ge r_2$; we say g is superadditive if $g(r_1) + g(r_2) \le g(r_1 + r_2)$.

Proposition 3. *The relaxed value function* $v^{\mathrm{R}}_t(\cdot)$ *is* concave *and* piecewise linear. *Moreover, both* $v^{\mathrm{R}}_t(\cdot)$ *and* $v_t(\cdot)$ *are* nondecreasing *and* superadditive.

Proof. The result can be proved by (backward) induction, by noticing that (15) has concave objective function and a convex feasibility set [we can rewrite the constraint set of (15) similar to (5)]. Indeed, using that argument, concavity follows even when D_t has continuous distribution. Alternatively, we can use the MSSIP formulation. It is clear that $v^{\mathrm{R}}_t(r)$ is the optimal value of an LP as a function of the right-hand side. Therefore, $v^{\mathrm{R}}_t(r)$ is concave and piecewise linear. The monotonicity and superadditivity properties follow from the fact that $v^{\mathrm{R}}_t(r)$ and $v_t(r)$ are optimal value functions of (integer) LPs. □

Our goal now is to show that $v_t(r)$ and $v^{\mathrm{R}}_t(r)$ are directly related to each other. We hereafter focus on the case with $m = 2$, i.e., two-leg problems. To set the stage, recall that an $m \times n$ matrix Q is *totally unimodular* (TU) if the determinant of each square submatrix of Q is 0, 1, or -1. TU matrices are important because if Q is TU, then the set $\mathcal{P} = \{y \in \mathbb{R}^n_+ : Qy \le b\}$ is integral for all $b \in \mathbb{Z}^m$ for which it is nonempty (Schrijver 1986). "Integral" means that all vertices of the polyhedron defined by $\mathcal{P}$ have integer coordinates. So, if the matrix of constraints in (13) (call it B_τ) is TU, then the solution of the corresponding linear problem, with the integrality constraints relaxed, coincides with the solution of (MSSIP) whenever the right-hand side is integral. This holds even if we allow demands to be correlated across periods, because such a change affects only the objective function and not the constraints. As we show below, in our setting with independent demands we have $v_1(r) = v^{\mathrm{R}}_1(r)$ whenever $r \in \mathbb{Z}^n_+$. In fact, this property holds for $v_t(r)$ and $v^{\mathrm{R}}_t(r)$ for a general $t \le \tau$—just notice that, as remarked in the paragraph following the MSSIP formulation, the constraint matrix for the problem corresponding to v_t (call it B^t_τ) is formed by B_τ and identity matrices. Thus, B^t_τ is TU if B_τ is TU. The theorem below shows that the latter is true; a proof is in the appendix.

Theorem 1. *Consider the function* $v^{\mathrm{R}}_t(r)$ *defined in* (15). *Suppose that the resource consumption matrix A is a* $\{0,1\}$*-matrix with only* two *rows. Then,* $v_t(r) = v^{\mathrm{R}}_t(r)$ *for all* $r \in \mathbb{Z}^n_+$.

The above result shows that, in case of *two-leg* problems with single-seat bookings, the functions $v^{\mathrm{R}}_t(r)$ and $v_t(r)$ coincide when $r \in \mathbb{Z}^n_+$. This is an important property, as it implies (from Proposition 3) that, for example, $v_t(r_1+1, r_2) - v_t(r_1, r_2)$ is decreasing in r_1 for each fixed t and r_2. From this it follows that there is a optimal policy with capacity thresholds for the local itineraries. A similar result can be found in Theorem 3.3 of You (1999), which states that the value function for two-leg flights is componentwise concave. We feel that our proof, based on the formulation of the problem as an MSSIP, is of independent interest because it uses completely different mathematical techniques. Notice also that our result shows a bit more than componentwise concavity—it actually shows that $v_t(r)$ coincides with a concave function on the points of its domain. For integer Δ, it follows that $v_t(r_1+1, r_1+\Delta+1) - v_t(r_1, r_1+\Delta)$ is decreasing in r_1. Therefore, if a particular "through" product is open when the remaining capacity is $(r_1, r_1+\Delta)$ at time t, then it also open for remaining capacity $(r_1', r_1'+\Delta)$ at time t if $r_1' \geq r_1$. This allows one to reduce the number of actions that must be considered when computing an optimal policy using backward induction. One might also ask whether the relaxed function $v_t^{\mathrm{R}}(r)$ belongs to the class of so-called M-concave functions, in which case $v_t(r)$ can be considered a discrete concave function (simultaneously in both of its arguments) in a very precise sense; see Murota (2003) for a comprehensive discussion on the concepts of discrete convex analysis. Such an investigation lies outside the scope of this work and is left for future research.

Incidentally, our argument can be easily adapted to yield an alternative proof of concavity of one-leg problems—just notice that, in such case, the vectors $a_0, \ldots, a_3$ defined in the proof of Theorem 1 are replaced by $a_0 = 0$, $a_1 = 1$. The proof then becomes much easier, because there is no need to analyze the case where a column has two nonzero entries.

It is natural to ask whether Theorem 1 is valid for problems with more than two legs. Unfortunately, the answer is negative—indeed, the proof of Theorem 1 hinges on the fact that the matrix A has only two rows. Example 1 shows that for a three-leg problem, the value function need not be componentwise concave.

Example 1. Suppose that the three legs are labeled 1, 2, and 3. There are three customer classes, also labeled 1, 2, and 3. Class 1 requires a seat on both legs 1 and 2, class 2 requires a seat on legs 1 and 3, and class 3 requires a seat on both legs 2 and 3. Therefore, the resource consumption matrix is composed of three columns: $[1\ 1\ 0]'$, $[1\ 0\ 1]'$, and $[0\ 1\ 1]'$. This example describes a situation where there are three cities; A, B, and C so that leg 1 goes from A to B, leg 2 goes from B to C, and leg 3 goes from C to A. Class 1 flies from A to C via B, class 2 flies from C to B via A, and class 3 flies from B to A via C. Hence, there is a cycle, of sorts, in the network. (Note we are assuming there is no leg from, say, B to A.)

Assume that each customer pays a fare of \$100. Suppose that $\tau = 6$ and that the arrival process is deterministic, with a class 1 arrival in periods 1 and 2, a class 2 arrival in periods 3 and 4, and a class 3 arrival in periods 5 and 6. Recall that $v_1(r_1, r_2, r_3)$ denotes the maximum expected revenue from $t=1$ onward, when remaining capacity on legs 1, 2, and 3 is, respectively, r_1, r_2, and r_3. Simple calculations now show that $v_1(2,2,0) = 200$, $v_1(2,2,1) = 200$, and $v_1(2,2,2) = 300$. Therefore, the value function is not componentwise concave.

The failure of componentwise concavity for networks with three or more legs is significant. Without this, one cannot show the existence of threshold-type optimal policies for the MDP. Such threshold-type optimal policies can be employed to save computational effort and storage space for one- and two-leg problems, but evidently, they may not exist for larger networks without imposition of additional assumptions. Notice also that, even when v_t in (6) is componentwise concave, such a property does not imply the componentwise concavity of the objective function in (12), because the term $v_\sigma(c - A\min\{x, D_{(\sigma-1)}\})$ in (12) consists of a composition of v_σ with a componentwise-convex mapping. Nevertheless, we can derive heuristic solution methods, described in the next section.

A closer look at Example 1 shows that the network discussed there has a certain type of "cycle." The absence of such cycles may be a sufficient condition for componentwise concavity of the value function v_t—indeed, we have not been able to find examples of acyclic networks where v_t is not componentwise concave (by "acyclic" we mean a network without directed or undirected cycles). Observe that a network with the same structure as in Example 1, but where there is no demand for two-leg flights, can naturally be separated into three one-leg problems, so componentwise concavity holds in that case. Hence, absence of cycles is not a necessary condition for componentwise concavity.

As the next example shows, there exist simple acyclic three-leg networks where the matrix of constraints B_τ in (13) fails to be TU. Thus, it appears that the technique used in the proof of Theorem 1—using total unimodularity of the constraint matrix to show that there exists an integral optimal solution—cannot be extended to more general cases.

Example 2. We now show that the constraint matrix for larger networks need not be totally unimodular, even when cycles like those in Example 1 are not present. Consider the case of a three-leg line-network, with leg 1 connecting city A to city B, leg 2 connecting city B to city C, and leg 3 connecting city C to city D. It is easy to see that the columns of A are duplicates of the columns below:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Therefore, the following matrix is a submatrix of B_τ (recall that B_τ is the matrix of constraints in (13)):

$$Q = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Let Q_1 be a submatrix of Q formed by the first, third, and fifth rows. Then, $\det(Q_1) = 2$, so B_τ is not TU.

7. Solution Procedures

We now discuss some procedures to solve (12). In its stated form, (12) is an integer nonlinear stochastic optimization problem and as such is notoriously difficult to solve. However, because the purpose of (12) is to provide a heuristic allocation, it is reasonable to consider approximations to that problem. In particular, we will relax the integrality constraint on the allocations.

