M&SOM MANUFACTURING & SERVICE OPERATIONS MANAGEMENT
Vol. 13, No. 1, Winter 2011, pp. 35–52
ISSN 1523-4614 | EISSN 1526-5498 | 11 | 1301 | 0035

informs

DOI 10.1287/msom.1100.0302

An Improved Dynamic Programming Decomposition Approach for Network Revenue Management

Dan Zhang
Desautels Faculty of Management, McGill University, Montreal, Quebec H8N 3E9, Canada,
dan.zhang@mcgill.ca

We consider a nonlinear nonseparable functional approximation to the value function of a dynamic programming formulation for the network revenue management (RM) problem with customer choice. We propose a simultaneous dynamic programming approach to solve the resulting problem, which is a nonlinear optimization problem with nonlinear constraints. We show that our approximation leads to a tighter upper bound on optimal expected revenue than some known bounds in the literature. Our approach can be viewed as a variant of the classical dynamic programming decomposition widely used in the research and practice of network RM. The computational cost of this new decomposition approach is only slightly higher than the classical version. A numerical study shows that heuristic control policies from the decomposition consistently outperform policies from the classical decomposition.

Key words: network revenue management; choice behavior; multinomial logit choice model; dynamic programming
History: Received: April 14, 2009; accepted: March 27, 2010. Published online in *Articles in Advance* October 13, 2010.

1. Introduction

Revenue management (RM) entails controlling the availability/pricing of different products that use the same set of resources in order to maximize revenue. Traditional application areas include airlines, hotels, and car rentals. The application has expanded to many other areas in recent years (Talluri and van Ryzin 2004b). One problem that has received considerable attention in the research and application of RM is the network RM problem (Gallego and van Ryzin 1997), where customer requests with varying resource requirements and revenue contributions are controlled dynamically in order to maximize revenue. A canonical example is the application in the airline industry, where the resources correspond to seats on scheduled flights and products refer to different fare-itinerary combinations. Network RM techniques have also been widely used in other application areas.

It is well known that the dynamic programming formulation of network RM suffers from the curse of dimensionality as the state space of such a formulation grows exponentially with the number of resources involved (see, e.g., Bertsimas and Popescu 2003). Dynamic programming decomposition, where a high-dimensional dynamic program is decomposed into a set of dynamic programs with small one-dimensional state space, is a powerful idea that has been exploited extensively in the research and application of network RM. For a review on decomposition-based approximations for network RM, see Talluri and van Ryzin (2004b, §3.4). Recently, Liu and van Ryzin (2008) extend the dynamic programming decomposition approach to network RM with customer choice. Their work inspired a stream of research that considers refined solution approaches of the same problem. (We defer a more detailed literature review to the next section.)

Traditional formulations of network RM often adopt the independent demand by fare class (independent demand for short) assumption where each arriving customer belongs to a specific fare class requesting a particular product (fare-itinerary combination). Recently, many authors have considered models that explicitly take into account customer choice behavior among the available fare products. Such models reflect the trend toward more realistic customer behavior modeling. With the increasing availability of product information and decreasing search cost in many RM settings, customers are empowered to make smarter choices that fit their needs. We choose to discuss our main results in the setting that considers customer choice behavior. To further facilitate our discussion, we will present our work in the airline RM context. We do point out that because the independent demand model can be cast as a customer choice model where each customer chooses a particular product with probability

one and other products with probability zero, the main theoretical results in this paper easily extend to that setting.

Despite the popularity of dynamic programming decomposition in RM research and practice, surprisingly few theoretical results are found in the literature. Zhang and Adelman (2009) show that a particular dynamic programming decomposition leads to an upper bound on the optimal expected revenue, which is tighter than the bound from a deterministic linear program called a choice-based deterministic linear program (CDLP); see Liu and van Ryzin (2008).

Zhang and Adelman (2009) follow the earlier work of Adelman (2007), which considers a linear functional approximation for network RM in an independent demand setting. In his approach, the dynamic programming value function for each time period is approximated by an affine function of the state, which refers to the vector of remaining inventories for all resources. The (time-dependent) coefficients for each state component can be interpreted as dynamic bid prices, which represent the estimated opportunity costs for resources. Such dynamic bid-prices can be used in the optimality equation of the dynamic programming formulation to generate control policies. Zhang and Adelman (2009) show that a dynamic programming decomposition approach using such dynamic bid-prices leads to an upper bound on the optimal expected revenue. However, their decomposition bound is more general. In fact, the bound holds for any bid-price vector; in particular, it holds for a static bid-price vector that is time invariant.

Motivated by the decomposition bound in Zhang and Adelman (2009), this paper explores a stronger form of functional approximation that is nonlinear and nonseparable. To the best of our knowledge, our work is the first in the RM literature to consider such a strong form of functional approximation. This new functional approximation leads to a nonlinear optimization problem with a large number of nonlinear constraints, and is potentially very difficult to solve. We show that a restricted version of the optimization problem can be efficiently solved via a simultaneous dynamic programming algorithm. The computational cost of the algorithm is only slightly higher than that of the dynamic programming decomposition presented in Liu and van Ryzin (2008). Because their decomposition approach is very similar in spirit to existing dynamic programming decomposition approaches in the literature, we refer to their approach as the classical dynamic programming decomposition. Furthermore, the new decomposition leads to a revenue bound that is tighter than the decomposition bound presented in Zhang and Adelman (2009).

It is possible to combine the approach proposed in this paper with approaches that generate "better" bid prices. This paper uses static bid-prices from CDLP. An immediate extension is to use bid-prices from other approaches, such as the dynamic bid-prices from Zhang and Adelman (2009). All the theoretical results in this paper carry over. A downside of using such an approach is that the computational cost would be much higher because it is quite computationally intensive to compute dynamic bid-prices.

For ease of reference, we will refer to the classical dynamic programming decomposition as DCOMP, and the one presented in this paper as DCOMP1. We will also refer to the heuristic policies from the two decomposition schemes as DCOMP and DCOMP1, respectively. We will make it clear from the context when we refer to bounds versus policies.

We test the performance of DCOMP1 on a set of randomly generated problem instances with different load factors and time horizons. Our simulation results show that DCOMP1 outperforms DCOMP in almost all test cases. In particular, it can outperform DCOMP by as much as 8% even in relatively large problem instances, which is quite remarkable given that DCOMP is often believed to be one of the strongest benchmarks available and is often the algorithm implemented in commercial applications. For comparison purposes, we also implement a policy that uses static bid-prices from CDLP, which we call CDLP. We show that overall DCOMP1 performs better than CDLP even when the latter is frequently resolved.

Our numerical results show that bounds from DCOMP1 can be much tighter than bounds from DCOMP, providing better benchmarks in numerical studies. However, even when DCOMP1 bounds are close to DCOMP bounds, the corresponding policy can still outperform those from DCOMP. We show that the DCOMP1 bounds from individual legs are more homogeneous than those from DCOMP, which we believe is the source of strength of the policies. This stems from the enhanced ability of DCOMP1 to capture network effects through its stronger functional approximation.

The customer choice model considered in our numerical study is the multinomial logit model with disjoint consideration sets (MNLD) that was originally introduced in Liu and van Ryzin (2008). The choice model was subsequently considered in several follow-up papers (Zhang and Adelman 2009, Meissner and Strauss 2009, Kunnumkal and Topaloglu 2008b). The approach presented in this paper can be extended to more general choice models, such as the multinomial logit choice model with overlapping segments introduced in Bront et al.

(2009). However, we believe that considering more complex forms of choice models digresses from the main purpose of the paper. We conjecture that a similar revenue lift can be observed for more general choice models because the main driver of the performance improvement is the stronger form of functional approximation that captures network effects better.

The main contribution of the present paper is the new decomposition scheme DCOMP1, which is based on a strong form of functional approximation of the dynamic programming formulation of the network RM with customer choice. The functional approximation employed can capture network effects better than classical dynamic programming decomposition to the same problem. To the best of our knowledge, our work is the first that adopts a nonlinear nonseparable functional approximation, which is shown to be efficiently solvable through a simultaneous dynamic programming algorithm. We show that the new decomposition approach leads to a better revenue bound than does the classical version. Coming up with tighter revenue bounds is interesting in its own right (Talluri 2008). A computational study demonstrates the strength of the proposed approach.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature, and §3 formulates the problem and reviews some preliminary results. The core of the paper is §4, which introduces the new dynamic programming decomposition approach. Section 5 introduces several heuristic policies tested in our numerical study. Section 6 reports computational results, and §7 summarizes.

2. Literature Review

Our work is closely related to the RM and approximate dynamic programming literature. For a comprehensive review of the RM literature, see Talluri and van Ryzin (2004b). Comprehensive references on approximate dynamic programming are offered by Bertsekas and Tsitsiklis (1996) and Powell (2007).

The problem considered in this paper builds on the growing literature on revenue management with customer choice, and draws heavily on the recent literature that considers such problems in network settings. Researchers in RM have long realized that demand for different products may be dependent through customer diversion and upgrading. Brumelle et al. (1990) consider seat allocations for a two-class single-leg RM problem when the demands for the two classes are stochastically dependent because of customer buy-up. Belobaba and Weatherford (1996) investigate variations of some well-known heuristics to account for customer diversion among customer classes. Zhao and Zheng (2001) consider a two-class seat allocation model with passenger diversion. Much of the early work considers problems that involve a single resource and relatively simple customer behavior models. Talluri and van Ryzin (2004a) consider customer choice among fare classes on a single-leg flight for a quite general discrete-choice model. Their work paved the way for later developments that consider more general choice models and more complex network structure.

