

Department of Industrial Engineering and Management Sciences

Northwestern University, Evanston, Illinois 60208-3119, U.S.A.

Working Paper No. 03-015

A Class of Hybrid Methods for Revenue Management

William L. Cooper

University of Minnesota

Tito Homem-de-Mello

Northwestern University

October 2003

This research has been supported
by the National Science Foundation under grant DMI-0115385.

A Class of Hybrid Methods for Revenue Management

William L. Cooper
Graduate Program in Industrial Engineering
Department of Mechanical Engineering
University of Minnesota
Minneapolis, MN 55455
`billcoop@me.umn.edu`

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

October 6, 2003

Abstract

We consider a Markov decision process model of a network revenue management problem. Working within this formal framework, we study policies that combine aspects of mathematical programming approaches and pure Markov decision process methods. The 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 resulting hybrid policies do perform strongly. By viewing the Markov decision process as a large stochastic program, we derive some structural properties of two-leg problems. We also show that these properties cannot, in general, be extended to larger networks.

1 Introduction

A wide number of industries face problems in which various products, comprised of combinations of resources, need to be 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 in order 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 McGill and van Ryzin (1999) 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 multi-leg 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 et al. (1999), who consider cancellations and no-shows, and Talluri and van Ryzin (2001) 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. For references to both MDP and math programming work, see McGill and van Ryzin (1999). We will provide more discussion of some math programming approaches in Sections 4 and 5.

In the present paper, we present a Markov decision process 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, since allocation and booking-limit policies are, in fact, used in practice. Nevertheless, since 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 we can actually do this.

One focus of this paper is a family of “hybrid” 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 hybrid 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 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, since 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 basic hybrid formulation is still 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 hybrid policies using allocations in the first stage perform strongly for certain problems. In fact, in a wide range of one- and two-leg problems, our experiments have shown that despite the apparent crudeness of allocation policies, the hybrid formulation with allocations performs nearly as well as the optimal policy. Indeed, these hybrid policies are motivated in part by recent results that suggest that, in a certain sense, 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). In addition, Subramanian et al. (1999) suggest (p. 163) that near the time of departure, when remaining demand and capacity are typically small, it is important to employ decision rules based upon a detailed, accurate model.

The hybrid formulation we propose falls into the general category of static stochastic optimization problems, where the goal is to minimize (or 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, since underlying distributions are assumed to be known.

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 EMSR formulation. McGill and van Ryzin (1999) 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 (1998) employ a stochastic-gradient method (another SA variant) to solve an overbooking problem.

Few papers have considered how to blend the MDP and math programming approaches. Bertsimas and de Boer (2000) 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. In order to handle larger problems they discuss methods for approximating an MDP value function — we will discuss this work in greater detail in Section 5. 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 problem as a multi-stage stochastic linear program and employing results from integer programming, we are able to show that the value function for two-leg problems is *concave*. You (1999) proves componentwise concavity of the value function for two-leg problems; however, our result is stronger, because concavity implies componentwise concavity, but not vice-versa. In addition, our result extends beyond the MDP setting to problems with non-Markovian demand processes. See, e.g., Lee and Hersh (1993) for results in the Markovian single-leg case. 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 hybrid 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) the formulation of the network revenue management problem as an MDP that allows analysis of well-known heuristics within its scope, (2) the description and analysis of the general hybrid approach within a formal stochastic and dynamic framework, (3) adaptation of recently-developed stochastic optimization techniques to solve a particular instance of the formulation, (4) derivation of some structural results on the MDP value function for multiple-leg problems, and (5) demonstration, via numerical examples, that hybrid policies can be a promising technique.

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

2 Problem Framework

We model the revenue management problem as a finite-horizon discrete-time Markov decision process with time units $t = 1, \dots, \tau$. We assume that time moves forward; this means $t = 1$ 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 be also be applied to models that do allow for choice behavior such as those in Talluri and van Ryzin (2001). Whether or not such methods would perform well remains to be seen. See also Zhang and Cooper (2003), where related methods are shown to work well in a slightly different modeling framework.]

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 customary in the revenue management literature, we assume there is at most one unit of demand per time period, so that $P(\exists j_1, j_2 : j_1 \neq j_2, D_{t,j_1} = D_{t,j_2} = 1) = 0$. 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 will 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 within a particular day. In this case, the resources are leg-cabin combinations (so the 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 0s in the columns corresponding to fare class-itineraries that do not use leg-cabin i , and 1s 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 he pays his fare f_j , consumes his resource requirement, and thereafter there are fewer resources available to sell to future customers (recall that we are not considering overbookings in our model). On the other hand, we say that a customer's request is rejected if he is not sold his requested product. In this case, no revenue is realized and no resources are consumed.

3 Markov Decision Process Methods

In this section, we describe a Markov decision process 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 upon 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 j -th 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 Sections 4 and 5) require this information. In particular, if we instead (as in most papers) 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 will denote the random sequence of states by $(S_1, \dots, S_{\tau+1})$ and the random sequence of actions by (U_1, \dots, U_{τ}) ; recall

that each S_t and U_t is an n -dimensional vectors. 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 $P(S_{t+1} = s' | S_t = s, U_t = u) = P(s + uD_t = s')$, 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 single-period reward function that is the *expected revenue* in the period. Since our ultimate objective is to maximize expected revenue (see 1 below), taking this “extra” expectation 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 $h_t = (s_1, u_1, s_2, u_2, \dots, s_t)$ where $s_k = (s_{k,1}, \dots, s_{k,n})$ is the state at time k and $u_k = (u_{k,1}, \dots, u_{k,n})$ is the action taken at time k . 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 $\pi_t(h_t) = a = (a_1, \dots, a_n) \in \{0, 1\}^n$ means that based upon the occurrence of h_t , we select action a .

We will 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 $h_t = (s_1, u_1, s_2, u_2, \dots, s_t)$ for which $c - As_t = r$.

A *policy* specifies the decision rules in effect each period. We will use $\pi = (\pi_1, \dots, \pi_\tau)$ to denote a generic policy, where π_t is the decision rule in effect for period t . For a particular policy, note that the actual action implemented by 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 upon the realized values of (D_1, \dots, 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). Since 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 or not the customer we booking period 1 was from class 1 or from class 2. Hence, we have no way of instituting the policy “accept at most one from class 2.”