To put the problem into the standard framework of stochastic optimization, let $G(x, D)$ denote the function

$$G(x, D) = f \cdot \min\{x, D_{(\sigma-1)}\} + v_\sigma(\lfloor c - A\min\{x, D_{(\sigma-1)}\}\rfloor), \qquad (16)$$

where x represents the partitioned allocation, and let $g(x) = \mathbb{E}[G(x, D)]$. We can now formulate the relaxed version of (12) as

$$\max\{g(x)\colon Ax \le c,\ x \ge 0\}. \qquad (17)$$

Notice that we only relax the constraint $x \in \mathbb{Z}_+^n$; we still work with the "non-relaxed" function v_σ since it is hard to evaluate the relaxed version v_σ^{R}.

One of the most studied techniques for stochastic optimization problems is the *Stochastic Approximation* (SA) method (see, e.g., Kushner and Yin 1997). Although widely applicable, this procedure has some drawbacks, especially the slow convergence and the difficulty in implementing good stopping criteria. A somewhat different approach is based on *Monte Carlo*-type methods. The basic idea is to replace the expected value $\mathbb{E}[G(x)]$ with the approximation $g_N(x) \equiv N^{-1}\sum_{i=1}^N G(x, D^i)$, where $D^1, \ldots, D^N$ are i.i.d. samples of $D_{(\sigma-1)}$, and then solve the approximating problem

$$\min_{x\in X} g_N(x). \qquad (18)$$

The rationale is that, by the strong law of large numbers, $g_N(x) \to g(x)$ as $N \to \infty$ with probability one, and thus (18) should provide a good approximation to (17) for N large. This idea, of course, is not new, and has been an object of study in different areas for many years. The convergence properties of such procedures are well established (see, for instance, Shapiro 2003 for a compilation of results).

An alternative to the basic Monte Carlo idea is proposed by Shapiro and Homem-de-Mello (1998). Instead of fixing a sample from the beginning and then minimizing the resulting deterministic function, they use *different samples* along the algorithm. In our context, this means that at the kth iteration of the algorithm we use the approximating function

$$\hat{g}_k(x) := \frac{G(x, D^{k,1}) + \cdots + G(x, D^{k,N_k})}{N_k}, \qquad (19)$$

where $D^{k,1}, \ldots, D^{k,N_k}$ are i.i.d. samples of $D_{(\sigma-1)}$, drawn at iteration k. In reality, there is no need to resample at every iteration—we can keep the same sample for a few iterations, particularly at the early stages of the process.

There are a few advantages with the above approach: One is that the sample sizes N_k can increase along the algorithm, so that sampling effort is not wasted in initial iterations of the process. Also, because the estimates at different iterations are independent, one can perform *statistical tests* to compare the estimates, which in turn can lead to *stopping criteria* for the algorithm. Moreover, the resampling procedure tends to minimize the chance of obtaining a bad solution because of a "bad" sample path, which can occur in (18) if N is not large enough. The disadvantage is that the function being optimized changes at every iteration—which means that standard deterministic optimization packages cannot be used. Rather, the code must be adapted to incorporate sampling in it. In Shapiro and Homem-de-Mello (1998), one such code, suitable for convex problems with linear constraints, is presented. A similar method is used by Homem-de-Mello, Shapiro, and Spearman (1999), who successfully apply these techniques to the optimization of release times of jobs in a single-line production environment.

Our procedure to solve (17) is based on such variable-sample methods. The procedure incorporates sampling and statistical techniques into a nonlinear programming method that combines line search with projection of the gradient of the objective function onto the null space of active constraints (see, for example, Bertsekas 1999 for a thorough discussion of the latter topic). Notice that, strictly speaking, the objective function in (17) is not differentiable because of the rounding and min operations. In what follows, the operator ∇ denotes an approximate gradient (cf. §7.1 below).

Because the optimization method we implement is very similar to the one used in Homem-de-Mello, Shapiro, and Spearman (1999), we will not repeat its description here in detail; refer to that paper for a full discussion. For the sake of completeness, we present an outline of the algorithm:

1. Choose a (small) sample size $N = N_1$ and initial value x^1 of allocations; set $k := 1$.
2. For the current iterate k, generate an i.i.d. sample of size N_k of vectors $D^1, \ldots, D^{N_k}$; compute the estimators $\hat{g}_k(x^k)$ (cf. (19)) and $\nabla\hat{g}_k(x^k)$ (cf. §7.1 below).
3. Compute an ascent direction δ_k by projecting the estimator $\nabla\hat{g}_k(x^k)$ onto an appropriate space and compute a stepsize α_k by a (crude) line search.
4. Set $x^{k+1} := x^k + \alpha_k\delta_k$ and compute $\hat{g}_k(x^{k+1})$ and $\nabla\hat{g}_k(x^{k+1})$ using the same demand-vector sample $D^1, \ldots, D^{N_k}$.
5. If the increase $\hat{g}_k(x^{k+1}) - \hat{g}_k(x^k)$ is significantly large, then go back to Step 3 with $k \leftarrow k+1$.
6. Otherwise, compute a new sample size N_{k+1} and generate a new sample $\widetilde{D}^1, \ldots, \widetilde{D}^{N_{k+1}}$. Compute $\hat{g}_{k+1}(x^{k+1})$, $\nabla\hat{g}_{k+1}(x^{k+1})$ based on that sample.
7. Test statistical stopping criteria based on the difference $\hat{g}_{k+1}(x^{k+1}) - \hat{g}_k(x^k)$ and on the estimator $\nabla\hat{g}_{k+1}(x^{k+1})$. If none of them is satisfied, go to Step 3 with $k \leftarrow k+1$. Otherwise, STOP.

Notice that, due to the lack of convexity/concavity in the present case, the procedure may end up at a stationary but perhaps suboptimal point. Thus, it is clear that the method provides only a heuristic approximation to (12). Nevertheless, the numerical results in §8 (and others not shown) suggest that the solutions obtained with such heuristics typically are satisfactory.

7.1. Computing Derivatives

Implementation of the algorithm described above requires calculation of an "approximate gradient" ∇g of the objective function $g(x)$. Consider the function $G^{\mathrm{R}}(x, D)$, defined as G in (16), but with the relaxed function v_σ^{R} defined in (15) in place of v_σ. Let $\partial G^{\mathrm{R}}(x, D)$ denote the *generalized gradient* of $G^{\mathrm{R}}(\cdot, D)$ at x, and let $\nabla G^{\mathrm{R}}(x, D)$ denote an arbitrary element of $\partial G^{\mathrm{R}}(x, D)$. The generalized gradient can be viewed as an extension of the concept of a subdifferential set, standard in convex analysis, to cases when the function is not convex (for details, see Clarke 1983). By writing $\min\{x_j, D_{(\sigma-1),j}\} = x_j\mathbb{I}\{x_j < D_{(\sigma-1),j}\} + D_{(\sigma-1),j}\mathbb{I}\{x_j \geq D_{(\sigma-1),j}\}$ we have that

$$[\nabla G^{\mathrm{R}}(x, D)]_j = f_j\mathbb{I}\{x_j < D_{(\sigma-1),j}\} - A_j^T\mathbb{I}\{x_j < D_{(\sigma-1),j}\} \cdot \nabla v_\sigma^{\mathrm{R}}(c - A\min\{x, D_{(\sigma-1)}\}),$$

where A_j is the jth column of A. Now, finiteness of support of $D_{(\sigma-1)}$ implies that we can exchange the derivative and expectation operators, i.e., $\partial g^{\mathrm{R}}(x) = \mathbb{E}[\partial G^{\mathrm{R}}(x, D)]$.

Using the above result, we can estimate ∇g^{R} via sampling, provided we can estimate $\nabla v_\sigma^{\mathrm{R}}$. The latter quantity can be approximated by *two-sided* finite differences, that is,

$$[\nabla v_\sigma^{\mathrm{R}}(r)]_j \approx \frac{v_\sigma^{\mathrm{R}}(r + e_j) - v_\sigma^{\mathrm{R}}(r - e_j)}{2}. \tag{20}$$

Here, e_j is the vector with 1 in the jth component and zeroes otherwise. The above approximation must be adjusted for points at the boundary of the domain, that is, when $r_j = 0$ or $r_j = c_j$. In those cases, a one-sided finite difference is used. Finally, by replacing v_σ^{R} by v_σ in (20) and rounding down the arguments of v_σ to get integer values, we obtain an estimate for ∇g that can be used in the algorithm. We use this technique to obtain the derivative estimates in the algorithm described above. A similar procedure is used by Bertsimas and de Boer (2005), although they work exclusively with one-sided derivatives.

7.2. The State-Space Decomposition Approach

The sampling-based approach described above can, in principle, be applied to a general problem of the form (16)–(17). However, evaluation of the value function may become computationally prohibitive. To address this issue, we propose now an approach whereby the network is decomposed into a collection of *two-leg* problems. The procedure we describe below borrows ideas from the method that partitions the network into *one-leg* problems, based on the so-called displacement-adjusted revenue (see §3.4 of Talluri and van Ryzin 2004b for details and references).