Zhang and Cooper (2005, 2009) consider seat allocation and pricing issues for multiple flights sharing the same origin and destination. Van Ryzin and Vulcano (2008) introduce a simulation-based optimization approach for network RM under a fairly general customer choice process. Gallego et al. (2004) propose a linear programming approach to analyze network RM for flexible products that was subsequently adopted by Liu and van Ryzin (2008) to study network RM with customer choice. They also extend the classic dynamic programming decomposition approach (see Talluri and van Ryzin 2004b, §3.4) to the network setting with customer choice. To demonstrate the strength of the approach, they consider problem instances where customer choice follows a MNLD.

The line of research on network RM with customer choice has been expanding. Bront et al. (2009) extend the work of Liu and van Ryzin (2008) to allow for overlapping consideration sets in the multinomial logit choice model. They adopt a dynamic programming decomposition approach similar to the one in Liu and van Ryzin (2008). Their paper provides a heuristic solution approach for the CDLP formulation under the more general choice model, which was empirically shown to perform well. Jiang and Miglionico (2006) consider a network RM problem with customer buy-up and explore several solution approaches. Zhang and Adelman (2009) adopt a linear functional approximation approach to generate dynamic bid prices that are subsequently used in a dynamic programming decomposition. Their work builds on the earlier work of Adelman (2007), which considers functional approximation for dynamic programming formulation of the network RM problem. Kunnumkal and Topaloglu (2008b) solve the network RM with customer choice using a Lagrangian relaxation approach to approximate dynamic programming. Kunnumkal and Topaloglu (2008a) propose a dynamic programming decomposition that aims at finding better fare proration than that used in Liu and van Ryzin (2008). Walczak (2008) introduces a product-based decomposition idea. Meissner and Strauss (2009) consider a separable concave functional approximation for the network RM with customer choice. They propose inventory aggregation as a way to reduce the computational burden and

show some theoretical results. Their work is related to the earlier work of Farias and Van Roy (2007), who posit a separable concave functional approximation for network RM without customer choice. Farias and Van Roy (2007) adopt a constraint sampling approach to solve the problem. With few exceptions, the MNLD choice model is considered in the majority of papers in this stream of research, including Zhang and Adelman (2009), Kunnumkal and Topaloglu (2008a, b), and Meissner and Strauss (2009). We also consider the MNLD choice model in the present paper.

The solution approach adopted in this paper is closely related to the functional approximation idea for approximate dynamic programming that was first considered by Schweitzer and Seidmann (1985). de Farias and Van Roy (2003) consider linear-programming based approaches to approximate dynamic programming. Adelman (2007) proposes the use of weighted basis functions as a way to approximately solve high-dimensional dynamic programming formulation of network RM. He demonstrates the strength of the approach using a linear functional approximation, which is a special case of the weighted basis function approach. Other papers that consider functional approximation approaches include the earlier-cited papers of Zhang and Adelman (2009), Farias and Van Roy (2007), and Meissner and Strauss (2009). For an example of a functional approximation approach for infinite-horizon dynamic programs, see Adelman and Mersereau (2008).

This paper considers a functional approximation that is nonlinear and nonseparable. To the best of our knowledge, our work is the first that considers such an approximation in the RM literature. We propose a simultaneous dynamic programming approach to solve the resulting problem. Compared with some of the existing work, the benefit of the proposed approach lies in its relatively low computational cost. Indeed, our numerical study shows that the computational time required is only slightly higher than the classical dynamic programming decomposition approach proposed in Liu and van Ryzin (2008). When computational time is not a concern, our approach can be combined with other approaches in the literature—for example, Zhang and Adelman (2009) and Meissner and Strauss (2009)—to generate potentially even stronger heuristics. Our results confirm, once again, the effectiveness of functional approximation as a useful approach for approximate dynamic programming in network RM and other applications. In addition, our approach leads to a better bound, which is subsequently used as the benchmark in our numerical study.

3. Model Formulation and Preliminaries

3.1. Dynamic Programming Formulation

This section formulates the network RM problem with customer choice as a finite-horizon dynamic program. The original problem was introduced in Liu and van Ryzin (2008). Our formulation and notations closely follow those of Zhang and Adelman (2009). For ease of exposition, we use airline terminology throughout the paper.

Consider a flight network with m legs and the set of capacities $c = (c_1, \ldots, c_m)^{\mathrm{T}}$, where c_i is the capacity of leg i and the superscript T denotes vector transpose. There are n products offered, where a product is an itinerary-fare combination. Let $N = \{1, \ldots, n\}$ be the set of products. The fare for product j is f_j. The consumption matrix is an $(m \times n)$-matrix $A \equiv (a_{ij})$. The entry $a_{ij} \in \{0, 1\}$ represents the amount of resource i required by a class j customer. By restricting A to a binary matrix, we explicitly rule out group requests that can use multiple units of a resource. This restriction is made primarily for convenience and can be relaxed for much of our analysis. The jth column A_j is the resource incidence vector for product j. There are τ discrete time periods that are counted forward, so period τ is the last period. The period $\tau + 1$ is used to denote the end of the horizon. To simplify notation, we reserve the symbols i, j, and t for legs, products, and time, respectively.

In each period, there is one customer arrival with probability λ, and no customer arrival with probability $1 - \lambda$. Note that we assume that λ is independent of time t only for notational simplicity. When a customer arrives, the firm must decide what products to offer. Let $S \subseteq N$ be the offer set of the firm. Note that $S = \varnothing$ means that no product is offered. Given offer set S, each arriving customer chooses the product $j \in S$ with probability $P_j(S)$, and makes no purchase with probability $P_0(S) = 1 - \sum_{j \in S} P_j(S)$. Note that the independent demand model where each customer requests a specific product but does not choose among the products can be modeled as a special case where $P_j(S) = \bar{p}_j \forall\, j \in S$ and $P_j(S) = 0$ otherwise.

The state at the beginning of any period t is an m-vector of unsold seats x. Let $v_t(x)$ be the maximum total expected revenue over periods $t, \ldots, \tau$ starting at state x at the beginning of period t. The optimality equation is

$$\text{(DP)} \quad v_t(x) = \max_{S \subseteq N(x)} \left\{ \sum_{j \in S} \lambda P_j(S)(f_j + v_{t+1}(x - A_j)) + (\lambda P_0(S) + 1 - \lambda) v_{t+1}(x) \right\}$$

$$= \max_{S\subseteq N(x)} \left\{ \sum_{j\in S} \lambda P_j(S)[f_j - \Delta_j v_{t+1}(x)] \right\} + v_{t+1}(x), \quad \forall\, t, x, \tag{1}$$

where $\Delta_j v_{t+1}(x) = v_{t+1}(x) - v_{t+1}(x - A_j)$ represents the opportunity cost of selling product j in period t. The boundary conditions are $v_{\tau+1}(x) = 0$ for all x. In the above, the set $N(x) = \{j \in N\colon x \geq A_j\}$ is the set of products that can be offered when the state is x.

The formulation DP suffers from the curse of dimensionality because the state space grows exponentially with the number of resources m. Therefore, it is difficult to solve the model even for moderately sized problems. The primary focus of the existing research is to find tractable approximations that are easily solvable. In the following, we briefly review the choice-based deterministic linear programming (CDLP) model and the classical dynamic programming decomposition approach introduced in Liu and van Ryzin (2008).

3.2. The Choice-Based Linear Program

The (CDLP) model can be viewed as an extension of the deterministic linear program (see, e.g., Talluri and van Ryzin 1998, Cooper 2002) for network RM to the setting with customer choice. The basic idea is to approximate random discrete quantities with deterministic continuous quantities. In the formulation, both resources (seats) and demands are viewed as continuous.

Let S denote the firm's offer set. Customer demand flows in at rate λ. When product set S is offered, product j is sold at rate $\lambda P_j(S)$ (i.e., a proportion $P_j(S)$ of the demand chooses product j). Let $R(S)$ denote the revenue from one unit of customer demand when the set S is offered. Then, $R(S) = \sum_{j\in S} f_j P_j(S)$. Note that $R(S)$ is a scalar. Similarly, let $Q_i(S)$ denote the resource consumption rate on flight i, $i = 1, \ldots, m$, given that the set S is offered. Let $Q(S) = (Q_1(S), \ldots, Q_m(S))^{\mathrm{T}}$. (The superscript T is used to denote vector transpose.) The vector $Q(S)$ satisfies $Q(S) = AP(S)$, where $P(S) = (P_1(S), \ldots, P_n(S))^{\mathrm{T}}$ is the vector of purchase probabilities.

Let $h(S)$ be the total time that the set S is offered. Because the demand is deterministic and the choice probabilities are time homogeneous, the order in which different offer sets are used does not matter. Consequently, only the total time that each product set is offered matters. The objective is to determine the total time $h(S)$ during which each set S should be offered to maximize the firm's revenue. The choice-based deterministic linear program can be written as follows:

$$\text{(CDLP)} \quad z_{\mathrm{CDLP}} = \max_{h} \sum_{S\subseteq N} \lambda R(S) h(S)$$
$$\sum_{S\subseteq N} \lambda Q(S) h(S) \leq c, \tag{2}$$
$$\sum_{S\subseteq N} h(S) \leq \tau, \tag{3}$$
$$h(S) \geq 0, \quad \forall\, S \subseteq N.$$

In the above, the constraint (2) is a resource constraint. (All vector inequalities should be interpreted componentwise.) The constraint (3) stipulates that the total time of all product offerings cannot exceed the available time τ. The dual values associated with the constraint (2) can be interpreted as the value of an additional unit of each resource, which can be used as a bid-price for the resource or used to construct other types of heuristics, such as the dynamic programming decomposition that we introduce next. Liu and van Ryzin (2008) show that the objective value of CDLP, z_{CDLP}, provides an upper bound on the optimal expected revenue; i.e., $z_{\mathrm{CDLP}} \geq v_1(c)$.