Alternatively, one could argue that we should employ a more-complicated state vector, since the airline could, in principle, know the entire demand sequence up to time t along with its own sequence of actions. However, most airlines apparently do not have such detailed information. Hence, we have chosen the “cruder” state vector detailed above.

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 that for each t is a function solely of the current state. Let s denote generic state. Then one can compute an optimal policy and the corresponding maximum in (1) via the recursive optimality equations

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

and $w_{\tau+1}(\cdot) = 0$. In the above, uD_t should be interpreted as componentwise multiplications; i.e., uD_t is the n -vector with entries $u_j \times D_{t,j}$. For each $t \in \{1, \dots, \tau + 1\}$, $w_t(s)$ represents the maximum possible expected revenue that can be generated in periods t, \dots, τ , provided that s is the state in period t . Moreover, $w_1(0)$ is the maximum in (1). If $\pi^* = (\pi_1^*, \dots, \pi_\tau^*)$ 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 only the current state.

As the following result shows, there exist Markovian optimal policies whose decisions depend only upon the remaining capacity vector $c - As$. 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 discrete-allocation and booking-limit policies unmeasurable with respect to the history. Hence, if we want to consider such policies within an MDP framework, it is necessary to prove that optimal policies can ignore the “extra” state information.

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 backwards 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 inductive step is identical, noticing that $A(s + uD_t) = A(\bar{s} + uD_t)$. 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

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

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 inasmuch 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 et al. (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 1 in the j -position and 0s elsewhere, and A^j be the j -th column of the matrix A . For each t , j and r define

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

Then, π^* is a Markovian optimal policy for (1). We omit proof of this fact, since results of this type have been obtained elsewhere; see, e.g., Lautenbacher and Stidham (1999) or 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

$$\mathcal{U} \equiv \{u \in \{0,1\}^n : r - A(uD_t) \geq 0 \text{ w.p.1}\} = \{u \in \{0,1\}^n : r - A^j u_j \geq 0, j = 1, \dots, n\}. \quad (5)$$

So, problem (3) can be rewritten as

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

Formulation (3) — or, more precisely, its rewritten form (6) — can be interpreted as a *multi-stage* $\{0, 1\}$ -stochastic linear program (SLP). Indeed, 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 the function $V_t(u^{t-1}, D^{t-1})$ as

$$\begin{aligned} V_t(u^{t-1}, D^{t-1}) &\equiv \max_{u_t \in \{0,1\}^n} f_t \cdot u_t + \mathbb{E} [V_{t+1}(u^t, D^t)] \\ &\text{s. to} \\ A^j u_{t,j} &\leq r - A(u_1 D_1) - \dots - A(u_{t-1} D_{t-1}), \quad j = 1, \dots, n, \end{aligned} \quad (7)$$

where $V_{\tau+1} \equiv 0$. Let $\mathbf{0}_{t-1}$ denote the $(t-1)$ -dimensional vector of zeros. It is clear that $V_t(\mathbf{0}_{t-1}, \mathbf{0}_{t-1}) = v_t(r)$ as defined in (6). Furthermore, problem (7) is in the standard form of multi-stage stochastic linear 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. This LP has a block-angular structure which is amenable to decomposition methods such as Benders's. This has led to the development of a class of methods, typically called *L-shaped*. For a discussion of multi-stage 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 Section 6 we will use the single “large” formulation to obtain structural characteristics of the value function.

4 Mathematical Programming Methods

The recursions in section 3 provide *optimal* solutions to the revenue management problem under study. As mentioned above, however, both the MDP and the multi-stage SLP approaches are 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 j -th entry is the expected class j demand over $(0, \tau]$; $\mu_j \equiv \mathbb{E}D_{(\tau),j}$. The linear program is given by

$$\max_x \{f \cdot x : Ax \leq c, 0 \leq x \leq \mu\}. \quad (8)$$

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} , since 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, see e.g. Chen et al. (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 discrete allocation policies. Note that optimal policies are generally not of this form. For certain multi-class, 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 creating allocations. Key references include Williamson (1992) and Talluri and van Ryzin (1998, 1999).

The LP-based method described above, while simple, has a major drawback — its inability to take randomness into account. The allocations are solely based upon 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, in order to capture the randomness inherent to the problem, one should replace the LP by something that explicitly models randomness. One such formulation is

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

We can use an optimal solution x^* of (9) to obtain a policy for the MDP — simply define $\pi_{t,j}^{\text{SP}}(s) = \mathbb{I}\{s_j \leq x_j^* - 1\}$. 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, since π^{SP} yields a feasible solution to (1), it follows that ν^{SP} is a lower bound on $v_1(c)$.

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 solved using methods described in, e.g., Birge and Louveaux (1997). A similar formulation is discussed in de Boer et al. (2002).

5 Hybrid Methods

We now describe a family of 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 hybrid 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 $t \leq \sigma - 1$, we use decision rule $\delta_t^\theta = (\delta_{t,1}^\theta, \dots, \delta_{t,n}^\theta)$ where, as before, $\delta_{t,j}^\theta$ maps the history of the booking process prior to time t to a decision as to whether or not we are willing to accept a class j request at time t . 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 $(\delta_1^\theta, \dots, \delta_{\sigma-1}^\theta)$. This yields the two-stage stochastic optimization problem,

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

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

$$\pi_t^{[\theta, \sigma]} = \begin{cases} \delta_t^\theta & \text{if } t < \sigma \\ \pi_t^* & \text{if } t \geq \sigma \end{cases} \quad (11)$$

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

Allocations. Perhaps the simplest example of the hybrid formulation involves using allocations prior to σ . Here we have that $\delta_{t,j}^\theta(s) = \mathbb{I}\{s_j \leq \theta_j - 1\}$ where $\theta = (\theta_1, \dots, \theta_n) \in \Theta_\sigma := \{\theta \in \mathbb{Z}_+^n : A\theta \leq c\}$. 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

$$\max_{\theta \in \mathbb{Z}_+^n} \mathbb{E} \left[f \cdot \min\{\theta, D_{(\sigma-1)}\} + v_\sigma(c - A \min\{\theta, D_{(\sigma-1)}\}) \right] \quad \text{subject to} \quad A\theta \leq c. \quad (12)$$

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

Bid Prices. In order 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 $\delta_{t,j}^\theta(s) = \mathbb{I}\{f_j \geq A^j \cdot \theta\} \mathbb{I}\{As + A^j \leq c\}$ where A^j is the j -th 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.