Let $\theta = (\theta_1, \ldots, \theta_m) \in \mathbb{R}^m$ be a vector of bid prices for each leg (see §5). For convenience of notation, we will denote by A^j the set of legs used by product j; that is, $i \in A^j$ if and only if $a_{ij} = 1$. For a product j and a subset of legs $q \subseteq \{1, \ldots, m\}$ with $A^j \cap q \neq \varnothing$, the *displacement-adjusted fare* for j on q is defined as

$$\varphi_j^q := \Bigl[f_j - \sum_{l \in A^j \setminus q} \theta_l\Bigr]^+. \tag{21}$$

The displacement-adjusted fare of product j on q discounts the fare of j by the bid prices of the legs other than q used by j.

Given legs q, we consider a *q-reduced problem* with only legs q and only those products j for which $A^j \cap q \neq \varnothing$. For a product j included in the q-reduced problem, the q-reduced problem replaces A^j by $A^j \cap q$ and replaces f_j by φ_j^q. Note that $\varphi_j^q = f_j$ if $A^j \subseteq q$; that is, the fare remains unchanged for products that use legs only in q.

Consider now a partition Q of the legs of the network into sets of one or two elements, and let

$q_1, \ldots, q_K$ denote these sets. Given the partition $Q = \{q_1, \ldots, q_K\}$, we then solve the collection of q_k-reduced problems; $k = 1, \ldots, K$. Each reduced problem can be solved using the time-decomposition approach that uses simulation-based optimization and Markov decision processes, as described earlier in this section. (In fact, one could use other approaches as well.) More specifically, let σ denote a switch time that is fixed for all reduced problems. Then, for each $k \in \{1, \ldots, K\}$ we solve (17) for the reduced problem. Let x^{*k} and π^{*k} denote, respectively, the resulting vector of allocations and the optimal MDP policy (from time σ onward) for the q_k-reduced problem.

The above procedure yields a collection of K booking control policies. Clearly, these policies may give conflicting results for the itineraries in the original problem that have been broken down into different elements of the partition. Thus, it is important to define rules that combine the different policies into a single "reconstituted" one that ultimately determines whether a request should be accepted in the original network problem.

For given Q, such a reconstituted policy can be described by decision rules $\delta_t^Q = (\delta_{t,j}^Q\colon\ j = 1, \ldots, n)$ that map the history of the booking process (for the original problem) prior to time t to a decision as to whether we are willing to accept a class j request at time t. We define $\bar{\delta}^Q$ and $\underline{\delta}^Q$ as

$$\bar{\delta}_{t,j}^Q(s) := \mathbb{I}\left\{s_j \le \left(\max_{k:\, j \in q_k} x_j^{*k}\right) - 1\right\} \mathbb{I}\{As + A^j \le c\} \tag{22}$$

$$\underline{\delta}_{t,j}^Q(s) := \mathbb{I}\left\{s_j \le \left(\min_{k:\, j \in q_k} x_j^{*k}\right) - 1\right\} \mathbb{I}\{As + A^j \le c\}. \tag{23}$$

Notice that the $\bar{\delta}^Q$ defined in (22) takes the *maximum* allocation to itinerary j among the K subproblems as the reconstituted allocation. The term $\mathbb{I}\{As + A^j \le c\}$ ensures that the booking is rejected if there is not enough capacity. The policy $\underline{\delta}^Q$ defined in (23) works analogously, instead using the *minimum* allocation. These decisions rules are used between times 1 and $\sigma - 1$.

Similarly, define $\bar{\pi}^Q$ and $\underline{\pi}^Q$ for the MDP portion of the horizon (i.e., times σ though τ) as

$$\bar{\pi}_{t,j}^Q(s) := \max_{k:\, j \in q_k} \pi_{t,j}^{*k}(s) \tag{24}$$

$$\underline{\pi}_{t,j}^Q(s) := \min_{k:\, j \in q_k} \pi_{t,j}^{*k}(s). \tag{25}$$

The policy $\bar{\pi}^Q$ defined in (24) accepts a request for itinerary j if and only if that request is accepted in *any* of the reduced problems where j is involved; likewise, the policy $\underline{\pi}^Q$ defined in (25) accepts a request for itinerary j if and only if that request is accepted in *all* reduced problems where j is involved. It is clear that the policies $\bar{\delta}^Q$, $\underline{\delta}^Q$, $\bar{\pi}^Q$, and $\underline{\pi}^Q$ can be combined in four different ways to yield an overall policy of the form (11); we will call those combinations "max-any," "min-any," "max-all," and "min-all," the meaning of each one being clear in light of the above discussion. In §8 we present numerical results for one of these cases.

The state-space decomposition approach described above is valid for any partition Q; however, certain choices of Q may behave better (in terms of overall revenue) than others. We propose the following heuristic procedure to choose a partition. Given a partition $Q = \{q_1, \ldots, q_K\}$ with $|q_k| \le 2$ for all $k = 1, \ldots, K$, define for each product j the quantity

$$\nu_j(Q) := \left|\{k\colon\ |A^j \cap q_k| = 2\}\right|, \tag{26}$$

and let

$$N(Q) := \{j\colon\ \nu_j(Q) \ge 1\}. \tag{27}$$

We define the value of a partition Q as

$$V(Q) := \sum_{j \in N(Q)} \nu_j(Q) f_j \mathbb{E} D_{(\tau),\, j}.$$

Then, the partition we choose to decompose the problem is the partition Q^* that *maximizes* the function $V(\cdot)$ over all partitions $Q = \{q_1, \ldots, q_K\}$ $(K = 1, 2, \ldots)$ with $|q_k| \le 2$ for all $k = 1, \ldots, K$. The rationale for our choice of Q^* is the following. The set $N(Q)$ contains those products that use both legs of some two-leg component of the partition Q. That is, a product j is in $N(Q)$ if and only if there is a component of Q, say $\{i_1, i_2\}$, such that j uses both legs i_1 and i_2. Because we are maximizing $V(Q)$, we favor the partitions Q that do not break down itineraries of products with a high value of demand-weighted fare—the latter quantity is perceived as a measure of the importance of that product. Hence, the "important" itineraries will tend not to be split. For example, suppose that the network is entirely *separable* into two-leg portions, in the sense that no itinerary requires more than two legs and no two-leg itineraries partially overlap. In that case, of course, there is an obvious partition Q^o. From (26) and (27) we immediately see that $N(Q^o)$ contains all two-leg products and hence Q^o maximizes V. The choice of Q^* as a maximizer of V has worked well in our experiments, as reported in §8.5.

It is worth noting that we need not calculate the value of all partitions with $|q_k| \le 2$ to find the maximizer; from (26)–(27) we see that the value of a partition with $q_k = \{i_1, i_2\}$ cannot be lower than that of a partition with $q_l = \{i_1\}$ and $q_j = \{i_2\}$. For the same reason, we need not consider two-leg components for which no product uses both legs.

Efficient algorithms to maximize $V(\cdot)$ can be developed for problems in which there are only one- and two-leg itineraries. In that case, consider a weighted graph in which the vertices correspond to the legs of the flight network, and the weight associated with an edge connecting vertices u and v is the sum of the $f_j \mathbb{E} D_{(\tau),j}$ over those products j that use both legs u and v. Then, maximizing $V(\cdot)$ is equivalent to finding a maximal weighted matching in the graph, a problem for which efficient algorithms exist (see, e.g., Galil, Micali, and Gabow 1986).

8. Numerical Experiments

In this section we present some numerical examples to compare the time-decomposition policy with allocations and some of its simpler "relatives" with certain other baseline heuristics as well as with optimal policies. In our tests (all of which use $\tau = 1{,}000$), we consider three different types of time-decomposition policies. The simplest, which we refer to as LP-MDP, involves solving the linear program (8) at the beginning of the time horizon (at $t = 0$) and following the resulting allocations up to a predetermined switch time, at which point we begin using the optimal policy. This can be viewed as the simplest way to implement the idea of using math programming far from time of departure, and switching to an optimal decision rule near time of departure. An optimal policy was computed using the standard backward induction algorithm and stored in a large lookup table.

The second type of time-decomposition policy that we consider is again based on the linear program. The policy LP(200)-MDP involves solving the LP at time 0 and following the resulting allocation for 200 time periods, at which point the LP is re-solved with capacity replaced by remaining capacity and expected demand replaced by expected remaining demand. The resulting policy is then followed until time 400. The procedure of successively re-solving and myopically applying the allocations is repeated every 200 time units, until the switch time σ is reached, at which point we commence using optimal decision rules. Note that although it is reasonable to expect these policies to yield higher expected revenue than the policies LP-MDP, this need not be the case (see Cooper 2002). We also consider the policies LP(25)-MDP and LP(50)-MDP, which instead re-solve every 25 and 50 time periods, respectively, prior to switching to the MDP.

The third type we test is an approximation from (17) to $\pi^{\mathrm{T}(\sigma)}$, the policy defined just after (11) with θ as in (12). Observe that for a particular switch time, $\pi^{\mathrm{T}(\sigma)}$ necessarily yields higher expected revenue than does the policy from LP-MDP, because the policy followed by LP-MDP is a feasible solution to (12). This is not the case for LP(25)-MDP, LP(50)-MDP, or LP(200)-MDP, because (12) does not consider policies that modify their allocations prior to the switch time. Our numerical results suggest that the time-decomposition policies with allocations appear to perform well in the two-leg problems we considered. For problems with more products, it may be more important to consider nested methods.