As suggested in Liu and van Ryzin (2008), the formulation CDLP can be extended to consider time-dependent arrival rates and choice probabilities by dividing the booking horizon into intervals, each with constant arrival rates and choice probabilities and defining the decision variables for each interval separately. We restrict ourselves to stationary problem parameters for convenience of presentation.

3.3. The Classical Dynamic Programming Decomposition

Liu and van Ryzin (2008) introduce a dynamic programming decomposition scheme that uses the dual values π^* on constraint (2) in CDLP. The main idea is to decompose the network problem that involves many resources into single-resource problems that are much easier to solve. Let π^* be the vector of dual values of the resource constraint in CDLP. Consider the following approximation for any fixed leg i:

$$v_t(x) \approx v_{t,i}(x_i) + \sum_{k\neq i} \pi_k^* x_k, \quad \forall\, t, x. \tag{4}$$

In (4), the value of having x_i seats in period t on leg i is approximated by the nonlinear term $v_{t,i}(x_i)$, whereas the value of each seat on leg k is approximated by π_k^* for resources $k \neq i$.

Using (4) in DP leads to the following dynamic program for each leg i:

$$\text{(DP}_i\text{)} \quad v_{t,i}(x_i) = \max_{S\subseteq N(x_i, c_{-i})} \left\{ \sum_{j\in S} \lambda P_j(S) \left[f_j - \sum_{k\neq i} a_{kj}\pi_k^* - (v_{t+1,i}(x_i) - v_{t+1,i}(x_i - a_{ij})) \right] \right\} + v_{t+1,i}(x_i), \quad \forall\, t, x_i.$$

Here, the vector (x_i, c_{-i}) refers to the vector c with the ith component replaced by x_i. The boundary conditions are $v_{\tau+1,i}(x_i) = 0$ for all x_i. Note that the term

$f_j - \sum_{k \neq i} a_{kj} \pi_k^*$ can be interpreted as prorated fare for leg i, because it reduced the fare f_j by the "value" of seats on other resources.

The dynamic program (DP_i) has a one dimensional state space and therefore dispels the curse of dimensionality faced by (DP). After $v_{t,i}(\cdot)$ is computed for each i, the value function $v_t(x)$ of (DP) can be approximated by $v_t(x) \approx \sum_{i=1}^m v_{t,i}(x_i)$, which can be plugged back in the recursion in (DP) to compute an approximate control policy. Therefore, as long as (CDLP) and (DP_i) are efficiently solvable, the above dynamic programming decomposition can be used to generate a control policy. Recently, Zhang and Adelman (2009) show that the approximation (4) leads to an upper bound, which is tighter than the upper bound from (CDLP).

Proposition 1 (Zhang and Adelman 2009). *The following relationships hold*

(i) $v_t(x) \leq \min_{l=1,\ldots,m}\{v_{t,l}(x_l) + \sum_{k \neq l} \pi_k^* x_k\} \leq v_{t,i}(x_i) + \sum_{k \neq i} \pi_k^* x_k$, $\forall\, i, t, x$;

(ii) $v_1(c) \leq v_{1,i}(c_i) + \sum_{k \neq i} \pi_k^* c_k \leq z_{\mathrm{CDLP}}$, $\forall\, i$.

Part (i) in Proposition 1 shows that the dynamic programming decomposition produces a statewise bound for each time period. Part (ii) shows that it produces a tighter upper bound than (CDLP). According to (4), the value $v_t(x)$ is approximated by the value from leg i and the values from the other legs. The network effect is captured purely by the fare proration in (DP_i), and the approximate values $v_{t,i}(\cdot)$ are calculated independent of all other legs. Hence, in general, the right-hand side of (4) is not the same for different resources.

4. A New Dynamic Programming Decomposition Scheme

In this section, we introduce a novel dynamic programming decomposition scheme from a strong functional approximation motivated by the decomposition bound in Proposition 1. The new functional approximation leads to a constrained nonlinear optimization problem that is, in general, very hard to solve. We show that a restricted version of the problem can be efficiently solved via a simultaneous dynamic programming algorithm. Furthermore, this restricted version of the problem produces a tighter upper bound than the bounds introduced in Proposition 1.

4.1. A Strong Functional Approximation

We first note that (DP) can be written as a linear program as follows (see Zhang and Adelman 2009):

$$\text{(LP)} \quad \min_{\{v_t(\cdot)\}_{\forall t}} \; v_1(c)$$

$$v_t(x) \geq \sum_{j \in S} \lambda P_j(S)(f_j + v_{t+1}(x - A_j)) + (\lambda P_0(S) + 1 - \lambda) v_{t+1}(x), \quad \forall\, t, x,\ S \subseteq N(x),$$

where we take $v_{\tau+1}(x) = 0$ for all x. Note that, in (LP), the decision variables are the dynamic programming value functions $v_t(\cdot)$ for all t. The formulation (LP) is equivalent to (DP) in the sense that it gives exactly the same value function when solved to optimality. This should be contrasted with (CDLP), which results from a deterministic approximation. Clearly, (LP) involves many decision variables and constraints, and therefore is as difficult to solve as the original formulation (DP). It can be shown by induction that any feasible solution to (LP) gives an upper bound for the value function in (DP).

One strategy to reduce the number of decision variables is to use a functional approximation for $v_t(\cdot)$. This approach was taken by Zhang and Adelman (2009), who use a linear functional approximation of the form

$$v_t(x) \approx \theta_t + \sum_{i=1}^m V_{t,i} x_i, \quad \forall\, t, x. \tag{5}$$

The set of values $\{V_{t,i}\}_{\forall t,i}$ can be interpreted as dynamic bid-prices because $V_{t,i}$ represents the estimated value of a seat on leg i in period t. The θ_ts are adjusting constants. The authors develop a solution strategy for the resulting linear program with decision variables $V_{t,i}$ and θ_t. The classical dynamic programming decomposition approach can be viewed as a functional approximation approach with the approximation scheme (4), where the decision variables are $\{v_{t,i}(\cdot)\}_{\forall t,i}$. Note that the approximation (5) is linear separable, and the approximation (4) is separable.

We introduce another functional approximation scheme motivated by the decomposition bound in Proposition 1. Given a vector π^* of resource dual prices and a collection of single-dimensional value functions $\{\hat{v}_{t,i}(\cdot)\}_{\forall t,i}$, we consider the functional approximation

$$v_t(x) \approx \min_i \left\{ \hat{v}_{t,i}(x_i) + \sum_{k \neq i} \pi_k^* x_k \right\}, \quad \forall\, t, x. \tag{6}$$

In the above, $\hat{v}_{t,i}(x_i)$ is a nonlinear term representing the value of x_i seats on leg i. Observe that the approximation (6) is nonlinear and nonseparable in x due to the minimization. The approximation (6) is able to better capture the network effect, because after $\hat{v}_{t,i}(\cdot)$s are determined for each leg i, the value $v_t(x)$ is approximated by a single minimum across the legs, whereas in the decomposition in §3.3, the network effect is only captured through fare proration.

Plugging (6) into (LP), we have

$$\text{(NLP)} \quad z_{\mathrm{NLP}} = \min_{\hat{v}_{t,i}(\cdot)\ \forall t,i} \min_i \left\{ \hat{v}_{1,i}(c_i) + \sum_{k \neq i} \pi_k^* c_k \right\}$$

$$\min_i \left\{ \hat{v}_{t,i}(x_i) + \sum_{k \neq i} \pi_k^* x_k \right\}$$

$$\geq \sum_{j \in S} \lambda P_j(S) \left(f_j + \min_l \left\{ \hat{v}_{t+1,l}(x_l - a_{lj}) + \sum_{k \neq l} \pi_k^* (x_k - a_{kj}) \right\} \right)$$

$$+(\lambda P_0(S)+1-\lambda)\min_l\left\{\hat{v}_{t+1,l}(x_l)+\sum_{k\neq l}\pi_k^* x_k\right\},$$
$$\forall\, t, x,\ S\subseteq N(x). \quad (7)$$

Being a nonlinear program with, potentially, a huge number of nonlinear constraints, the problem (NLP) is in general extremely difficult to solve. The primary difficulty in using strong functional approximations in approximate dynamic programming is precisely the difficulty associated with solving the resulting problem. In the remainder of this section, we focus on developing an approximate solution procedure for (NLP). We will show that this solution procedure leads to tighter bounds and approximations than the solution in §3.3.

First, observe that the minimization on the left-hand side of (7) can be removed by writing each constraint as m equivalent constraints

$$\hat{v}_{t,i}(x_i)+\sum_{k\neq i}\pi_k^* x_k$$
$$\geq\sum_{j\in S}\lambda P_j(S)\left(f_j+\min_l\left\{\hat{v}_{t+1,l}(x_l-a_{lj})+\sum_{k\neq l}\pi_k^*(x_k-a_{kj})\right\}\right)$$
$$+(\lambda P_0(S)+1-\lambda)\min_l\left\{\hat{v}_{t+1,l}(x_l)+\sum_{k\neq l}\pi_k^* x_k\right\},$$
$$\forall\, i, t, x,\ S\subseteq N(x). \quad (8)$$

The constraint (8) is still difficult to handle. In the following, we consider a restriction of (8) that renders the resulting problem efficiently solvable via a simultaneous dynamic programming approach.