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 upon the computation 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 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 (2000), who use a variant of virtual nesting (see Smith et al. 1992) together with a detailed value function approximation procedure. More recently, van Ryzin and Vulcano (2003) 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 hybrid 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 is expected revenue. 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 $\sigma_1 \leq \sigma_2$ 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 $t = 1, \dots, \sigma_1 - 1$. Then the expected revenue $\mathbb{E}[R^{\text{H}(\sigma)}]$ is non-increasing as a function of switch time σ .*

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

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

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

$$\mathbb{E}R_{\sigma_1}(\phi) \geq \mathbb{E}R^{\text{H}(\sigma_2)} = \mathbb{E}R_{\sigma_2}(\theta^{**})$$

Recall that $\mathbb{E}R_{\sigma_2}(\theta^{**})$ is the expected revenue from the policy $\bar{\pi} = (\delta_1^{\theta^{**}}, \dots, \delta_{\sigma_2-1}^{\theta^{**}}, \pi_{\sigma_2}^*, \dots, \pi_\tau^*)$. By construction $\tilde{\pi}$ and $\bar{\pi}$ are identical over periods $t = 1, \dots, \sigma_1 - 1$. Hence, we need only show that the expected revenue accrued by $\tilde{\pi}$ during periods σ_1, \dots, τ is at least as great as that accrued by $\bar{\pi}$ over the same span. For any policy π the expected revenue over periods σ_1, \dots, τ can be written as

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

Since $\tilde{\pi}$ and $\bar{\pi}$ are identical over periods $t = 1, \dots, \sigma_1 - 1$, we have $P(H_{\sigma_1}^{\tilde{\pi}} = h) = P(H_{\sigma_1}^{\bar{\pi}} = h)$. Moreover, since $\tilde{\pi}$ coincides with π^* over periods σ_1, \dots, τ , it follows that $\tilde{\pi}$ maximizes the quantity $\mathbb{E} \left[\sum_{t=\sigma_1}^{\tau} g(S_t^\pi, \pi_t(H_t^\pi)) \middle| H_{\sigma_1}^\pi = h \right]$ over all allowable policies. This completes the proof. \blacksquare

6 Structural Properties

In this section, we return to the MDP formulation of Section 3. By viewing the problem as a large stochastic program, we prove some structural properties of the value function.

For each $t = 1, \dots, \tau$, let ω^t denote the “randomness” up to time t . Let $\omega_{k_1, \dots, k_t}^t$ be the specific element in ω^t defined by the k_1 th outcome at stage 1, the k_2 th outcome at stage 2, and so on. The

“0th” outcome at stage t means $D_t = 0$, whereas the “ j th” outcome at stage t means $D_t = e_j$. Let $u_t(\omega^{t-1})$ be the vector of decision variables of the optimization problem defined by $V_t(u^{t-1}, D^{t-1})$ in (7). By convention, $u_{t,0} \equiv 0$, $A^0 \equiv 0$, and $u_1(\omega^0) \equiv u_1$. Below, all summations are from 0 to n . We can now rewrite (7) for $t = 1$ in a single binary IP:

$$\begin{aligned} \max \quad & (fp_1) \cdot u_1 + \sum_{k_1} p_{1,k_1} [(fp_2) \cdot u_2(\omega_{k_1}^1)] + \sum_{k_1} p_{1,k_1} \sum_{k_2} p_{2,k_2} [(fp_3) \cdot u_3(\omega_{k_1,k_2}^2)] + \dots \\ & + \sum_{k_1} p_{1,k_1} \sum_{k_2} p_{2,k_2} \dots \sum_{k_{\tau-1}} p_{\tau-1,k_{\tau-1}} [(fp_\tau) \cdot u_\tau(\omega_{k_1,k_2,\dots,k_{\tau-1}}^{\tau-1})] \\ \text{s.to} \quad & \hspace{20em} \text{(MSSLP)} \end{aligned}$$

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

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 $v_t(r)$, $t \leq \tau$, we simply add the constraints $u_s(\omega^{s-1}) = 0$, $s = 1, \dots, t-1$, and set $p_{s,0} = 1$, $s = 1, \dots, t-1$ (recall that, as discussed above, $V_t(\mathbf{0}_{t-1}, \mathbf{0}_{t-1}) = v_t(r)$).

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; since 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 mn^τ 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 MSSLP formulation will allow us to derive important properties of the original problem.

It is worthwhile noticing that, unlike the MDP model, formulation (MSSLP) is valid even without the requirement of independence of demand across stages. That is, one could, in principle,

start from (MSSLP) and model an arbitrary correlation structure. While this may not be feasible from the computational point of view due to the size of the problem, an immediate implication is that *the properties derived below are valid in a much more general context.*

Proposition 3 *Consider the relaxed value function defined by*

$$v_t^R(r) = \max_{u \in [0,1]^n} \{ \mathbb{E} [f \cdot (uD_t) + v_{t+1}^R(r - A(uD_t))] : P(r - A(uD_t) \geq 0) = 1 \} \quad (14)$$

with $v_{\tau+1}^R(\cdot) \equiv 0$ (notice that u is no longer required to be integer). Then, $v_t^R(\cdot)$ is concave and piecewise linear. Moreover, both $v_t^R(\cdot)$ and $v_t(\cdot)$ are nondecreasing and superadditive, i.e., $v_t(r_1) \geq v_t(r_2)$ if $r_1 \geq r_2$ and $v_t(r_1) + v_t(r_2) \leq v_t(r_1 + r_2)$.