In the tables and charts, SP-MDP is used to denote the policy obtained from the variable-sample stochastic optimization routine that we employ to solve (17), which approximates (12). Because this solution is not guaranteed to be optimal for (12), it could possibly be the case that the expected revenue obtained from following it will be worse than that from LP-MDP. However, we have not observed this phenomenon in any examples. In addition, because the policy SP-MDP depends on the (random) outcome of a simulation-based procedure, it is necessary to broaden the definition of policy to include *randomized policies* (for further discussion, see Puterman 1994). Note that this is necessitated not by the formulation itself, but rather by the use of a simulation-based solution approach.

The algorithm for computing the policies used to generate the SP-MDP policies requires samples from $D_{(\sigma-1)}$. To simplify the algorithm, rather than sampling from the distribution of $D_{(\sigma-1)}$, we instead drew samples from a Poisson distribution with mean $\mathbb{E} D_{(\sigma-1)}$. Formal justification for this can be found in Chapter 10 of Ross (1996). As a practical matter, our experience with a variety of problem instances suggests that this method does work well.

8.1. Simulation Methodology

The following methodology was applied to the two-leg examples described in §§8.2 and 8.3. For each fixed switch time σ, we simulated the five policies discussed above by generating a realization of the actual Bernoulli arrival process described in §3. We then computed the average revenue for each policy over 1,000 independent replications. For each of the 1,000 simulation replications, common random numbers were used to generate the arrival process for each policy; hence, for a fixed σ, all policies under consideration see the same stream of demands.

The process above was repeated for $\sigma = 1$, 201, 401, 601, 801, and 1,001 to understand the behavior of the policies as a function of the switch time. We used *independent* random numbers for the simulations carried out with different values of σ; for example, the demand stream for evaluating the policies with $\sigma = 201$ was different from (and independent of) that for evaluating the policies with $\sigma = 401$.

Results are presented in Tables 1–4, and depicted graphically in Figures 1 and 3. In Tables 1 and 3, there are three numbers in each cell. The top left number is an estimate of the expected revenue from following the policy determined by the column with the switch time determined by the row. This estimate is the sample average of 1,000 simulation runs of the policy in

Table 1 **Mean Revenues and Corresponding Percentages of the Simulated Optimal Value for the Various Values of Switch Time and Various Methods (for the Example in §8.2)**

σ	LP-MDP	LP(200)-MDP	LP(50)-MDP	LP(25)-MDP	SP-MDP
1	49,906.03, 284.52 100%				
201	49,836.23, 279.31 99.9%	49,836.23, 279.31 99.9%	49,824.05, 278.48 99.8%	49,823.25, 278.35 99.8%	49,834.80, 279.39 99.9%
401	49,630.15, 277.72 99.5%	49,621.13, 277.72 99.4%	49,605.63, 276.56 99.4%	49,606.25, 276.58 99.4%	49,630.23, 277.89 99.5%
601	49,645.75, 279.28 99.5%	49,639.75, 279.28 99.5%	49,628.98, 277.29 99.4%	49,628.70, 277.37 99.4%	49,652.85, 280.35 99.5%
801	49,162.78, 254.82 98.5%	49,327.15, 254.82 98.8%	49,387.98, 259.97 99.0%	49,354.73, 258.77 98.9%	49,372.15, 279.73 98.9%
1,001	46,743.98, 177.40 93.7%	48,340.28, 252.64 96.9%	48,971.40, 262.59 98.1%	48,910.63, 263.29 98.0%	48,512.18, 291.58 97.2%

question. The top right number is the half-width of a 98% confidence interval for the expected revenue. By Bonferroni's inequality (see, e.g., Law and Kelton 2000), it follows that the *overall* confidence level on the intervals obtained for all five policies for a particular switch time σ is at least 90% (this means the a priori probability is at least 0.90 that *each* of the five intervals simultaneously holds the corresponding true expected revenue). The bottom number is a percentage ratio between the expected revenue in that cell and the expected revenue obtained with $\sigma = 1$, i.e., estimated by simulating the pure MDP policy.

The purpose of Tables 2 and 4 is to compare each LP-based policy with the SP-MDP policy. There are two numbers in each cell. The first number is an estimate of $\mathbb{E}[R^{\text{SP-MDP}} - R^{\text{LP(n)-MDP}}]$, where $R^{\text{SP-MDP}}$ is the revenue obtained from the SP-MDP policy and $R^{\text{LP(n)-MDP}}$ is the revenue for the corresponding LP-based policy (both for the value of σ specified by the row). The second number is the half-width of a 95% confidence interval for the difference $\mathbb{E}[R^{\text{SP-MDP}} - R^{\text{LP(n)-MDP}}]$. Again, Bonferroni's inequality implies that the overall confidence on the four intervals obtained for each fixed switch time is at least 80%.

The above confidence intervals for the differences should be interpreted as follows. If zero is contained in such interval, then for the switch time in question, there is no indication that the policy SP-MDP gives different revenue than the corresponding LP-based policy. Otherwise, if the calculated interval is to the right of zero, then there is indication that the SP-MDP policy performs better than the corresponding LP-based policy for that particular switch time. Such cases are signaled with "+" in the difference tables. Similarly, a "−" is displayed when the confidence interval falls to the left of zero, which indicates superiority of the corresponding LP-based policy over the SP-MDP policy for that switch time.

Table 2 **Differences in Revenue (for the Example in §8.2)**

σ	LP-MDP	LP(200)-MDP	LP(50)-MDP	LP(25)-MDP
201	−1.43, 1.62	−1.43, 1.62	10.75, 4.83 +	11.55, 4.79 +
401	0.08, 1.94	9.10, 4.55 +	24.60, 6.23 +	23.98, 6.54 +
601	7.10, 6.45 +	13.10, 10.64 +	23.88, 12.46 +	24.15, 12.82 +
801	209.38, 51.66 +	45.00, 53.37	−15.83, 49.56	17.43, 51.24
1,001	1,768.20, 137.14 +	171.90, 96.12 +	−459.23, 76.14 −	−398.45, 75.58 −

8.2. A Two-Leg Example

Consider a two-leg problem with capacity of 150 seats on each leg. There are three possible itineraries (numbered 0 for the through itinerary that uses both legs, 1 for the local itinerary that uses only leg 1, and 2 for the local itinerary that uses only leg 2), each with three fare categories. Thus, in this problem there are $n = 9$ itinerary-fare classes. The through itinerary fares are \$600, \$100, and \$75. The fares for each local itinerary are \$500, \$75, and \$50. There are $\tau = 1{,}000$ time periods. The arrival probabilities for the highest (Category 1), second-highest (Category 2), and lowest (Category 3) category on each itinerary are given by $P(D_{t,(1,l)} = 1) = \alpha\beta_{(1,l)} \sin(t\pi/2\tau)$, $P(D_{t,(2,l)} = 1) = \alpha\beta_{(2,l)} \sin(t\pi/\tau)$, and $P(D_{t,(3,l)} = 1) = \alpha\beta_{(3,l)} \sin((\pi/2) + (t\pi/2)\tau)$, where $D_{t,(k,l)}$ is the demand for category k, itinerary l at time t. Observe that this means that high-category (Category 1) expected demand per time unit grows as we get closer to time of departure, whereas low-category (Category 3) expected demand per time unit falls as we approach time of departure. Category 2 expected demand per time unit peaks in the middle of the booking process. In this example, we use $\alpha = 0.46$, $\beta_{(1,0)} = 0.05$, $\beta_{(2,0)} = 0.07$, $\beta_{(3,0)} = 0.08$, and $\beta_{(1,1)} = \beta_{(1,2)} = 0.10$, $\beta_{(2,1)} = \beta_{(2,2)} = 0.12$, $\beta_{(3,1)} = \beta_{(3,2)} = 0.18$. This yields a total expected demand of approximately 175 seats per leg.