By moving the second term on the left-hand side to the right-hand side, the constraint (8) can be written as

$$\hat{v}_{t,i}(x_i)\geq\sum_{j\in S}\lambda P_j(S)\left(f_j+\min_l\left\{\hat{v}_{t+1,l}(x_l-a_{lj})+\sum_{k\neq l}\pi_k^*(x_k-a_{kj})-\sum_{k\neq i}\pi_k^* x_k\right\}\right)$$
$$+(\lambda P_0(S)+1-\lambda)\cdot\min_l\left\{\hat{v}_{t+1,l}(x_l)+\sum_{k\neq l}\pi_k^* x_k-\sum_{k\neq i}\pi_k^* x_k\right\},$$
$$\forall\, i, t, x,\ S\subseteq N(x).$$

Simplifying the above leads to

$$\hat{v}_{t,i}(x_i)\geq\sum_{j\in S}\lambda P_j(S)\left(f_j+\min_l\left\{\hat{v}_{t+1,l}(x_l-a_{lj})-\pi_l^* x_l-\sum_{k\neq l}\pi_k^* a_{kj}+\pi_i^* x_i\right\}\right)$$
$$+(\lambda P_0(S)+1-\lambda)\cdot\min_l\{\hat{v}_{t+1,l}(x_l)-\pi_l^* x_l+\pi_i^* x_i\},$$
$$\forall\, i, t, x,\ S\subseteq N(x).$$

Breaking up the minimization terms on the right-hand side leads to

$$\hat{v}_{t,i}(x_i)\geq\sum_{j\in S}\lambda P_j(S)\left(f_j+\min\left\{\hat{v}_{t+1,i}(x_i-a_{ij})-\sum_{k\neq i}\pi_k^* a_{kj},\ \min_{l\neq i}\left\{\hat{v}_{t+1,l}(x_l-a_{lj})-(x_l-a_{lj})\pi_l^*-\sum_k a_{kj}\pi_k^*+\pi_i^* x_i\right\}\right\}\right)$$
$$+(\lambda P_0(S)+1-\lambda)\min\left\{\hat{v}_{t+1,i}(x_i),\ \min_{l\neq i}\{\hat{v}_{t+1,l}(x_l)-\pi_l^* x_l+\pi_i^* x_i\}\right\},$$
$$\forall\, i, t, x,\ S\subseteq N(x).$$

Next, we restrict the constraint such that, for each fixed i, the constraint only involves x_i, which can be achieved by taking the maximum over x_k for all $k\neq i$ for each fixed i. With a little more algebra, this leads to

$$\hat{v}_{t,i}(x_i)\geq\sum_{j\in S}\lambda P_j(S)\left(f_j+\min\left\{\hat{v}_{t+1,i}(x_i-a_{ij})-\sum_{k\neq i}\pi_k^* a_{kj},\ \min_{l\neq i}\left\{\max_{0\leq y_l\leq c_l-a_{lj}}[\hat{v}_{t+1,l}(y_l)-y_l\pi_l^*]-\sum_k a_{kj}\pi_k^*+\pi_i^* x_i\right\}\right\}\right)$$
$$+(\lambda P_0(S)+1-\lambda)\min\left\{\hat{v}_{t+1,i}(x_i),\ \min_{l\neq i}\left\{\max_{0\leq y_l\leq c_l}[\hat{v}_{t+1,l}(y_l)-\pi_l^* y_l]+\pi_i^* x_i\right\}\right\},$$
$$\forall\, i, t, x,\ S\subseteq N(x).$$

Recognizing that $N(x)\subseteq N(x_i, c_{-i})$ with c_{-i} being the vector c without the ith component, the above constraint can be further restricted by taking $S\subseteq N(x_i, c_{-i})$ instead of $S\subseteq N(x)$. Note that this is a constraint restriction because for each fixed i, t, x, replacing $N(x)$ with $N(x_i, c_{-i})$ adds constraints for all $S\subseteq N(x_i, c_{-i})\backslash N(x)$. After this step, we can remove redundant constraints by replacing x with x_i for each i, because the constraints for each i do not involve other components of x except x_i. For convenience of reference, we rewrite (NLP) with the restricted constraint as

$$(\widehat{\text{NLP}})\quad z_{\widehat{\text{NLP}}}=\min_{\hat{v}_{t,i}(\cdot)\ \forall\, t,i}\ \min_i\left\{\hat{v}_{1,i}(c_i)+\sum_{k\neq i}\pi_k^* c_k\right\}$$
$$\hat{v}_{t,i}(x_i)\geq\sum_{j\in S}\lambda P_j(S)\left(f_j+\min\left\{\hat{v}_{t+1,i}(x_i-a_{ij})-\sum_{k\neq i}\pi_k^* a_{kj},\ \min_{l\neq i}\left\{\max_{0\leq y_l\leq c_l-a_{lj}}[\hat{v}_{t+1,l}(y_l)-y_l\pi_l^*]-\sum_k a_{kj}\pi_k^*+\pi_i^* x_i\right\}\right\}\right)$$

$$+(\lambda P_0(S)+1-\lambda)\min\Big\{\hat{v}_{t+1,i}(x_i),\ \min_{l\neq i}\Big\{\max_{0\le y_l\le c_l}[\hat{v}_{t+1,l}(y_l)-\pi_l^* y_l]+\pi_i^* x_i\Big\}\Big\},$$
$$\forall\, i,t,x_i,\ S\subseteq N(x_i,c_{-i}). \quad (9)$$

From the construction of the restricted constraints, all feasible solutions to ($\widehat{\text{NLP}}$) are also feasible for (NLP). The following proposition is a direct consequence of this observation.

Proposition 2. $z_{\widehat{\text{NLP}}} \ge z_{\text{NLP}}$.

In the next subsection, we introduce a simultaneous dynamic programming approach to solve ($\widehat{\text{NLP}}$). We will show that the approach leads to an efficient solution algorithm for the problem ($\widehat{\text{NLP}}$). Furthermore, the approach also leads to simple inductive proof of the relationships among different bounds.

4.2. A Simultaneous Dynamic Programming Approach

The formulation ($\widehat{\text{NLP}}$) is still a nonlinear optimization problem with a large number of nonlinear constraints. Therefore, a brute-force solution of the problem is very computationally intensive, if possible at all. We introduce, in this section, a simultaneous dynamic programming approach to solve the problem by exploiting the special structure of the constraints. We first show the following proposition.

Proposition 3. *For each feasible solution* $\{\hat{v}_{t,i}(\cdot)\}_{\forall t,i}$ *of* ($\widehat{NLP}$), *there exists a feasible solution with equal or smaller objective value such that equality in* (9) *holds for some* $S_{t,i}(x_i)\subseteq N(x_i,c_{-i})$ *for each* t, i, x_i.

Proof. Let $\{\hat{v}_{t,i}\}_{\forall t,i}$ be a feasible solution to ($\widehat{\text{NLP}}$). Suppose $\hat{v}_{t,i}(x_i)$ is strictly greater than the right-hand side of (9) for all $S\subseteq N(x_i,c_{-i})$ for fixed t, i, x_i. Then, the solution $\{\hat{v}_{t,i}\}_{\forall t,i}$ can be modified by replacing $\hat{v}_{t,i}(x_i)$ with

$$v^{\dagger}_{t,i}(x_i)=\max_{S\subseteq N(x_i,c_{-i})}\sum_{j\in S}\lambda P_j(S)\Big(f_j+\min\Big\{\hat{v}_{t+1,i}(x_i-a_{ij})-\sum_{k\neq i}\pi_k^* a_{kj},\ \min_{l\neq i}\Big\{\max_{0\le y_l\le c_l-a_{lj}}[\hat{v}_{t+1,l}(y_l)-y_l\pi_l^*]-\sum_k a_{kj}\pi_k^*+\pi_i^* x_i\Big\}\Big\}\Big)$$
$$+(\lambda P_0(S)+1-\lambda)\min\Big\{\hat{v}_{t+1,i}(x_i),\ \min_{l\neq i}\Big\{\max_{0\le y_l\le c_l}[\hat{v}_{t+1,l}(y_l)-\pi_l^* y_l]+\pi_i^* x_i\Big\}\Big\}.$$

It is easy to check that the new solution is still feasible. Repeating this procedure for all values in the solution such that strict inequality holds in (9) yields the result. □

An immediate corollary for Proposition 3 is the following.

Corollary 1. *There exists an optimal solution* $\{\hat{v}_{t,i}(\cdot)\}_{\forall t,i}$ *to* ($\widehat{NLP}$) *such that equality holds for some* $S_{t,i}(x_i)\subseteq N(x_i,c_{-i})$ *for each fixed* t, i, x_i.

Observe that, by construction, each constraint in ($\widehat{\text{NLP}}$) only involves x_i for one resource i, but not all other resources. Define a set of value functions $\{\tilde{v}_{t,i}(\cdot)\}$ for all t and i as follows:

($\widehat{\text{DP}}$)

$$\tilde{v}_{t,i}(x_i)=\max_{S\subseteq N(x_i,c_{-i})}\sum_{j\in S}\lambda P_j(S)\Big(f_j+\min\Big\{\tilde{v}_{t+1,i}(x_i-a_{ij})-\sum_{k\neq i}\pi_k^* a_{kj},\ \min_{l\neq i}\Big\{\max_{0\le y_l\le c_l-a_{lj}}[\tilde{v}_{t+1,l}(y_l)-y_l\pi_l^*]-\sum_k a_{kj}\pi_k^*+\pi_i^* x_i\Big\}\Big\}\Big)$$
$$+(\lambda P_0(S)+1-\lambda)\min\Big\{\tilde{v}_{t+1,i}(x_i),\ \min_{l\neq i}\Big\{\max_{0\le y_l\le c_l}[\tilde{v}_{t+1,l}(y_l)-\pi_l^* y_l]+\pi_i^* x_i\Big\}\Big\},$$
$$\forall\, i,t,x_i$$

with boundary conditions $\tilde{v}_{\tau+1,i}(x_i)=0$ for all i, x_i.