Proof. The result can be proved by (backwards) induction, by noticing that (14) has concave objective function and a convex feasibility set [we can rewrite the constraint set of (14) similar to (5)]. Indeed, using that argument, concavity follows even when D_t has continuous distribution. Alternatively, we can use the MSSLP formulation. It is clear that $v_t^R(r)$ is the optimal value of an LP as a function of the right-hand side. Therefore, $v_t^R(r)$ is concave and piecewise linear. The monotonicity and superadditivity properties follow from the fact that $v_t^R(r)$ and $v_t(r)$ are optimal value functions of (integer) LPs. ■

The above proposition shows that the relaxed function $v_t^R(r)$ is indeed concave. Our goal now is to show that $v_t(r)$ and $v_t^R(r)$ are directly related to each other. We hereafter focus on the case with $m = 2$, i.e., two-leg problems. In preparation, we first define the base matrix B_1 as

$$B_1 \equiv \begin{pmatrix} A^0 & & & \\ & A^1 & & \\ & & \ddots & \\ & & & A^n \end{pmatrix}. \quad (15)$$

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

$$B_\tau = \begin{pmatrix} \tilde{B}_{\tau-1}^0 & & & \\ & \tilde{B}_{\tau-1}^1 & & \\ & & \ddots & \\ & & & \tilde{B}_{\tau-1}^n \end{pmatrix}, \quad (16)$$

where, \tilde{B}_t^i is defined as $[\tilde{A}_t^i \ B_t]$, and \tilde{A}_t^i is a column vector formed by multiple replications of A^i , the i th column of matrix A (recall that, by convention, A^0 is a vector of zeros). For example, when

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

To set the stage for the result, 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 $\mathcal{P} = \{y \in \mathbb{R}_+^n : Qy \leq b\}$ 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 B_τ of constraints in (13) is TU, then the solution of the corresponding linear problem, with the integrality constraints relaxed, coincides with the solution of MSSLP whenever the right-hand side is integral. In other words, $v_1(r) = v_1^R(r)$ whenever $r \in \mathbb{Z}_+^n$. In fact, this property can be extended to $v_t(r)$ and $v_t^R(r)$ for a general $t \leq \tau$ — just notice that, as remarked in the paragraph following the MSSLP formulation, the constraint matrix B_τ^t for the problem corresponding to v_t is formed by B_τ and an identity matrix. Thus, B_τ^t is TU if B_τ is TU. The theorem below shows that the latter is true:

Theorem 1 *Consider the function $v_t^R(r)$ defined in (14). Suppose that the resource consumption matrix A is a $\{0, 1\}$ -matrix with only two rows. Then, $v_t(r) = v_t^R(r)$ for all $r \in \mathbb{Z}_+^n$. In that sense, the value function $v_t(r)$ is concave.*

Proof. 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 is 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 , $j = 1, \dots, k + 1$ 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 upon its location in B_τ . Also, we will call an entry of M even or odd depending upon 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 $\ell = k + 1$. If M^{k+1} contains two or more even entries

of \tilde{A}_t^i or two or more odd entries of \tilde{A}_t^i then by property (II) M contains at least two identical rows and 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 non-zero entry. If M^{k+1} is all zeros, 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 $\ell = k + 1$ then $\det(M) \in \{-1, 0, 1\}$.

Suppose now that $\ell \leq k$. Then, by the definition of ℓ , columns $\ell + 1, \dots, k + 1$ 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 , $\ell + 1 \leq j \leq k$, cannot be located “southwest” of the nonzero elements of any M^p , $j + 1 \leq p \leq k + 1$. 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} , $j = \ell + 1, \dots, k$.

Consider now column ℓ . If M^ℓ contains a single replication of some a_i , then by the definition of ℓ we must have $i \neq 3$, 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}_j^i , and that at least one of those entries is located “northwest” of the vector a_3 in $M^{\ell+1}$. Then, that element has only zeros to its right. Next, let M_1 and M_2 denote the $k \times k$ matrices formed from M by deleting column $\ell + 1$ and deleting the row corresponding, respectively, to the first and second nonzero entry of $M^{\ell+1}$. By expanding the determinant over column $\ell + 1$, 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). Since $\det(M_i) \in \{-1, 0, 1\}$ 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 $M^{\ell+1}$. 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 zeros to its right, which can then be

analyzed using the argument in the previous paragraph):

$$M = \begin{pmatrix} D & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ Y & a_i & a_3 & & 0 & 0 & & 0 \\ Y & \vdots & 0 & \ddots & 0 & 0 & & 0 \\ Y & a_i & 0 & & a_3 & 0 & & 0 \\ E_1 & 0 & 0 & & 0 & a_3 & & 0 \\ \vdots & 0 & 0 & & 0 & 0 & \ddots & 0 \\ E_\eta & 0 & 0 & & 0 & 0 & & a_3 \\ G & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix}. \quad (18)$$

Here, D and G are certain (possibly empty) matrices with $\ell - 1$ columns, E_1, \dots, E_η are $2 \times (\ell - 1)$ matrices, and Y is a $2 \times (\ell - 1)$ 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 $j \neq 3$ 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 non-zero 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 $\ell + 1$ and deleting the row corresponding, respectively, to the first and second nonzero entry of $M^{\ell+1}$. 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 $M^{\ell+2}$ and the row corresponding to the j -th nonzero entry of $M^{\ell+2}$. By expanding over the columns $\ell + 1$ and $\ell + 2$ of M we obtain that

$$\det(M) = \pm[\det(M_{11}) - \det(M_{12}) - \det(M_{21}) + \det(M_{22})]. \quad (19)$$

When M has configuration (18), we can see that we have $\det(M_{11}) = \det(M_{22}) = 0$. Moreover, $\det(M_{12}) = -\det(M_{21})$, since M_{12} and M_{21} are equal except for the exchange of two rows. Hence, it follows from (19) that $\det(M) = 0$.