Results are presented in Tables 1 and 2 and Figure 1. For this example, exact computation for the MDP shows that the maximum expected revenue is $v_1(c) = 49{,}737.23$, which does lie within the confidence interval shown in the $\sigma = 1$ row. Examination

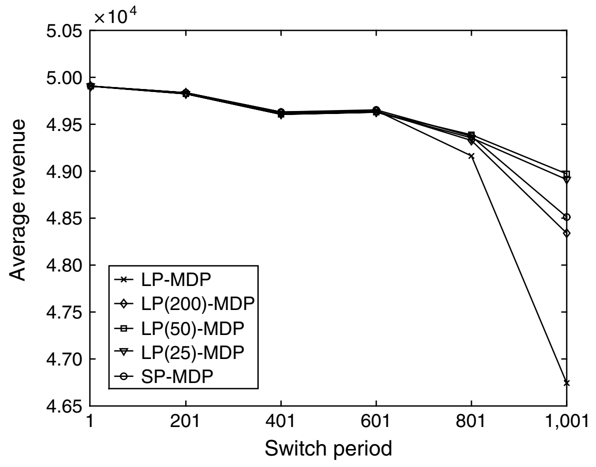

Figure 1 **Mean Revenue as a Function of Switch Time (for the Example in §8.2)**

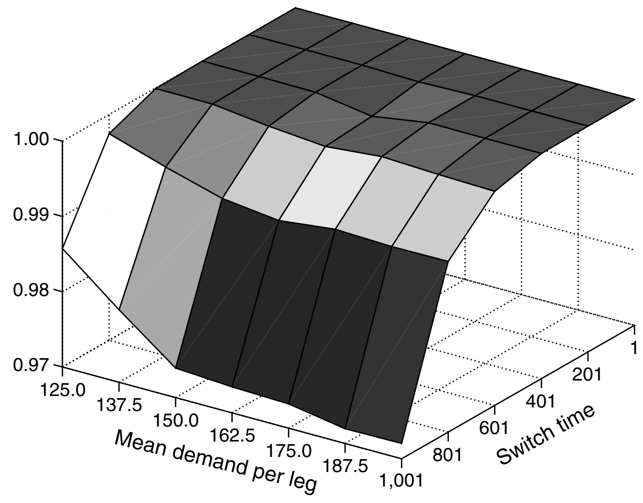

Figure 2 **Ratio of Expected Revenue from the Time-Decomposition Heuristic to Optimal Expected Revenue, Plotted Against Switch Time and Mean Demand per Leg**

of Tables 1 and 2 shows that SP-MDP appears to be slightly stronger than the other heuristics, except when there is frequent reoptimization for $\sigma = 1{,}001$. In that case, frequent reoptimization of the LP gave a better policy. Although the difference in performance is statistically significant in many of the cases shown, the estimated differences in revenues are indeed quite small. It is interesting to point out that all of the switching heuristics can obtain "most" of the benefit from the MDP by using optimal decision rules in less than half of the booking horizon. Even when switching with only one-fifth of the horizon remaining, three of the four heuristics performed quite well, and the fourth (LP-MDP) performed reasonably well.

To examine the effect of demand-capacity ratio, we also considered versions of the problem with mean demand of 125, 137.5, 150, 162.5, 187.5, and 200 per leg. These problems were identical to that described above, except that we took α to be approximately 0.33, 0.36, 0.39, 0.42, 0.49, and 0.53, respectively. For different values of the switch time σ, Figure 2 shows the estimated ratio of the expected revenue from following the time-decomposition heuristic to the optimal expected revenue. The figure also shows the case with mean demand of 175 per leg, which was discussed above. From Table 1 we see that for mean demand of 175 per leg, using $\sigma = 1{,}001$ for SP-MDP yields expected revenue that is roughly 97.2% of optimal. Likewise, for $\sigma = 801$, SP-MDP yields expected revenue that is roughly 98.9% of optimal. Examining Figure 2, we can see that this seemed to be the case so long as mean demand per leg was not too small. If mean demand per leg was small (125), then even $\sigma = 1{,}001$ gave an average revenue that was approximately 98.6% of optimal, reflecting the fact that in such situations, any policy that accepts almost all booking requests will perform very well.

8.3. Another Two-Leg Example

We now briefly describe a two-leg example where the SP-MDP is the clear winner among the heuristics. Consider a two-leg problem with three fare categories, 150 seats, and $\tau = 1{,}000$ (so again there are $n = 9$ itinerary-fare classes). The through fares are \$1,000, \$900, and \$100, and the local fares are \$800, \$700, and \$50. All Category 2 and 3 itineraries have identical constant arrival probabilities of 0.0656 from $t = 1$ through 800, and zero thereafter. Up to time 800 there are no Category 1 arrivals on any itinerary. After time 800, the Category 1 arrival probabilities are constant at 0.0375. The expected total demand per leg is therefore 225. In Tables 3 and 4 and Figure 3, we see that SP-MDP performs significantly better than all the other heuristics, and nearly as well as the pure MDP, which has a value of 142,344.7. In this example, the difference between high and low fares is quite high; such cases are the ones in which the stochastic programming formulation appears to be of more importance, since randomness plays a larger role. We will say a bit more about this in the next section.

8.4. General Comments on Computation

Aside from the examples presented in the previous section, we performed tests on a variety of one- and two-leg problems. In this section, we describe some of our observations based on these experiments. We first discuss the issue of computational time and data storage. The stochastic optimization algorithm does seem to yield significant savings in this respect, particularly for large problems. For the problems described in §§8.2 and 8.3, there was no gain in computational time, but there was an advantage in storage requirements, because the MDP routine needs to store the optimal decision rules. The comparison becomes

Table 3 Mean Revenues and Corresponding Percentages of the Simulated Optimal Value for the Various Values of Switch Time and Various Methods (for the Example in §8.3)

σ	LP-MDP	LP(200)-MDP	LP(50)-MDP	LP(25)-MDP	SP-MDP
1	142,891.10, 627.77 100%				
201	139,799.80, 475.63 97.8%	139,799.80, 475.63 97.8%	140,244.50, 496.86 98.2%	140,181.10, 494.39 98.1%	142,187.55, 638.50 99.5%
401	138,112.40, 383.44 96.7%	138,405.10, 435.85 96.9%	138,311.35, 446.16 96.8%	138,212.70, 446.64 96.7%	142,703.10, 647,77 99.9%
601	137,821.10, 382.88 96.5%	137,479.80, 423.08 96.2%	137,336.25, 444.09 96.1%	137,246.85, 445.10 96.1%	141,834.40, 646.76 99.2%
801	135,730.60, 384.81 95.0%	136,119.10, 425.51 95.3%	136,353.75, 441.63 95.4%	136,500.55, 441.55 95.5%	141,299.40, 634.60 98.9%
1001	134,136.20, 383.28 93.9%	134,520.55, 428.40 94.1%	135,521.50, 454.54 94.9%	135,818.55, 452.32 95.1%	139,460.90, 599.69 97.6%

starker when the problem is scaled up. For example, it is easy to check that in two-leg problems, the time taken by the MDP routine is proportional to the product of the capacity of the legs; thus, by doubling the capacity of each leg, we expect the MDP algorithm to take four times as long. Indeed, in experiments for two-leg problems with capacity of 300 seats per leg, the time-decomposition method yielded savings of 40% in time, with even bigger savings in terms of memory requirements, and little loss in the value of the objective function. For networks with more legs, such computational savings nevertheless do not result in a tractable method, necessitating the introduction of state-space decomposition or some other approach.

The time required by the simulation-based optimization routine to solve (12) depends on the sample sizes used in Step 2 of the algorithm and the significance level used for the statistical stopping criteria in Step 7 (among other things). Naturally, the routine tends to be slower when we use larger sample sizes and lower significance levels. Moreover, the time spent within the stochastic optimization routine increases as the switch time becomes closer to departure. This occurs because the sampling variance of $D_{(\sigma-1)}$ increases as σ increases. However, this is counterbalanced by a decrease in the time spent on the MDP, computing v_σ. Therefore, the net effect is that computational times decrease as σ increases.

Policy SP-MDP tends to perform better than LP(n)-MDP when there are different categories with very different fares but the same resource requirements arriving simultaneously, as in the example in §8.3. In part, this is because LP-based allocations have difficulties with such situations, regardless of how often the policy is updated, because the LP allocations are in many instances unaffected by the magnitude of the difference in fares. When there are no such significant overlaps, LP(n)-MDP and SP-MDP typically perform similarly, with LP(n)-MDP with "frequent-enough" updating sometimes performing a bit better than SP-MDP, given a fixed switch time σ, but with SP-MDP usually beating LP(n)-MDP with infrequent updating and large σ. Nevertheless, we cannot make a blanket statement that one is better than the other, because performance is greatly influenced by the specifics of a particular problem. Our numerical experience also suggests that, in most cases, re-solving the LP more frequently yields higher or equivalent average revenues, compared with less frequent re-solving. This is not so surprising, since re-solving does incorporate

Table 4 Differences in Revenue (for the Example in §8.3)

σ	LP-MDP	LP(200)-MDP	LP(50)-MDP	LP(25)-MDP
201	2,387.75, 245.71 +	2,387.75, 245.71 +	1,943.05, 215.59 +	2,006.45, 218.41 +
401	4,590.70, 324.32 +	4,298.00, 301.39 +	4,391.75, 291.82 +	4,490.40, 290.02 +
601	4,013.30, 322.62 +	4,317.25, 636.08 +	4,498.15, 288.59 +	4,587.55, 289.35 +
801	5,568.80, 319.65 +	5,180.30, 296.13 +	4,945.65, 272.23 +	4,798.85, 274.04 +
1,001	5,324.70, 263.10 +	4,940.35, 254.24 +	3,939.40, 243.12 +	3,642.35, 242.56 +

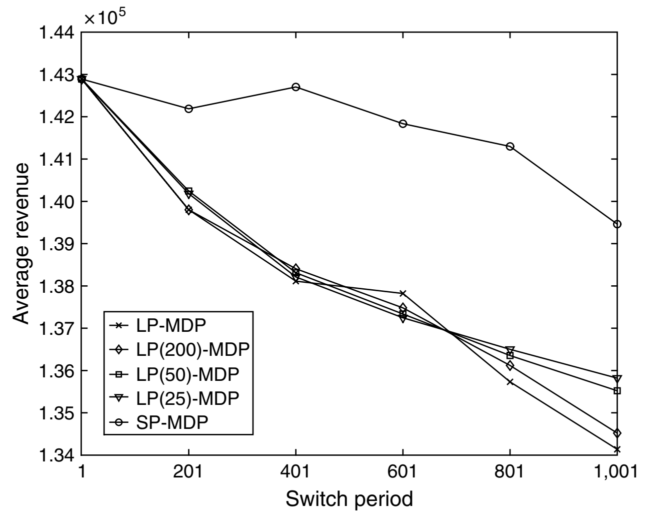

Figure 3 Mean Revenue as a Function of Switch Time (for the Example in §8.3)

new information into the optimization problem. The advantages are more noticeable for larger values of σ. It is perhaps tempting to conjecture that (frequent) re-solving must be better than not doing so; however, this need not be the case, as shown in §5 of Cooper (2002), which contains an example that shows analytically that a policy based on re-solving the LP can, in fact, yield lower expected revenue than one that does not re-solve the LP. Nonetheless, the above conjecture does hold when one repeatedly solves (9) instead of the LP (8); see Chen and Homem-de-Mello (2006).