Next, we establish the equivalence between ($\widehat{\text{NLP}}$) and ($\widehat{\text{DP}}$). We have the following result:

Proposition 4. *There exists an optimal solution* $\{\hat{v}^*_{t,i}(\cdot)\}_{\forall t,i}$ *to* ($\widehat{NLP}$) *such that* $\tilde{v}_{t,i}(x_i)=\hat{v}^*_{t,i}(x_i)$ *for all* t, i, x, *where* $\tilde{v}_{t,i}(x_i)$ *is defined in* ($\widehat{DP}$).

Proof. From Corollary 1, it is without loss of optimality to restrict our attention in ($\widehat{\text{NLP}}$) to optimal solutions such that equalities hold in the constraint (9). Let $\{\hat{v}^*_{t,i}(x_i)\}_{\forall t,i}$ denotes such an optimal solution to ($\widehat{\text{NLP}}$). Then,

$$\hat{v}^*_{t,i}(x_i)=\max_{S\subseteq N(x_i,c_{-i})}\sum_{j\in S}\lambda P_j(S)\Big(f_j+\min\Big\{\hat{v}^*_{t+1,i}(x_i-a_{ij})-\sum_{k\neq i}\pi_k^* a_{kj},\ \min_{l\neq i}\Big\{\max_{0\le y_l\le c_l-a_{lj}}[\hat{v}^*_{t+1,l}(y_l)-y_l\pi_l^*]-\sum_k a_{kj}\pi_k^*+\pi_i^* x_i\Big\}\Big\}\Big)$$
$$+(\lambda P_0(S)+1-\lambda)\min\Big\{\hat{v}^*_{t+1,i}(x_i),\ \min_{l\neq i}\Big\{\max_{0\le y_l\le c_l}[\hat{v}^*_{t+1,l}(y_l)-\pi_l^* y_l]+\pi_i^* x_i\Big\}\Big\}$$
$$\forall\, i,t,x_i. \quad (10)$$

Comparing (10) with ($\widehat{\text{DP}}$) yields the result. □

Because of the equivalence of $\tilde{v}_{t,i}(x_i)$ in ($\widehat{\text{DP}}$) and $\hat{v}^*_{t,i}(x_i)$, we will only use $\hat{v}^*_{t,i}(x_i)$ in the remainder

of the paper. Proposition 4 shows that it suffices to solve ($\widehat{\text{DP}}$) instead of ($\widehat{\text{NLP}}$). The formulation ($\widehat{\text{DP}}$) involves m single-resource dynamic programs that need to be solved simultaneously because the optimality equation for $\hat{v}^*_{t,i}(x_i)$ involves $\hat{v}^*_{t+1,k}(x_k)$ for all x and $k = 1, \ldots, m$.

Even though ($\widehat{\text{DP}}$) is considerably simpler to solve than ($\widehat{\text{NLP}}$), it is potentially much more computationally intensive than the dynamic programming decomposition presented in §3.3. In particular, ($\widehat{\text{DP}}$) involves many inner minimization and maximization operations that could potentially expend heavy computational effort. However, this computational effort can be substantially reduced by precomputing in each period $G^*_{t,i}$ and $G^\dagger_{t,i}$ where

$$G^*_{t,i} = \max_{0 \le x_i \le c_i - a_{ij}} [\hat{v}^*_{t,i}(x_i) - x_i \pi^*_i], \quad \forall\, t, i,$$

$$G^\dagger_{t,i} = \max_{0 \le x_i \le c_i} [\hat{v}^*_{t,i}(x_i) - x_i \pi^*_i], \quad \forall\, t, i.$$

Note that

$$G^\dagger_{t,i} = \max\{G^*_{t,i}, \hat{v}^*_{t,i}(c_i) - c_i \pi^*_i\}, \quad \forall\, t, i,$$

which can be used to further reduce the computational cost. Our numerical study shows that this simplification can reduce computational time by 20 times compared with a naive implementation, and therefore is key to any practical implementation.

4.3. Comparison of Bounds

Recall that $\{v_{t,i}(\cdot)\}_{\forall t,i}$ are the value functions from the dynamic programming decomposition in §3.3. We show that the bounds from ($\widehat{\text{NLP}}$) are tighter than the decomposition bound in Proposition 1 originally shown in Zhang and Adelman (2009).

Proposition 5. *The following results hold*:
(i) $\hat{v}^*_{t,i}(x_i) \le v_{t,i}(x_i)$, $\forall\, i, x_i$;
(ii) $v_1(c) \le z_{\text{NLP}} \le z_{\widehat{NLP}} = \min_i\{\hat{v}^*_{1,i}(c_i) + \sum_{k \ne i} \pi^*_k c_k\} \le \min_i\{v_{1,i}(c_i) + \sum_{k \ne i} \pi^*_k c_k\} \le z_{\text{CDLP}}$.

Proof. (i) The proof follows by induction on time t for each state x. Fix i. Because $\hat{v}^*_{\tau+1,i}(x_i) = v_{\tau+1,i}(x_i) = 0$ for each x_i, the inequality holds trivially for time $\tau+1$. Now, suppose that the inequality holds for $t+1$, i.e.,

$$\hat{v}^*_{t+1,i}(x_i) \le v_{t+1,i}(x_i), \quad \forall\, x.$$

For each state x_i, we have

$$\begin{aligned}
&\hat{v}^*_{t,i}(x_i)\\
&\le \max_{S \subseteq N(x_i, c_{-i})} \sum_{j \in S} \lambda P_j(S) \Bigg(f_j + \min\Bigg\{ v_{t+1,i}(x_i - a_{ij}) - \sum_{k \ne i} \pi^*_k a_{kj},\\
&\qquad \min_{l \ne i} \Bigg\{ \max_{0 \le y_l \le c_l - a_{lj}} [v_{t+1,l}(y_l) - y_l \pi^*_l] - \sum_k a_{kj} \pi^*_k + \pi^*_i x_i \Bigg\} \Bigg\} \Bigg)\\
&\quad + (\lambda P_0(S) + 1 - \lambda) \min\Bigg\{ v_{t+1,i}(x_i),\\
&\qquad \min_{l \ne i} \Bigg\{ \max_{0 \le y_l \le c_l} [v_{t+1,l}(y_l) - \pi^*_l y_l] + \pi^*_i x_i \Bigg\} \Bigg\}\\
&\le \max_{S \subseteq N(x_i, c_{-i})} \sum_{j \in S} \lambda P_j(S) \Bigg(f_j + v_{t+1,i}(x_i - a_{ij}) - \sum_{k \ne i} \pi^*_k a_{kj} \Bigg)\\
&\quad + (\lambda P_0(S) + 1 - \lambda) v_{t+1,i}(x_i)\\
&= \max_{S \subseteq N(x_i, c_{-i})} \Bigg\{ \sum_{j \in S} \lambda P_j(S) \Bigg[f_j - \sum_{k \ne i} a_{kj} \pi^*_k - (v_{t+1,i}(x_i)\\
&\qquad - v_{t+1,i}(x_i - a_{ij})) \Bigg] \Bigg\} + v_{t+1,i}(x_i)\\
&= v_{t,i}(x_i).
\end{aligned}$$

In the above, the first inequality follows from the formulation ($\widehat{\text{DP}}$) and the inductive assumption; the second inequality follows by taking the first term in each minimization on the right-hand side, and the last equality follows from (DP_i).

(ii) The first inequality follows because an optimal solution to (NLP) is a feasible solution to (LP); the second inequality is from Proposition 2; the equality is from Proposition 4, and the third inequality follows from Part (i). The last inequality is from Proposition 1. □

Part (i) in Proposition 5 implies that the separable approximation $\sum_i \hat{v}^*_{t,i}(x_i)$ is tighter than the separable approximation $\sum_i v_{t,i}(x_i)$. Even though we are unable to establish analytically such separable approximations as either upper bounds or lower bounds on $v_t(x)$, we do observe in all our numerical examples that $\sum_i v_{1,i}(c_i) \ge \sum_i \hat{v}^*_{1,i}(c_i) \ge \min_i\{\hat{v}^*_{1,i}(c_i) + \sum_{k \ne i} \pi^*_k c_k\}$. Therefore, for practical purposes, $\sum_i \hat{v}^*_{1,i}(c_i)$ can be viewed as a tighter approximation than $\sum_i v_{1,i}(c_i)$ for the total expected revenue $v_1(c)$.

Before we move on, we would like to make some comments on the use of the nonlinear nonseparable approximation (6). Note that all our results carry over for time-dependent bid-price vectors. Although we propose the use of π^* from (CDLP) in (6), we can also use dynamic bid-prices $V^*_{t,i}$ from Zhang and Adelman (2009). The main disadvantage is that the computation of $V^*_{t,i}$, as shown in Zhang and Adelman (2009), can be quite computationally intensive.