Case 3: $q \geq 3$. Here, *any* submatrix of M formed by deleting columns $M^{\ell+1}, \dots, M^{\ell+q}$ and *any* q rows will have at least two repeated rows. Hence, any such submatrix has determinant zero. Expanding over columns $M^{\ell+1}, \dots, M^{\ell+q}$, 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 . Since 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. \blacksquare

The above result shows that, in case of *two-leg* problems with single-seat bookings, the functions $v_t^R(r)$ and $v_t(r)$ coincide when $r \in \mathbb{Z}_+^n$. This is an important property, as it implies (from Proposition 3) that $v_t(r)$ is, in a sense, concave. A weaker result can be found in Theorem 3.3 of You (1999), which states that the value function for two-leg flights is componentwise concave. In addition, our proof, based upon the formulation of the problem as a MSSLP, is of independent interest because it uses completely different mathematical techniques. 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, \dots, a_3 defined in the proof of Theorem 1 are replaced by $a_0 = 0$, $a_1 = 1$. The proof then becomes much easier, since 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, and consequently need not be concave.

Example 1 The following deterministic example shows that for networks with three or more legs, the MDP value function is, in general, not concave. 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 concave. Furthermore, the value function also does not satisfy the weaker condition of componentwise concavity.

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 is concave, such property does not imply the 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.” It appears that the (non-)existence of such cycles is related to concavity of the value function v_t — indeed, we have not been able to find examples of acyclic networks where v_t is not concave (by “acyclic” we mean a network without directed or undirected cycles). However, 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 1st, 3rd and 5th 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. Moreover, even when the value function v_σ is concave the objective function in (12) is not concave or convex, since its second term is the composite of a nondecreasing concave function (v_σ) with a convex function ($c - A \min\{x, D_{(\sigma-1)}\}$). However, since 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 constraint $x \in \mathbb{Z}_+^n$ to $x \in \mathbb{R}_+^n$.

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

$$G(x, D) = f \cdot \min\{x, D_{(\sigma-1)}\} + v_\sigma(\lfloor c - A \min\{x, D_{(\sigma-1)}\} \rfloor), \quad (20)$$

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

$$\max \{g(x) : Ax \leq c, x \geq 0\}. \quad (21)$$

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, \dots, D^N are i.i.d. samples of $D_{(\sigma-1)}$, and then solve the approximating problem

$$\min_{x \in X} g_N(x). \quad (22)$$

The rationale is that, by the strong law of large numbers, $g_N(x) \rightarrow g(x)$ as $N \rightarrow \infty$ with probability one, and thus (22) should provide a good approximation to (21) for N large. This idea, of course, is not new, and has been object of study in different areas for many years. The convergence properties of such procedures are well-established; see, for instance, Shapiro (2002) 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 k th iteration of the algorithm we use the approximating function

$$\hat{g}_k(x) := \frac{G(x, D^{k,1}) + \dots + G(x, D^{k,N_k})}{N_k}, \quad (23)$$

where $D^{k,1}, \dots, D^{k,N_k}$ are i.i.d. the samples of $D_{(\sigma-1)}$, drawn at iteration k . In reality, there is no need to re-sample 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 re-sampling procedure tends to minimize the chance of obtaining a bad solution because of a “bad” sample path, which can occur in (22) 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 et al. (1999), who successfully apply these techniques to the optimization of release times of jobs in a single-line production environment.

Our procedure to solve the hybrid problem (21) 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 (21) is not differentiable, due to the rounding and min operations. In what follows, the operator ∇ denotes an approximate gradient (cf. section 7.1 below).

Since the optimization method we implement is very similar to the one used in Homem-de-Mello et al. (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 below 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, \dots, D^{N_k} ; compute the estimators $\hat{g}_k(x^k)$ (cf. (23)) and $\nabla \hat{g}_k(x^k)$ (cf. section 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, \dots, 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 $\tilde{D}^1, \dots, \tilde{D}^{N_{k+1}}$. 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 sub-optimal point. Thus, it is clear that such method provides only a heuristic approximation to (12). Nevertheless, the numerical results in section 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 $G^R(x, D)$ defined as G in (20), but with the relaxed function v_σ^R defined in (14) in place of v_σ . Let $\partial G^R(x, D)$ denote the *generalized gradient* of $G^R(\cdot, D)$ at x , and let $\nabla G^R(x, D)$ denote an arbitrary element of $\partial G^R(x, D)$. The generalized gradient can be viewed as an extension of the concept of 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

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

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

Using the above result, we can estimate ∇g^R via sampling, provided we can estimate ∇v_σ^R . The latter quantity can be approximated by *two-sided* finite differences, i.e.,

$$[\nabla v_\sigma^R(r)]_j \approx \frac{v_\sigma^R(r + e_j) - v_\sigma^R(r - e_j)}{2}. \quad (24)$$

Here, e_j is the vector with 1 in the j th component and zeros 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 (24) 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 (2000), though they work exclusively with one-sided derivatives.

8 Numerical Experiments

In this section we present some numerical examples that enable us to compare the hybrid allocation policy and some of its simpler “relatives” with certain other baseline heuristics as well as with optimal policies. In our tests (all of which use $\tau = 1000$), we consider three different types of hybrid 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.

The second type of hybrid policy that we consider is again based upon 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 of hybrid policy we test is an approximation to $\pi^{\text{H}(\sigma)}$, obtained from (21). Observe that for a particular switch time, $\pi^{\text{H}(\sigma)}$ 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, and LP(200)-MDP, since (12) does not consider policies that modify their allocations prior to the switch time. Since our main focus is the broad class of hybrid formulations and policies, we have not compared our results with other known policies, such as those based on nesting, for example. Nevertheless, our numerical results suggest that hybrid policies do appear to perform well.

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 (21), which approximates (12). Since 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, since the policy SP-MDP depends upon 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 hybrid 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 all the examples described below. 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 section 3. We then computed the average revenue for each policy over 1000 independent replications. For each of the 1000 simulation runs, the *same stream* of random numbers was 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 1001 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 ; e.g., 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 1–2. In Tables 1 and 3, there are two numbers in each cell. The first 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 1000 simulation runs of the policy in question. The second number is the half-width of a 98% confidence interval for the expected revenue. Notice that, by computing a 98% confidence interval for each policy, we ensure that the *overall* confidence on the intervals obtained for all five policies (for a particular switch time σ) is at least 90%; this follows immediately from Bonferroni’s inequality (see, e.g., Law and Kelton (2000)).