Another comment concerns the switch time σ. The example in §8.2 suggests that, if we switch to the MDP by time 401 or 601 (for the 1,000 time-unit problem), the switching heuristics all tend to yield nearly the same (from a statistical point of view) mean revenue as the pure MDP. This is again consistent with our original motivation for using switching methods—namely, that it suffices to use relatively crude decision rules far from departure so long as the key decisions near departure are made using the MDP methods. However, in the example in §8.3, the switching time appears to be more critical for the LP(n)-MDP policies than for SP-MDP.

8.5. Numerical Results for a Hub-and-Spoke Network

To test the state-space decomposition heuristic described in §7.2, we developed a series of examples for a hub-and-spoke network with three inbound flights and three outbound flights. The network is depicted in Figure 4. There are three inbound one-leg "local" itineraries, three outbound one-leg "local" itineraries, and nine two-leg "through" itineraries. For each itinerary, we considered two fare categories—"high" and "low" giving a total of $n = (3 + 3 + 9) \times 2 = 30$ itinerary-fare classes. Each flight is assumed to have 100 seats. Such a problem with six legs ($m = 6$) cannot be solved by direct MDP methods because of the curse of dimensionality.

To describe the various problem instances we considered, we begin by describing what we term the

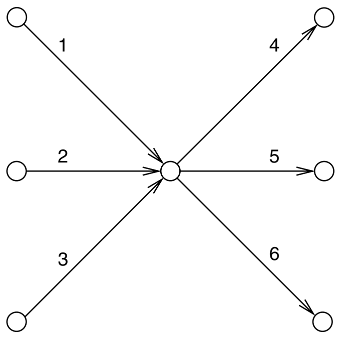

Figure 4 A Six-Leg Hub-and-Spoke Network

"separable" problem. The separable problem is essentially comprised of three two-leg problems. Specifically, among the "through" itineraries, there is demand only for those that use the following leg pairs:(1, 4), (2, 5), and (3, 6). There is no demand for itineraries on the pairs (1, 5), (1, 6), (2, 4), (2, 6), (3, 4), and (3, 5). Under these assumptions, it is easy to see that the problem with six legs decomposes into three two-leg problems. (We will relax the separability assumption for most of our examples, as described below.)

For each of the through itineraries (1, 4), (2, 5), and (3, 6), the overall mean demand for the high fare was 28.67 and the overall mean demand for the low fare was 57.25. For each local itinerary, the overall mean demand for the high fare was 9.56, and the overall mean demand for the low fare was 19.08. Hence, the total expected demand on each leg was 114.56. As in our earlier examples, we assumed there to be $\tau = 1{,}000$ time periods. At time t the probability of a request for each particular high-fare (respectively, low-fare) through itinerary is—approximately, because of rounding—$q_{\mathrm{H}}(t) = 0.015\sin(\pi t/2000)$ $[q_{\mathrm{L}}(t) = 0.03\cos(\pi t/2000)]$. For each individual high-fare (low-fare) local itinerary, the request probability is $0.045\sin(\pi t/2000)[0.09\cos(\pi t/2000)]$. Note that the arrival probabilities for high-fare (low-fare) requests increase (decrease) as departure nears.

We assumed that the high (low) fare on the through itineraries was \$800 (\$500). For each local itinerary, the high (low) fare was \$500 (\$300). We also considered a variety of instances with different "asymmetric" fares on different through and local itineraries. Introducing such asymmetry did not appear to have a clear systematic effect on the results, so in the interest of simplicity we report on just the symmetric case.

For scalar $\lambda \in [0, 1]$ we can create another problem instance by setting the arrival probabilities on the high-fare (low-fare) through itineraries (1, 4), (2, 5), and (3, 6) to $(1/3)(1 + 2\lambda) \times q_{\mathrm{H}}(t)$ $[(1/3)(1 + 2\lambda) \times q_{\mathrm{L}}(t)]$ and setting the arrival probabilities for each high-fare (low-fare) through itinerary among (1, 5), (1, 6), (2, 4), (2, 6), (3, 4), and (3, 5) to $(1/3)(1 - \lambda) \times q_{\mathrm{H}}(t)$ $[(1/3)(1 - \lambda) \times q_{\mathrm{L}}(t)]$. Other problem parameters (fares, local arrival probabilities, and so forth) remain unchanged. The parameter λ measures roughly how close a problem instance is to separable. Observe that setting λ to 1 gives the separable problem. In contrast, setting λ to 0 yields a symmetric problem in the sense that the arrival process for each of the nine high-fare through itineraries has a common distribution, and the arrival process for each of the nine low-fare through itineraries has a common distribution. In the $\lambda = 0$ case, the mean demand for each of the nine high-fare (low-fare) though itineraries is 9.56 (19.08). Observe also that mean demand per leg does not change with λ.

Table 5 **Mean Revenues and Corresponding Percentages of the Upper Bound for the Various Values of Switch Time and the Various Values of λ (for the Six-Leg Hub-and-Spoke Network)**

σ/λ	0	1/3	2/3	1
1	190,200.70, 275.21 97.1%	190,349.70, 273.90 97.2%	190,559.30, 274.82 97.3%	190,916.00, 278.33 97.5%
201	190,200.80, 275.41 97.1%	190,344.70, 273.32 97.2%	190,536.20, 275.45 97.3%	190,914.10, 278.19 97.5%
401	190,109.80, 275.35 97.1%	190,234.00, 279.75 97.1%	190,415.10, 276.47 97.2%	190,858.60, 279.66 97.4%
601	189,484.90, 308.29 96.7%	189,515.30, 313.56 96.8%	189,760.20, 302.54 96.9%	190,413.60, 276.40 97.2%
801	186,668.70, 383.96 95.3%	186,983.50, 368.29 95.5%	187,298.10, 359.14 95.6%	188,846.20, 348.99 96.4%
1,001	175,883.80, 389.32 89.8%	174,793.90, 356.62 89.2%	174,613.20, 328.23 89.2%	182,352.10, 304.57 93.1%
LP	175,433.20, 354.07 89.6%	174,271.90, 348.51 89.0%	176,432.00, 364.84 90.1%	181,314.80, 351.37 92.6%

In Table 5 we report simulation results for values $\lambda = 0$, 1/3, 2/3, and 1. The first six rows of the table show the performance of the policy derived from the state-space decomposition heuristic (with the resulting two-leg problems solved with time decomposition) for the various values of switch times. The table shows the results when the policy from the decomposed problems is reconstituted using the "max-any" method. We also tested the other three reconstitution methods mentioned in §7.2. The max-any method seemed to be the best among the four, although the difference in average revenue among the four was small—typically less than 0.1% for given values of σ and λ. In view of the similarity, we do not show the results for the other three methods. As a simple benchmark, the row labeled "LP" shows the performance of the LP allocation policy (with no re-solving). In each cell, the top left number shows the mean revenue, over 1,000 simulation replications, for the policy in question. The top right number shows the half-width of a 99% confidence interval for the expected revenue from following the policy. Hence, the overall confidence level for the intervals displayed in a single column of the table is at least 93%. For a given λ, on each of the 1,000 simulation replications, common random numbers were used to generate the arrival process for each value of σ as well as for the LP policy.

To give an idea of the quality of the results, we display the percentage ratios (this is the bottom number in each cell) between the expected revenues and the value of the objective function of the LP, which was 195,863.9 in each case. Recall that the LP objective function gives an upper bound on the expected revenue of an optimal policy. For instance, if $\sigma = 401$, we see that the decomposition heuristic is within 2.9% of optimum for $\lambda = 1/3$. This is based on comparison with an upper bound on the value function, so the gap may actually be smaller. Note also that the entry for $\lambda = 1$, $\sigma = 1$ is an estimate of the value function of the MDP for the $\lambda = 1$ (separable) case. Here, exact calculation of the value function gives 190,839.5. For the other values of λ, the problem does not separate; hence, the value function cannot be computed exactly, and we must rely on simulation.