It is also possible to consider more general functional approximations. Meissner and Strauss (2009) consider inventory-sensitive bid-prices where the inventory level x_i for each resource can be divided into several ranges and a different bid-price value is used for each inventory range. In the most general case, each unit of inventory is associated with a different bid-price value, which is equivalent to the separable functional approximation $v_t(x) = \sum_{i=1}^m w_{t,i}(x_i)$, $\forall\, t, i, x$, for a set of value functions $\{w_{t,i}(\cdot)\}_{\forall t,i}$. This

approximation is very appealing because it directly corresponds to the separable approximation used to generate control policies. The main disadvantage of the approach is its computational cost, which is at least as expensive as the linear functional approximation approach in Zhang and Adelman (2009). Meissner and Strauss (2009) consider inventory aggregation where inventory levels are divided into ranges and one value is used for each range. They further point out that it is possible to combine inventory aggregation and time aggregation to achieve additional reduction in computational cost, which is a potentially interesting topic for future research.

The main advantage of our approach in this paper lies in its relatively low computational cost. In fact, its computational time requirement is only slightly higher than that of the decomposition approach in §3.3, as will be shown in our numerical study.

5. Heuristic Policies

In this section, we introduce heuristic policies from ($\widehat{\text{NLP}}$) (or equivalently ($\widehat{\text{DP}}$)). For comparison purposes, we also consider policies from (CDLP) and the classical dynamic programming decomposition in §3.3. For our computational study, we consider the MNLD choice model (Liu and van Ryzin 2008).

5.1. MNLD Choice Model

In MNLD, each customer is interested in a subset of the products. Let $L = \{1, \ldots, \bar{l}\}$ be the set of customer segments. We assume that customer segment $l \in L$ has consideration set $N_l \subseteq N$. We assume $\forall\, l_1 \neq l_2$, $N_{l_1} \cap N_{l_2} = \varnothing$; i.e., different customer segments have disjoint consideration sets.

Within each segment, customer choice follows a multinomial logit (MNL) model. Under MNL, the choice probability can be defined by a vector of weights, which are also referred to as preference values. To completely specify the choice probabilities, we need to specify the preference value v_{lj} for $l \in L$, $j \in N_l$, and the no-purchase value v_{l0} for $l \in L$. In general, a choice set S can also be represented by an availability vector. Given a choice set S, we use a binary vector $u_l(S)$ to denote the product availability for segment l such that $u_{lk}(S) = 1\{k \in S\}$ for all $k \in N_l$. For convenience, we use vector and set notations interchangeably. Then, the probability a segment l customer purchases product $j \in N_l$ is $\tilde{P}_{lj}(S) = u_{lj}(S)v_{lj}/[\sum_{k\in N_l} u_{lk}(S)v_{lk} + v_{l0}]$.

To accommodate the MNLD choice model in the framework of §3, we assume that an arriving customer first chooses his relevant segment, and then chooses products within the given segment. In particular, we assume the probability that an arriving customer belongs to segment l is λ_l/λ, where $\sum_l \lambda_l = \lambda$. It then follows that, for $j \in N_l$, $P_j(S) = \lambda_l \tilde{P}_{lj}(S)/\lambda$.

Liu and van Ryzin (2008) show that, for each fixed time t and state x, the maximization in (DP) is efficiently solvable. Furthermore, (CDLP) can be efficiently solved using a column generation procedure. Their work builds on the earlier work of Talluri and van Ryzin (2004a) and Gallego et al. (2004). Because the maximization for each fixed t and state x in ($\widehat{\text{DP}}$) has the same structure as in (DP), the maximization in ($\widehat{\text{DP}}$) is also efficiently solvable for the MNLD choice model.

5.2. DCOMP1

In this subsection, we introduce a heuristic policy from the solution of ($\widehat{\text{DP}}$). Recall that the solution is denoted by $\{\hat{v}^*_{t,i}(\cdot)\}_{\forall i,t}$. Consider the separable approximation $v_t(x) \approx \sum_i \hat{v}^*_{t,i}(x_i)$, $\forall\, t, x$. Then, the opportunity cost of selling product j in period t for $x \geq A_j$ is given by

$$\begin{aligned}\Delta_j v_{t+1}(x) &= v_{t+1}(x) - v_{t+1}(x - A_j)\\ &\approx \sum_i \hat{v}^*_{t+1,i}(x_i) - \sum_i \hat{v}^*_{t+1,i}(x_i - a_{ij})\\ &\equiv \tilde{\Delta}_j v_{t+1}(x).\end{aligned}$$

By replacing $\Delta_j v_{t+1}(x)$ in (DP) with $\tilde{\Delta}_j v_{t+1}(x)$ for all j, an offer set in period t can be computed by taking

$$\hat{S}^*_t(x) \in \arg\max_{S\subseteq N(x)} \sum_{j\in S} \lambda P_j(S)[f_j - \tilde{\Delta}_j v_{t+1}(x)].$$

A heuristic policy that offers the product set $\hat{S}^*_t(x)$ in period t and state x is called DCOMP1 in our numerical study. Note again that the maximization above can be solved efficiently for the MNLD choice model by a ranking procedure (see Liu and van Ryzin 2008, Proposition 6).

5.3. CDLP

Let π^* be the vector of dual values corresponding to the resource constraints in (CDLP). In our numerical study We call the policy that approximates $\Delta_j v_t(x)$ with $\sum_i \pi^*_i a_{ij}$ for each j for all t and $x \geq A_j$ CDLP. Note that product j will not be offered when $x_i < a_{ij}$ for some i. Unlike bid-price policies for independent demand models (Talluri and van Ryzin 1998), the offer set from CDLP in general depends on the vector π^*, and it is possible that a product j with $p_j > \sum_i a_{ij}\pi^*_i$ may not be offered. CDLP has the lowest computational cost in the policies we test in this paper because, unlike the decomposition-based approaches, it does not involve the computation of dynamic programming value functions. One way to improve the performance of policies from static approximations such as CDLP is to resolve the static model, taking into account changes in the capacity and remaining

time. In our numerical study, we also consider versions of CDLP where the bid-prices are updated multiple times by resolving throughout the time horizon. We note that, in general, resolving does not necessarily improve the performance of the policy (Cooper 2002, Secomandi 2008).

5.4. DCOMP

Alternatively, the dual values π^* can be used in the dynamic programming decomposition approach outlined in §3.3. The resulting solution $\{v_{t,i}(\cdot)\}_{\forall t,i}$ can be used in place of $\{\hat{v}^*_{t,i}(\cdot)\}_{\forall t,i}$ in an approximation approach as in §5.2. This approach is the decomposition heuristic proposed in Liu and van Ryzin (2008) and is called DCOMP in our numerical study.

6. Numerical Study

In this section, we report results from a numerical study that investigates the revenue and computational performance of the policy DCOMP1 introduced in §5.2. The performance of DCOMP1 is compared to three other policies:

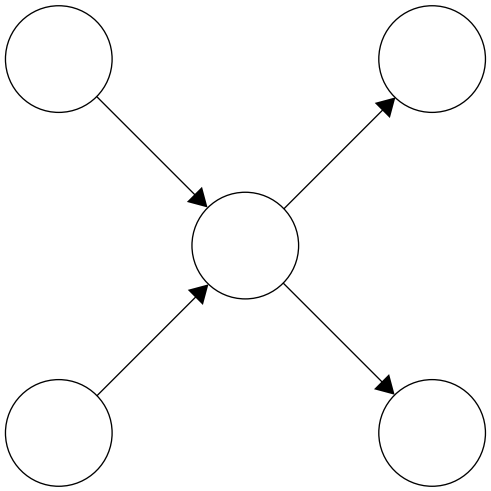
- DCOMP: The classical dynamic programming decomposition (§§3.3 and 5.4);
- CDLP: The policy based on the dual values of resource constraints in the choice-based deterministic linear program (§5.3);
- CDLP10: A version of CDLP that resolves 10 times with equally spaced resolving intervals.

6.1. Computational and Bound Performance

6.1.1. Description of Test Instances. To study the computational and bound performance, we randomly generate 16 hub-and-spoke network instances. The instances are labeled A1 to A16. We vary the number of periods τ in the set {100, 200, 400, 800}. We consider networks with 4, 8, 16, and 24 nonhub locations. See Figure 1 for a hub-and-spoke network with 4 nonhub locations. Half of the nonhub locations each have one flight to the hub and the other half each have one flight from the hub. Local itineraries are trips to or from the hub. Customers can also travel between nonhub locations with flights to the hub and nonhub locations with flights from the hub. Itineraries between nonhub locations are called through itineraries. For each itinerary (local or through itinerary), there are two fare products with different fares. The largest problem instance we consider has 24 nonhub locations and 336 fare products, representing a fairly large network.

Next, we briefly describe the method used to generate fares and choice parameters. The fares for local

Figure 1 Hub-and-Spoke Network with Four Locations



itineraries to the hub are generated by adding 100 to a truncated normal random variable with mean 800 and standard deviation 400, whereas the fares for local itineraries from the hub are generated by adding 50 to a truncated normal random variable with mean 400 and standard deviation 200. Each through itinerary fare is 0.95 times the sum of the fares of the two local itineraries it covers. The choice weight for each product is generated from uniform $[1, 6]$. The choice weights for no-purchase is chosen such that no-purchase probability is 1/3 when all products are available for purchase. The parameter $\lambda_l = 0.5U_l / \sum_{k=1}^{L} U_k$, where U_ks are independent uniform $[0, 1]$ random variables.