The purpose of Tables 2 and 4 is to compare each LP-based policy with the SP-MDP pol-

icity. Again, there are two numbers in each cell. The first number is an estimate of $\mathbb{E}[R^{\text{SP-MDP}} - R^{\text{LP}^{(n)\text{-MDP}}}]$, where $R^{\text{SP-MDP}}$ is the revenue obtained from the SP-MDP policy and $R^{\text{LP}^{(n)\text{-MDP}}}$ 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 $\mathbb{E}[R^{\text{SP-MDP}} - R^{\text{LP}^{(n)\text{-MDP}}}]$. 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.

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 classes. (We are now using class and itinerary in the more traditional sense.) The through itinerary fares are \$600, \$100, and \$75. The fares for each local itinerary are \$500, \$75, and \$50. There are $\tau = 1000$ time periods. The arrival probabilities for the highest (Class 1), second-highest (Class 2), and lowest (Class 3) class on each itinerary are given by $P(D_{t,(1,\ell)} = 1) = \alpha\beta_{(1,\ell)} \sin\left(\frac{t\pi}{2\tau}\right)$, $P(D_{t,(2,\ell)} = 1) = \alpha\beta_{(2,\ell)} \sin\left(\frac{t\pi}{\tau}\right)$, and $P(D_{t,(3,\ell)} = 1) = \alpha\beta_{(3,\ell)} \sin\left(\frac{\pi}{2} + \frac{t\pi}{2\tau}\right)$, where $D_{t,(k,\ell)}$ is the demand for class k , itinerary ℓ at time t . Observe that this means that high-class (Class 1) expected demand per time-unit grows as we get closer to time of departure, whereas low-class (Class 3) expected demand per time-unit falls as we approach time of departure. Class 2 expected demand per time-unit peaks in the middle of the booking process. In this example, we use $\alpha = .46$, $\beta_{(1,0)} = 0.05$, $\beta_{(2,0)} = 0.07$, $\beta_{(3,0)} = 0.08$, 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 $v_1(c) = 49737.23$, which does lie within the confidence interval shown in the $\sigma = 1$ row. Examination of Tables 1 and 2 shows that SP-

MDP appears to be slightly stronger than the other heuristics, except when $\sigma = 1001$. 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.

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 classes, 150 seats, and $\tau = 1000$. The through fares are \$1000, \$900, and \$100, and the local fares are \$800, \$700, and \$50. All class 2 and class 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 class 1 arrivals on any itinerary. After time 800, the class 1 arrival probabilities are constant at 0.0375. The expected total demand per leg is therefore 225. In Tables 3 and 4 and Figure 2, 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 142344.7. In this example, the difference between high-class fares and low-class 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 Some 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 upon 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 example, it took 11 seconds (on a 1.7GHz Pentium) to run the MDP code for a typical two-leg example such as the ones described in sections 8.2 and 8.3, with $\tau = 1000$ and 150 seats per leg. The hybrid method with switch time, say, $\sigma = 601$ — which typically “nearly” preserved the optimal value — took between 7 and 10 seconds for the stochastic optimization routine, plus 5 seconds for the MDP algorithm, which in this case was run from time 601 onwards. So, while there was no gain in computational time, there was an advantage in storage requirements, since the MDP routine needs to store the optimal decision rules.

The above comparison becomes 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. We verified this empirically, using a two-leg example with $\tau = 1000$ and 300 seats per leg. The MDP took 46 seconds to find the solution, as opposed to 11 seconds as before. The stochastic optimization routine, however, is largely unaffected by changes in capacity, since such a modification simply translates into a change on the right-hand side of the constraints. Again, we verified this conjecture numerically. Using $\sigma = 601$ as before, the stochastic optimization routine took the same 7-10 seconds as in the 150-seat case, whereas the MDP from time 600 onwards took 20 seconds. That is, the overall time for the hybrid method was 27-30 seconds, compared to 46 seconds to the full MDP — a savings of 40% in time, with even bigger savings in terms of memory requirements and little loss in the value of the objective function.

The time required by the simulation-based optimization routine to solve (12) depends upon 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 classes with very different fares but the same resource requirements arriving simultaneously, as in the example in section 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, since 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 to less frequent re-solving. This is not so surprising, since re-solving does incorporate new information into the optimization problem. The advantages are more noticeable for larger values of σ .

Another comment concerns the switch time σ . The example in section 8.2 suggests that, if we

switch to the MDP by time 401 or 601 (for the 1000 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 section 8.3, the switching time appears to be more critical for the LP(n)-MDP policies than for SP-MDP.

9 Conclusions

Many authors have considered variants of Markov decision process 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 a class of hybrid revenue management policies that combine aspects of MDP and math-programming approaches. We have presented 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 have adapted recently-developed variable-sample stochastic optimization techniques to solve a particular case of the formulation.

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

We have not considered overbooking, cancellations, or consumer choice behavior. Since MDP models that incorporate these effects do exist, see, e.g., Subramanian et al. (1999) and Talluri and van Ryzin (2001), 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 (1998). Other directions for future work include a theoretical study of the effect of the switch time on the revenue, as well as methods for addressing uncertainty in the forecasts of model parameters.

Acknowledgment

We would like to thank Dan Zhang for conducting numerical tests. This material is based upon work supported by the National Science Foundation under Grant No. DMI-0115385.