As can be seen from the table, using the decomposition heuristic with switch time $\sigma = 1{,}001$—so that the MDP is never invoked—gives performance similar to that of the LP. However, for the other values of σ the performance is considerably better. As might be expected, the heuristic performs better for lower values of σ. Much of the benefit of incorporating the MDP can be obtained by switching to the detailed policy at $\sigma = 601$.

9. Conclusions

Many authors have considered variants of MDP models for revenue management. Likewise, there is a large body of work that describes how to solve revenue management problems using mathematical programming techniques. In this paper, we have studied revenue management policies that combine aspects of MDP and math-programming approaches by decomposing the booking time horizon into two parts and applying different optimization methods in each portion. We present a formulation that casts the problem as a two-stage stochastic optimization problem and have shown that some models in the literature fit such a framework. In addition, we adapt recently developed variable-sample stochastic optimization techniques to solve a particular case of the formulation. Because the algorithm is suited for two-leg problems, we have developed a novel state-space decomposition approach under which a network is partitioned into two-leg subnetworks, and then the results of the subproblems are combined. Our numerical results suggest that the proposed procedure is effective.

On the more theoretical side, we prove new structural properties of the MDP value function by viewing the two-leg problem as a multistage stochastic linear integer program. Moreover, we give an example that shows that such properties do not, in general, hold for networks with three or more legs.

We do not consider overbooking, cancellations, or consumer choice behavior. Because MDP models that incorporate these effects do exist (e.g., Subramanian, Stidham, and Lautenbacher 1999; Talluri and van Ryzin 2004a), it is not difficult from a conceptual standpoint to incorporate such effects into our framework. In addition, stochastic optimization techniques appear to be equally well suited for handling some such problems (see Karaesmen and van Ryzin 2004; van Ryzin and Vulcano 2006b). Other directions for future work include a theoretical study of the effect of the switch time on the revenue, methods for addressing uncertainty in the forecasts of model parameters, and a study of necessary and sufficient conditions for concavity of the value function in the network case. It would also be interesting to study the time-decomposition paradigm discussed in this paper with other algorithms, such as those listed in §5.

Acknowledgments
The authors would like to thank Dan Zhang for conducting numerical tests. The authors also thank the associate editor and two anonymous referees for their comments. This material is based on work supported by the National Science Foundation under Grant DMI-0115385.

Appendix

Proof of Theorem 1. In preparation, we first define the base matrix B_1 as

$$B_1 \equiv \begin{pmatrix} A^0 & & & \\ & A^1 & & \\ & & \ddots & \\ & & & A^n \end{pmatrix}. \tag{28}$$

Then, by properly arranging variables, the matrix of constraints in (13) has the following recursive form:

$$B_\tau = \begin{pmatrix} \tilde{B}^0_{\tau-1} & & & \\ & \tilde{B}^1_{\tau-1} & & \\ & & \ddots & \\ & & & \tilde{B}^n_{\tau-1} \end{pmatrix}, \tag{29}$$

where $\tilde{B}^i_t$ is defined as $[\tilde{A}^i_t \; B_t]$, and $\tilde{A}^i_t$ is a column vector formed by multiple replications of A^i, the ith column of matrix A (recall that, by convention, A^0 is a vector of zeroes). The number of replications is as many as is necessary to make $\tilde{A}^i_t$ have the same number of rows as B_t. For example, when $n=3$ and $\tau=2$, we have

$$B_\tau = \left(\begin{array}{cccccccccccccccccccc}
A^0 & A^0 \\
A^0 & & A^1 \\
A^0 & & & A^2 \\
A^0 & & & & A^3 \\
& & & & & A^1 & A^0 \\
& & & & & A^1 & & A^1 \\
& & & & & A^1 & & & A^2 \\
& & & & & A^1 & & & & A^3 \\
& & & & & & & & & & A^2 & A^0 \\
& & & & & & & & & & A^2 & & A^1 \\
& & & & & & & & & & A^2 & & & A^2 \\
& & & & & & & & & & A^2 & & & & A^3 \\
& & & & & & & & & & & & & & & A^3 & A^0 \\
& & & & & & & & & & & & & & & A^3 & & A^1 \\
& & & & & & & & & & & & & & & A^3 & & & A^2 \\
& & & & & & & & & & & & & & & A^3 & & & & A^3
\end{array}\right).$$

The matrix B_τ has the following fundamental properties, each of which can be verified directly from (29):

(I) Each column c of B_τ is of the form

$$c = \begin{pmatrix} 0 \\ \tilde{A}^i_t \\ 0 \end{pmatrix} \tag{30}$$

for some i and some t. By assumption, A has only two rows, so each column of A is one of the following vectors: $a_0 \equiv [0 \;\; 0]'$, $a_1 \equiv [1 \;\; 0]'$, $a_2 \equiv [0 \;\; 1]'$, or $a_3 \equiv [1 \;\; 1]'$. Therefore, each column of B_τ is formed by successive replications of some a_j, with an even number of zeroes both above and below $\tilde{A}^i_t$. For a column c of B_τ, we write $c \in C_i$ ($i \in \{0,1,2,3\}$) if the vector $\tilde{A}^i_t$ in (30) is comprised of replications of a_i.

(II) Let b be an arbitrary column of B_τ. Let $\tilde{A}^i_t$ be the vector in that column (cf. 30), and let u and v denote the rows corresponding to, respectively, the first and last elements of $\tilde{A}^i_t$. Then, the submatrix S_{uv} formed by columns $1,\ldots,b$ and rows $u,\ldots,v$ of B_τ is such that all odd-numbered rows are identical and all even-numbered rows are identical.

(III) In the notation of (II), let $S_{v+1,w}$ be the submatrix formed by columns $1,\ldots,b$ and rows $v+1,\ldots,w$, where w is the total number of rows in B_τ. Each odd-numbered (respectively even-numbered) row of $S_{v+1,w}$ coincides with the odd-numbered (respectively even-numbered) row of S_{uv} up to a certain column, and then the remaining elements in the row of $S_{v+1,w}$ are zeroes. Moreover, the odd-numbered (respectively even-numbered) rows of $S_{v+1,w}$ have an increasing number of zeroes.

To prove the theorem, it suffices to prove that B_τ is indeed TU. To do so, we shall show by induction that the determinant of each square submatrix of B_τ is 0, 1, or -1. It easy to see that, because B_τ has only $\{0,1\}$-entries, the determinant of any 2×2 square submatrix of B_τ is 0, 1, or -1. Suppose now this is true for any square submatrix of order k. We want to show the property is valid for any square submatrix of order $k+1$.

Let M be a square submatrix of B_τ of order $k+1$, with M^j, $j=1,\ldots,k+1$ denoting the columns of M. Recall the definition of C_i, $i=0,1,2,3$, from property (I) above. We will say that a column j of M is an element of C_i if the column to which j belonged in B_τ is an element of C_i. Similarly, we will call a row even or odd depending on its location in B_τ. Also, we will call an entry of M even or odd depending on whether the row where such entry lies is even or odd.

Let l denote the index of the right-most column of M that either (a) is not an element of C_3 or (b) is an element of C_3 but does not contain exactly one even-indexed "1" and exactly one odd-indexed "1." Suppose initially that $l=k+1$. If M^{k+1} contains two or more even entries of $\tilde{A}^i_t$ or two or more odd entries of $\tilde{A}^i_t$, then by Property (II) M contains at least two identical rows, so the determinant of M is zero. By the definition of l, M^{k+1} cannot contain exactly one even and one odd "1." Hence, the only other possibilities are those in which M^{k+1} has at most a single nonzero entry. If M^{k+1} is all zeroes, then obviously $\det(M)=0$. If M^{k+1} contains exactly one "1," then we have $\det(M)=(\pm 1)\det(M')$, where M' is a matrix of order k. By the induction hypothesis, $\det(M')\in\{-1,0,1\}$, and thus $\det(M)\in\{-1,0,1\}$. Therefore, if $l=k+1$, then $\det(M)\in\{-1,0,1\}$.

Suppose now that $l\le k$. Then, by the definition of l, columns $l+1,\ldots,k+1$ are elements of C_3, each with a single odd "1" and a single even "1." By Property (III) above, the nonzero elements of M^j, $l+1 \le j \le k$, cannot be located "southwest" of the nonzero elements of any M^p, $j+1 \le p \le k+1$. Moreover, if the nonzero elements of M^j are located next to (i.e., "west of") the nonzero elements of M^{j+1}, then M has two identical columns, and thus $\det(M) = 0$. It follows that we only need to analyze the case where the a_3 vector in M^j is located "northwest" of the a_3 vector in M^{j+1}, $j = l+1, \ldots, k$.

Consider now column l. If M^l contains a single replication of some a_i, then by the definition of l we must have $i \ne 3$, which implies M^l has at most one nonzero entry. In this case we have that either $\det(M) = 0$ (in case $i = 0$) or $\det(M) = (\pm 1)\det(M')$, where M' is a matrix of order k. By the induction hypothesis, we have then $\det(M) \in \{-1, 0, 1\}$.