Table 1 shows more details of the test instances A1 to A16. The parameters column shows the number of periods, the number of nonhub locations, the number of flights, and the number of fare products. For example, for the test instance A16, there are 800 periods, 24 nonhub locations, 24 flights, and 336 fare products. Capacity per leg shows the number of seats available for each flight. The load factor column shows the load factor as defined by

$$\rho = \frac{\lambda \sum_{t=1}^{\tau} \sum_{j \in S^*} \sum_{i=1}^{m} a_{ij} P_j(S^*)}{\sum_{i=1}^{m} c_i} \tag{11}$$

with $S^* \in \arg\max_{S \subseteq N} \sum_{j \in S} P_j(S) f_j$ being the revenue-maximizing set of open products when there is ample capacity of each resource. The definition of load factor in (11) follows Zhang and Adelman (2009). Note that in the no-choice setting S^* is the set of all products and the load factor defined in (11) is the same as the one commonly used in the literature.

6.1.2. Computational Time. Table 1 shows the CPU seconds for problem instances A1 to A20. The code is implemented in MATLAB version R2009a and runs on a desktop with Intel Core Quad CPU Q6600

Table 1 CPU Seconds for CDLP, DCOMP, and DCOMP1 in Hub-and-Spoke Test Cases

				CPU seconds			
Case #	Parameters	Capacity per leg	Load factor	CDLP	DCOMP	DCOMP1	(DCOMP1 – DCOMP)/DCOMP (%)
A1	(100, 4, 4, 16)	10	1.17	0.16	2.03	2.75	35.38
A2	(200, 4, 4, 16)	20	1.27	0.23	7.89	10.88	37.82
A3	(400, 4, 4, 16)	40	1.19	0.16	31.92	43.73	37.00
A4	(800, 4, 4, 16)	80	1.28	0.20	127.66	174.48	36.68
A5	(100, 8, 8, 48)	5	1.43	1.52	5.75	7.47	29.89
A6	(200, 8, 8, 48)	10	1.36	0.72	22.83	29.58	29.57
A7	(400, 8, 8, 48)	20	1.35	1.61	91.67	118.92	29.73
A8	(800, 8, 8, 48)	40	1.21	0.72	362.84	471.73	30.01
A9	(100, 16, 16, 160)	2	1.65	4.64	15.09	19.42	28.67
A10	(200, 16, 16, 160)	5	1.45	4.69	75.84	96.97	27.85
A11	(400, 16, 16, 160)	10	1.29	2.92	303.66	388.97	28.10
A12	(800, 16, 16, 160)	20	1.40	3.64	1,218.67	1,560.19	28.02
A13	(100, 24, 24, 336)	1	1.45	3.69	24.72	31.81	28.70
A14	(200, 24, 24, 336)	2	1.35	4.59	98.52	127.36	29.28
A15	(400, 24, 24, 336)	5	1.29	4.39	492.73	630.84	28.03
A16	(800, 24, 24, 336)	10	1.38	4.23	1,978.14	2,532.20	28.01

Note. The Parameters column shows the number of periods, the number of nonhub locations, the number of flights, and the number of fare products.

2.4 GHz and 8 GB memory. The operating system is Windows XP Professional 64 bit. We report the computational time for CDLP, DCOMP, and DCOMP1. Note that for this set of problems, CDLP can be solved to optimality very quickly, in less than five seconds for all instances. From a computational perspective, this is very appealing compared with the other two methods, DCOMP and DCOMP1.

We are primarily interested in the comparison between DCOMP and DCOMP1. The last column in Table 1 reports the time increase of DCOMP1 compared with DCOMP. On average, DCOMP1 takes 30% more time to solved than DCOMP. The added computational cost is mainly from finding the values of G_t^* and $G_t^\dagger$. Although DCOMP1 is more computationally intensive, it is still feasible for reasonably large problems. Among the instances we considered, the largest instance A16 with 24 locations and 336 products takes about 43 minutes to solve using DCOMP1. Note that we do not claim the most efficient implementation, but are mainly interested in the comparison of the two methods. More efficient implementation of the methods are possible with a lower-level programming language and through parallel computing. Note that because the subproblems in DCOMP are independent from each other, it can be easily implemented through parallel computing, whereas the implementation of DCOMP1 can be more involved because the subproblems need to exchange information in the solution process.

6.1.3. Bounds. Table 2 reports the bounds for problem instances A1 to A16. We report three bounds: the CDLP objective value, the decomposition bound

Table 2 Comparison of Bounds in Hub-and-Spoke Test Cases

				Bound improvement (%)		%-difference across legs (%)	
Case #	CDLP bound	DCOMP bound	DCOMP1 bound	%-CDLP	%-DCOMP	DCOMP	DCOMP1
A1	24,078.90	23,985.56	22,900.49	5.15	4.74	4.46	0.00
A2	48,367.58	48,328.43	47,588.56	1.64	1.55	1.87	0.36
A3	89,312.44	87,576.49	86,729.90	2.98	0.98	2.36	0.00
A4	213,102.50	211,854.85	211,087.37	0.95	0.36	0.58	0.00
A5	32,521.30	31,029.90	30,726.17	5.84	0.99	3.18	0.05
A6	70,541.63	68,760.67	68,617.41	2.80	0.21	2.18	0.22
A7	107,831.01	106,339.36	106,153.32	1.58	0.18	1.09	0.00
A8	216,080.83	212,915.61	212,848.05	1.52	0.03	1.49	0.00
A9	26,347.76	24,953.24	24,764.75	6.39	0.76	4.69	0.00
A10	60,629.35	58,489.12	58,118.33	4.32	0.64	2.95	0.03
A11	101,616.47	100,069.27	99,771.63	1.85	0.30	1.52	0.01
A12	224,780.69	222,558.53	222,231.72	1.15	0.15	0.94	0.00
A13	13,074.04	11,845.73	10,386.38	25.88	14.05	10.37	0.00
A14	26,296.19	24,926.41	24,373.33	7.89	2.27	5.50	0.00
A15	74,112.13	72,089.14	71,617.55	3.48	0.66	2.80	0.03
A16	131,457.79	129,589.28	129,273.91	1.69	0.24	1.44	0.00

from DCOMP, and the decomposition bound from DCOMP1. The bound improvement column shows the percentage difference between DCOMP1 bound and the other two bounds. First note that both decomposition bounds are better than the CDLP bound. In particular, the DCOMP1 bound can be 25% smaller than the CDLP bound for problem instance A13 with capacity per leg of 1. Furthermore, the DCOMP1 bound is tighter than the DCOMP bound, confirming our theoretical results. Although DCOMP bound and DCOMP1 bound are fairly close in most problem instances, the DCOMP1 bound can be 14% smaller than the DCOMP bound for problem instance A13 with capacity per leg of 1. On average, DCOMP1 bounds are 4.69% and 1.76% tighter than the CDLP and DCOMP bounds in this set of problem instances. This suggests that the DCOMP1 bound is a better benchmark in computational studies. Note that the load factors are similar for this set of problem instances. Therefore, load factor alone cannot explain the difference among the different bounds.

Because both the DCOMP bound and DCOMP1 bounds are taken to be minimum across all legs for each problem, the value $\hat{v}^*_{1,i}(c_i) + \sum_{k\neq i} \pi^*_k c_k$ is also an upper bound for each leg i. In general, the values from different legs are different. To measure the variability of the values across legs, we report the maximum percentage difference across legs for DCOMP and DCOMP1 in the last column of Table 2. For example, the maximum percentage difference across legs for DCOMP1 is given by

$$\frac{\max_i\{\hat{v}^*_{1,i}(c_i) + \sum_{k\neq i} \pi^*_k c_k\} - \min_i\{\hat{v}^*_{1,i}(c_i) + \sum_{k\neq i} \pi^*_k c_k\}}{\min_i\{\hat{v}^*_{1,i}(c_i) + \sum_{k\neq i} \pi^*_k c_k\}} \times 100\%.$$

Observe that the values from different legs can be substantially different for DCOMP, with the average relative difference of 2.96%. On the other hand, DCOMP1 values from different legs are much closer, with an average relative difference of 0.04% across legs. In this sense, DCOMP1 values are more homogeneous, which is a nice property to have.

6.2. Policy Performance

The performance of different policies are evaluated through simulation. In our computer implementation, arrival and choice processes in each period are constructed from a uniform $[0, 1]$ random variable U. Given an offer set $S \subseteq N$, a unit of product j is sold in the given period if $\lambda \sum_{k=1}^{j-1} P_k(S) \leq U < \lambda \sum_{k=1}^{j} P_k(S)$. There are no sales in a period if $U \geq \lambda \sum_{k=1}^{n} P_k(S)$. We randomly generated 20,000 demand streams where each demand stream contains a random variable for each period. These 20,000 demand streams are used in the simulation for all problem instances we consider.

6.2.1. Hub-and-Spoke Network with Two Nonhub Locations. We first consider a hub-and-spoke network with two nonhub locations. There are two fare products for each local itinerary and two fare products for the through itinerary. So, in total there are six products. The definition of segments and products are described in Tables 3 and 4. We then vary the number of periods τ in the set {100, 200, 400, 800}. We also vary the capacity per leg to generate cases with different load factors. In all, we consider 20 cases with different time horizons and load factors, which are labeled B1 to B20. Table 5 reports the results for the 20 cases considered in our simulation. The revenues reported are averages over 20,000 simulations. With very few exceptions, the relative error (measured by half width of 95% confidence interval divided by the mean) is less than 0.3%, with a maximum relative error of 0.8%. This shows that the simulation is accurate enough for our purpose because the magnitude of the error is far less than the relative performance gap reported for the different policies we consider in the study. The DCOMP1 bound column reports the revenue bounds from Proposition 5, which is used as the benchmark in our study. The OPT-GAP column in the table reports the optimality gap of the policy DCOMP1, which is calculated as (DCOMP1 REV − DCOMP1 Bound)/(DCOMP1 Bound) × 100%. The results show that DCOMP1 performs quite well with an optimality gap of less than 5% in the majority of problem instances. There are a few cases where the optimality gap is more than 5%. However, we note that the gap is measured against a theoretical upper bound. The last three columns measure the revenue gains of DCOMP1 against CDLP, CDLP10, and DCOMP.