References

- D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 2nd edition, 1999.
- D. Bertsimas and S. de Boer. A stochastic booking-limit control policy for airline network revenue management. Working paper, Operations Research Center, Massachusetts Institute of Technology, 2000.
- D. Bertsimas and I. Popescu. Revenue management in a dynamic network environment. *Transportation Science*, 37:257–277, 2003.
- J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research. Springer-Verlag, New York, NY, 1997.
- G. Bitran and R. Caldentey. An overview of pricing models for revenue management. *Manufacturing & Service Operations Management*, 5:202–229, 2003.
- V. C. P. Chen, D. Günther, and E. L. Johnson. A Markov decision problem based approach to the airline YM problem. Working paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, 1998.
- F. H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, Inc., New York, 1983. Reprint by SIAM, Philadelphia, 1990.
- W. L. Cooper. Asymptotic behavior of an allocation policy for revenue management. *Operations Research*, 50:720–727, 2002.
- S. V. de Boer, R. Freling, and N. Piersma. Mathematical programming for network revenue management revisited. *European Journal of Operational Research*, 137:72–92, 2002.
- Y. Feng and B. Xiao. Optimal policies of yield management with multiple predetermined prices. *Operations Research*, 48:332–343, 2000.
- G. Gallego and G. van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science*, 40:999–1020, 1994.
- G. Gallego and G. van Ryzin. A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research*, 45:24–41, 1997.
- D. Günther. *Airline Yield Management: Optimal Bid Prices, Markov Decision Processes, and Routing Considerations*. PhD thesis, Georgia Institute of Technology, 1998.

- T. Homem-de-Mello, A. Shapiro, and M. L. Spearman. Finding optimal material release times using simulation based optimization. *Management Science*, 45:86–102, 1999.
- I. Karaesmen and G. J. van Ryzin. Overbooking with substitutable inventory classes. Working paper, Columbia University, Graduate School of Business, 1998.
- A. J. Kleywegt and J. D. Papastavrou. The dynamic and stochastic knapsack problem. *Operations Research*, 46:17–35, 1998.
- H. J. Kushner and G. G. Yin. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 1997.
- C. J. Lautenbacher and S. Stidham. The underlying Markov decision process in the single-leg airline yield management problem. *Transportation Science*, 33:136–146, 1999.
- A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, NY, 3rd. edition, 2000.
- T. C. Lee and M. Hersh. A model for dynamic airline seat inventory control with multiple seat bookings. *Transportation Science*, 27:252–265, 1993.
- J. I. McGill and G. J. van Ryzin. Revenue management: Research overview and prospects. *Transportation Science*, 33:233–256, 1999.
- A. Prékopa. *Stochastic Programming*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1995.
- M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, 1994.
- L. W. Robinson. Optimal and approximate control policies for airline booking with sequential nonmonotonic fare classes. *Operations Research*, 43:252–263, 1995.
- S. M. Ross. *Stochastic Processes*. John Wiley & Sons, New York, 1996.
- A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, New York, NY, 1986.
- A. Shapiro. Monte Carlo sampling methods. In A. Ruszczyński and A. Shapiro., editors, *Handbook of Stochastic Optimization*. Elsevier Science Publishers B.V., Amsterdam, Netherlands, 2002. (Forthcoming).
- A. Shapiro and T. Homem-de-Mello. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming*, 81:301–325, 1998.

- B. C. Smith, J. F. Leimkuhler, and R. M. Darrow. Yield management at American Airlines. *Interfaces*, 22:8–31, 1992.
- J. Subramanian, S. Stidham, and C. J. Lautenbacher. Airline yield management with overbooking, cancellations, and no-shows. *Transportation Science*, 33:147–168, 1999.
- K. Talluri and G. van Ryzin. An analysis of bid-price controls for network revenue management. *Management Science*, 44:1577–1593, 1998.
- K. Talluri and G. van Ryzin. A randomized linear programming method for computing network bid prices. *Transportation Science*, 33:207–216, 1999.
- K. Talluri and G. van Ryzin. Revenue management under a general discrete choice model of consumer behavior. Working paper, Universitat Pompeu Fabra, 2001.
- G. van Ryzin and G. Vulcano. Simulation-based optimization of virtual nesting controls for network revenue management. Working paper DRO-2003-01, Columbia University, Graduate School of Business, 2003.
- E. L. Williamson. *Airline Network Seat Control*. PhD thesis, Massachusetts Institute of Technology, 1992.
- P.-S. You. Dynamic pricing in airline seat management for flights with multiple flight legs. *Transportation Science*, 33:192–206, 1999.
- D. Zhang and W. L. Cooper. Revenue management for parallel flights with consumer choice behavior. Working paper, University of Minnesota, Department of Mechanical Engineering. Available at <http://www.menet.umn.edu/~billcoop/abstracts.html#RMPPF>, 2003.

σ	LP-MDP	LP(200)-MDP	LP(50)-MDP	LP(25)-MDP	SP-MDP
1	49906.03, 284.52				
201	49836.23, 279.31	49836.23, 279.31	49824.05, 278.48	49823.25, 278.35	49834.80, 279.39
401	49630.15, 277.72	49621.13, 277.72	49605.63, 276.56	49606.25, 276.58	49630.23, 277.89
601	49645.75, 279.28	49639.75, 279.28	49628.98, 277.29	49628.70, 277.37	49652.85, 280.35
801	49162.78, 254.82	49327.15, 254.82	49387.98, 259.97	49354.73, 258.77	49372.15, 279.73
1001	46743.98, 177.40	48340.28, 252.64	48971.40, 262.59	48910.63, 263.29	48512.18, 291.58

Table 1: Mean revenue as a function of switch time for the example in section 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
1001	1768.20, 137.14 +	171.90, 96.12 +	-459.23, 76.14 -	-398.45, 75.58 -

Table 2: Differences in revenue for the example in section 8.2.