Suppose now that M^l contains more than one even (odd) element of $\tilde{A}^i_j$ and that at least one of those entries is located "northwest" of the vector a_3 in M^{l+1}. Then that element has only zeroes to its right. Next, let M_1 and M_2 denote the $k \times k$ matrices formed from M by deleting column $l+1$ and deleting the row corresponding, respectively, to the first and second nonzero entry of M^{l+1}. By expanding the determinant over column $l+1$, we have that $\det(M) = \pm[\det(M_1) - \det(M_2)]$. It follows that either M_1 or M_2 (or both) have repeated rows, and so $\det(M_1) = 0$ or $\det(M_2) = 0$ (or both). Because $\det(M_i) \in \{-1, 0, 1\}$ by the induction hypothesis, it follows that $\det(M) \in \{-1, 0, 1\}$.

It remains to analyze the situation where M^l contains more than one replication of some vector a_i, and those elements are located "west" and "southwest" of the vector a_3 in M^{l+1}. Within this context, the only configurations of M we need to study are the following (all other possibilities entail one of the replications of an element of a_i having only zeroes to its right, which can then be analyzed using the argument in the previous paragraph):

$$M = \begin{pmatrix} D & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ Y & a_i & a_3 & & 0 & 0 & & 0 \\ Y & \vdots & 0 & \ddots & 0 & 0 & & 0 \\ Y & a_i & 0 & & a_3 & 0 & & 0 \\ E_1 & 0 & 0 & & 0 & a_3 & & 0 \\ \vdots & 0 & 0 & & 0 & 0 & \ddots & 0 \\ E_\eta & 0 & 0 & & 0 & 0 & & a_3 \\ G & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix}. \qquad (31)$$

Here, D and G are certain (possibly empty) matrices with $l-1$ columns, $E_1, \ldots, E_\eta$ are $2 \times (l-1)$ matrices, and Y is a $2 \times (l-1)$ matrix.

Let q be the number of replications of a_i in column M^l. We consider three different cases:

Case 1. $q = 1$. Notice that in this case we must have $j \ne 3$ by the definition of l. Therefore, we have $\det(M) = \pm\det(M')$, where M' is obtained from M by deleting the column M^l and the row corresponding to the nonzero element of M^l. Hence, $\det(M) \in \{-1, 0, 1\}$ by the induction hypothesis.

Case 2. $q = 2$. For $i = 1, 2$, let M_1 and M_2 be the $k \times k$ matrices formed from M by deleting column $l+1$ and deleting the row corresponding, respectively, to the first and second nonzero entry of M^{l+1}. For $i = 1, 2$; $j = 1, 2$, let M_{ij} denote the $(k-1) \times (k-1)$ matrix obtained from M_i by deleting the column corresponding to M^{l+2} and the row corresponding to the jth nonzero entry of M^{l+2}. By expanding over the columns $l+1$ and $l+2$ of M we obtain that

$$\det(M) = \pm[\det(M_{11}) - \det(M_{12}) - \det(M_{21}) + \det(M_{22})]. \qquad (32)$$

When M has configuration (31), we can see that we have $\det(M_{11}) = \det(M_{22}) = 0$. Moreover, $\det(M_{12}) = -\det(M_{21})$, because M_{12} and M_{21} are equal except for the exchange of two rows. Hence, it follows from (32) that $\det(M) = 0$.

Case 3. $q \ge 3$. Here, *any* submatrix of M formed by deleting columns $M^{l+1}, \ldots, M^{l+q}$ and *any* q rows will have at least two repeated rows. Hence, any such submatrix has determinant zero. Expanding over columns $M^{l+1}, \ldots, M^{l+q}$, we have $\det(M) = 0$.

Finally, we need to consider the case in which l does not exist; that is, every column of M contains exactly one replication of a_3. Because a_3 has two entries, there must be repeated columns (otherwise M could not be square), in which case $\det(M) = 0$. This concludes the proof. □

References

Bertsekas, D. P. 1999. *Nonlinear Programming*, 2nd ed. Athena Scientific, Belmont, MA.

Bertsimas, D., S. de Boer. 2005. Simulation-based booking limits for airline revenue management. *Oper. Res.* **53** 90–106.

Bertsimas, D., I. Popescu. 2003. Revenue management in a dynamic network environment. *Transportation Sci.* **37** 257–277.

Birge, J. R., F. Louveaux. 1997. *Introduction to Stochastic Programming.* Springer-Verlag, New York.

Bitran, G., R. Caldentey. 2003. An overview of pricing models for revenue management. *Manufacturing Service Oper. Management* **5** 202–229.

Chen, L., T. Homem-de-Mello. 2006. Re-solving stochastic programming models for airline revenue management. Working Paper 04-012, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL. www.optimization-online.org.

Chen, V. C. P., D. Günther, E. L. Johnson. 1998. A Markov decision problem based approach to the airline YM problem. Working paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Clarke, F. H. 1983. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, Inc., New York.

Cooper, W. L. 2002. Asymptotic behavior of an allocation policy for revenue management. *Oper. Res.* **50** 720–727.

de Boer, S. V., R. Freling, N. Piersma. 2002. Mathematical programming for network revenue management revisited. *Eur. J. Oper. Res.* **137** 72–92.

Feng, Y., B. Xiao. 2000. Optimal policies of yield management with multiple predetermined prices. *Oper. Res.* **48** 332–343.

Galil, Z., S. Micali, H. Gabow. 1986. An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Comput.* **15**(1) 120–130.

Gallego, G., G. van Ryzin. 1994. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Sci.* **40** 999–1020.

Gallego, G., G. van Ryzin. 1997. A multiproduct dynamic pricing problem and its applications to network yield management. *Oper. Res.* **45** 24–41.

Günther, D. 1998. Airline yield management: Optimal bid prices, Markov decision processes, and routing considerations. Unpublished doctoral dissertation, Georgia Institute of Technology, Atlanta, GA.

Homem-de-Mello, T., A. Shapiro, M. L. Spearman. 1999. Finding optimal material release times using simulation based optimization. *Management Sci.* **45** 86–102.

Karaesmen, I., G. van Ryzin. 2004. Overbooking with substitutable inventory classes. *Oper. Res.* **52** 83–104.

Kleywegt, A. J., J. D. Papastavrou. 1998. The dynamic and stochastic knapsack problem. *Oper. Res.* **46** 17–35.

Kushner, H. J., G. G. Yin. 1997. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, Berlin, Germany.

Lautenbacher, C. J., S. Stidham. 1999. The underlying Markov decision process in the single-leg airline yield management problem. *Transportation Sci.* **33** 136–146.

Law, A. M., W. D. Kelton. 2000. *Simulation Modeling and Analysis*, 3rd ed. McGraw-Hill, New York.

Lee, T. C., M. Hersh. 1993. A model for dynamic airline seat inventory control with multiple seat bookings. *Transportation Sci.* **27** 252–265.

Murota, K. 2003. *Discrete Convex Analysis.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Prékopa, A. 1995. *Stochastic Programming*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

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

Robinson, L. W. 1995. Optimal and approximate control policies for airline booking with sequential nonmonotonic fare classes. *Oper. Res.* **43** 252–263.

Ross, S. M. 1996. *Stochastic Processes*. John Wiley & Sons, New York.

Schrijver, A. 1986. *Theory of Linear and Integer Programming*. Wiley, New York.

Shapiro, A. 2003. Monte Carlo sampling methods. A. Ruszczynski, A. Shapiro, eds. *Handbook of Stochastic Optimization*. Elsevier Science Publishers B.V., Amsterdam, The Netherlands.

Shapiro, A., T. Homem-de-Mello. 1998. A simulation-based approach to two-stage stochastic programming with recourse. *Math. Programming* **81** 301–325.

Smith, B. C., J. F. Leimkuhler, R. M. Darrow. 1992. Yield management at American Airlines. *Interfaces* **22** 8–31.

Subramanian, J., S. Stidham, C. J. Lautenbacher. 1999. Airline yield management with overbooking, cancellations, and no-shows. *Transportation Sci.* **33** 147–168.

Talluri, K., G. van Ryzin. 1998. An analysis of bid-price controls for network revenue management. *Management Sci.* **44** 1577–1593.

Talluri, K., G. van Ryzin. 1999. A randomized linear programming method for computing network bid prices. *Transportation Sci.* **33** 207–216.

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

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

van Ryzin, G., J. McGill. 2000. Revenue management without forecasting or optimization: An adaptive algorithm for determining airline seat protection levels. *Management Sci.* **46** 760–775.

van Ryzin, G., G. Vulcano. 2006a. Simulation-based optimization of virtual nesting controls for network revenue management. *Oper. Res.* Forthcoming.

van Ryzin, G., G. Vulcano. 2006b. Computing virtual nesting controls for network revenue management under customer choice behavior. Working Paper DRO-2004-09, Columbia University, Graduate School of Business, New York.

Williamson, E. L. 1992. Airline network seat control. Unpublished doctoral dissertation, Massachusetts Institute of Technology, Cambridge, MA.

You, P.-S. 1999. Dynamic pricing in airline seat management for flights with multiple flight legs. *Transportation Sci.* **33** 192–206.

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