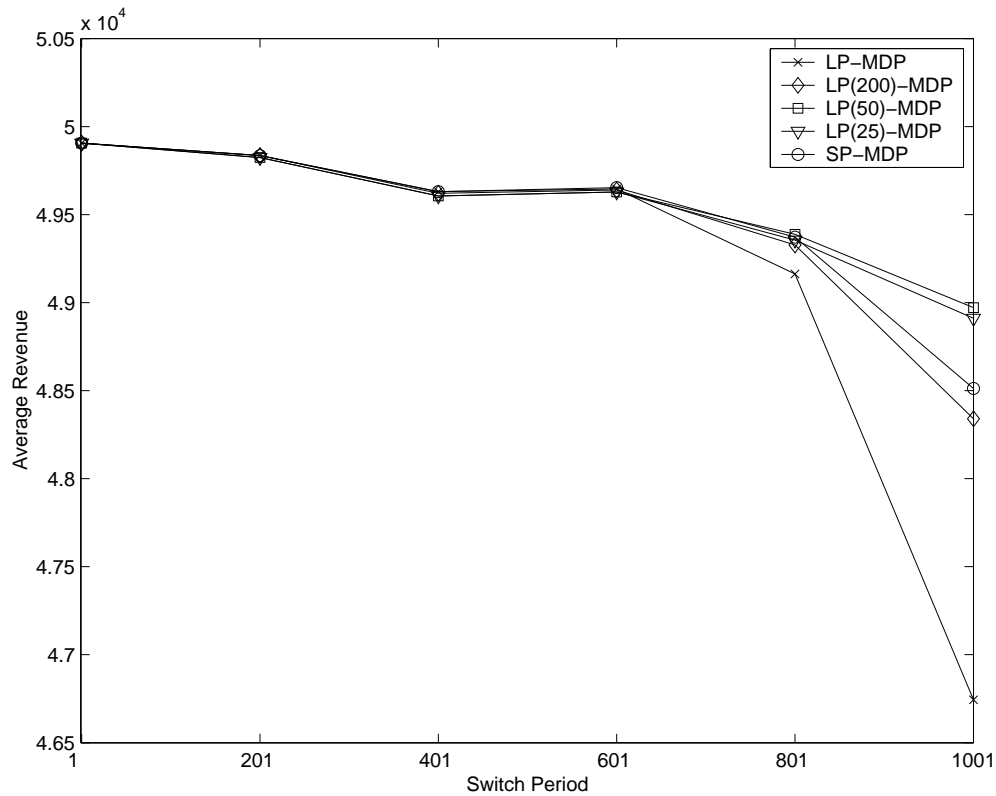


Figure 1: Mean revenue as a function of switch time for the example in section 8.2.

σ	LP-MDP	LP(200)-MDP	LP(50)-MDP	LP(25)-MDP	SP-MDP
1	142891.10, 627.77				
201	139799.80, 475.63	139799.80, 475.63	140244.50, 496.86	140181.10, 494.39	142187.55, 638.50
401	138112.40, 383.44	138405.10, 435.85	138311.35, 446.16	138212.70, 446.64	142703.10, 647.77
601	137821.10, 382.88	137479.80, 423.08	137336.25, 444.09	137246.85, 445.10	141834.40, 646.76
801	135730.60, 384.81	136119.10, 425.51	136353.75, 441.63	136500.55, 441.55	141299.40, 634.60
1001	134136.20, 383.28	134520.55, 428.40	135521.50, 454.54	135818.55, 452.32	139460.90, 599.69

Table 3: Mean revenue as a function of switch time for the example in section 8.3.

σ	LP-MDP	LP(200)-MDP	LP(50)-MDP	LP(25)-MDP
201	2387.75, 245.71 +	2387.75, 245.71 +	1943.05, 215.59 +	2006.45, 218.41 +
401	4590.70, 324.32 +	4298.00, 301.39 +	4391.75, 291.82 +	4490.40, 290.02 +
601	4013.30, 322.62 +	4317.25, 636.08 +	4498.15, 288.59 +	4587.55, 289.35 +
801	5568.80, 319.65 +	5180.30, 296.13 +	4945.65, 272.23 +	4798.85, 274.04 +
1001	5324.70, 263.10 +	4940.35, 254.24 +	3939.40, 243.12 +	3642.35, 242.56 +

Table 4: Differences in revenue for the example in section 8.3.

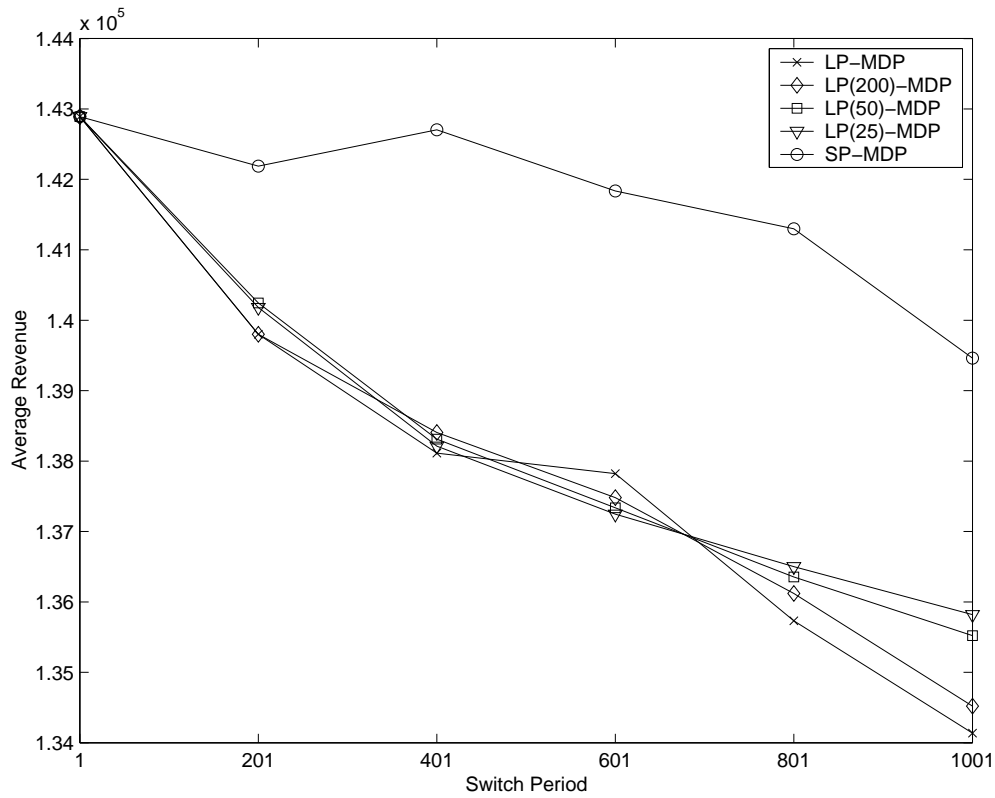


Figure 2: Mean revenue as a function of switch time for the example in section 8.3.