Table 3 **Segments in Hub-and-Spoke Network with Two Nonhub Locations**

l	λ_l	N_l	v_{l0}
1	0.12	{1, 2}	1.62
2	0.04	{3, 4}	2.68
3	0.09	{5, 6}	4.00

Table 4 **Products in Hub-and-Spoke Network with Two Nonhub Locations**

j	f_j	v_j	A^{T}_j
1	621.91	1.83	[1 0]
2	133.00	1.41	[1 0]
3	936.10	3.64	[0 1]
4	525.89	1.73	[0 1]
5	1,480.11	5.62	[1 1]
6	625.94	2.38	[1 1]

Table 5 Simulation Results for Hub-and-Spoke Network with Two Nonhub Locations

										DCOMP1 revenue gains (%)		
Case #	τ	Load factor	Capacity per leg	CDLP REV	CDLP10 REV	DCOMP REV	DCOMP1 REV	DCOMP1 bound	OPT-GAP (%)	%CDLP	%CDLP10	%DCOMP
B1	100	2.40	4	2,041.49	5,320.27	5,621.55	5,775.47	5,964.48	−3.17	182.90	8.56	2.74
B2	200	2.13	9	4,281.30	12,542.84	12,739.75	13,262.92	13,538.51	−2.04	209.79	5.74	4.11
B3	400	2.13	18	25,714.80	25,540.14	25,448.01	25,456.41	27,236.23	−6.53	−1.00	−0.33	0.03
B4	800	2.13	36	17,242.49	51,666.33	50,338.88	53,946.59	54,599.61	−1.20	212.87	4.41	7.17
B5	100	1.60	6	5,396.36	7,800.48	8,027.36	8,034.27	8,661.98	−7.25	48.88	3.00	0.09
B6	200	1.60	12	15,269.14	16,713.44	16,372.88	17,318.40	17,727.87	−2.31	13.42	3.62	5.77
B7	400	1.60	24	8,610.58	33,854.61	32,852.85	35,472.06	35,919.34	−1.25	311.96	4.78	7.97
B8	800	1.60	48	46,157.80	68,704.48	65,619.10	65,618.95	72,267.91	−9.20	42.16	−4.49	0.00
B9	100	1.37	7	8,183.45	9,147.58	9,093.38	9,269.56	9,845.49	−5.85	13.27	1.33	1.94
B10	200	1.28	15	17,836.09	20,320.91	19,608.36	20,521.01	21,308.99	−3.70	15.05	0.98	4.65
B11	400	1.28	30	36,785.29	41,760.28	39,273.65	42,471.91	43,382.17	−2.10	15.46	1.70	8.14
B12	800	1.28	60	75,209.11	84,766.35	82,297.72	86,841.58	87,848.39	−1.15	15.47	2.45	5.52
B13	100	1.07	9	10,795.78	11,055.57	11,069.36	11,107.51	11,297.66	−1.68	2.89	0.47	0.34
B14	200	1.07	18	22,195.55	23,007.13	23,242.32	23,268.75	23,456.05	−0.80	4.84	1.14	0.11
B15	400	1.07	36	39,166.15	46,785.95	47,807.99	47,824.97	47,956.30	−0.27	22.11	2.22	0.04
B16	800	1.07	72	89,878.19	94,964.14	96,987.90	96,993.79	97,067.36	−0.08	7.92	2.14	0.01
B17	100	0.96	10	9,573.53	11,683.40	11,832.66	11,854.02	11,955.46	−0.85	23.82	1.46	0.18
B18	200	0.91	21	24,618.37	24,950.34	25,248.58	25,259.70	25,278.33	−0.07	2.61	1.24	0.04
B19	400	0.91	42	39,638.90	50,346.34	51,589.78	51,593.10	51,575.10	0.03	30.16	2.48	0.01
B20	800	0.91	84	79,396.81	101,731.96	104,375.64	104,376.37	104,361.48	0.01	31.46	2.60	0.00

In our numerical study, we also tried more frequent resolving of CDLP, but did not find significant revenue improvement. The results show that DCOMP1 outperforms CDLP10 and DCOMP in the majority of test cases; it generates revenue gains as high as 8% compared with both CDLP10 and DCOMP. The performance of DCOMP1 is remarkable considering the fact that DCOMP is often viewed as one of the strongest heuristics available for network RM. A comparison with CDLP10 shows that DCOMP1, even without resolving, beats a frequently resolved version of CDLP policies from the choice-based deterministic linear program. Because resolving is not always desirable in practice, this property makes DCOMP1 particularly appealing. The revenue gains of DCOMP1 tend to be larger in cases where the load factors are medium or high. This is, to some extent, not surprising because when the load factors are low, using revenue-maximizing sets would tend to work well and the value of more accurate dynamic control diminishes. It is also noteworthy that the performance lifts against DCOMP are achieved in several cases with relatively long time horizon. This suggests the potential of DCOMP1 in relatively large realistic-sized problems. Table 6 reports the empirical load factors, which are calculated as (average capacity units sold/total network capacity) over the 20,000 simulation runs. It is interesting to note that DCOMP1 has higher empirical load factors than DCOMP and CDLP10 in almost all test cases.

Table 7 shows the performance of various bounds for the test cases B1–B20. DCOMP1 bound can be substantially tighter than CDLP bound, with a maximum difference of 8.57%. The difference between the DCOMP1 bound and DCOMP bound is relatively small, with a maximum difference of 0.91%. Therefore, even when the bounds are close, there can be substantial performance difference in policies between DCOMP and DCOMP1. We conjecture that this is due to the fact that DCOMP1 generates

Table 6 Empirical Load Factors for Simulated Test Instances

	B				C			
Case	CDLP	CDLP10	DCOMP	DCOMP1	CDLP	CDLP10	DCOMP	DCOMP1
1	0.27	0.88	0.94	0.96	0.70	0.82	0.82	0.82
2	0.25	0.92	0.94	0.97	0.50	0.84	0.84	0.85
3	0.95	0.94	0.94	0.94	0.75	0.86	0.85	0.85
4	0.26	0.95	0.93	0.99	0.52	0.87	0.87	0.87
5	0.63	0.87	0.90	0.90	0.51	0.79	0.79	0.80
6	0.82	0.93	0.92	0.96	0.72	0.81	0.80	0.82
7	0.19	0.94	0.92	0.98	0.61	0.83	0.81	0.81
8	0.68	0.95	0.92	0.92	0.83	0.84	0.84	0.84
9	0.75	0.88	0.88	0.89	0.73	0.77	0.76	0.78
10	0.77	0.91	0.89	0.92	0.76	0.79	0.77	0.80
11	0.79	0.93	0.89	0.95	0.78	0.81	0.78	0.82
12	0.81	0.94	0.92	0.97	0.48	0.82	0.80	0.82
13	0.82	0.83	0.83	0.83	0.73	0.74	0.72	0.73
14	0.84	0.86	0.87	0.87	0.76	0.77	0.76	0.76
15	0.70	0.87	0.90	0.90	0.75	0.79	0.78	0.78
16	0.85	0.89	0.91	0.91	0.79	0.80	0.80	0.80
17	0.62	0.79	0.80	0.80	0.68	0.71	0.69	0.70
18	0.80	0.81	0.82	0.82	0.71	0.73	0.72	0.72
19	0.61	0.81	0.84	0.84	0.72	0.74	0.74	0.74
20	0.61	0.82	0.85	0.85	0.73	0.75	0.75	0.75

Table 7 Bounds for Hub-and-Spoke Network with Two Nonhub Locations

				Bound improvement (%)		% Difference across legs (%)	
Case #	CDLP bound	DCOMP bound	DCOMP1 bound	%-CDLP	%-DCOMP	DCOMP	DCOMP1
B1	6,099.91	5,964.48	5,964.48	2.27	0.00	1.35	0.55
B2	13,679.92	13,538.51	13,538.51	1.04	0.00	0.64	0.19
B3	27,359.84	27,236.23	27,236.23	0.45	0.00	0.26	0.02
B4	54,719.69	54,599.61	54,599.61	0.22	0.00	0.12	0.00
B5	9,060.13	8,661.98	8,661.98	4.60	0.00	3.81	3.38
B6	18,120.25	17,727.87	17,727.87	2.21	0.00	1.90	1.56
B7	36,240.50	35,919.34	35,919.34	0.89	0.00	0.74	0.56
B8	72,481.01	72,267.91	72,267.91	0.29	0.00	0.22	0.13
B9	10,540.24	9,845.49	9,845.49	7.06	0.00	6.11	5.81
B10	22,560.58	21,308.99	21,308.99	5.87	0.00	5.51	5.28
B11	45,121.16	43,382.17	43,382.17	4.01	0.00	3.88	3.73
B12	90,242.33	87,848.39	87,848.39	2.73	0.00	2.66	2.59
B13	12,266.02	11,400.82	11,297.66	8.57	0.91	3.69	0.12
B14	24,532.03	23,594.66	23,456.05	4.59	0.59	2.11	0.03
B15	49,064.06	48,104.32	47,956.30	2.31	0.31	1.26	0.00
B16	98,128.12	97,172.28	97,067.36	1.09	0.11	0.78	0.00
B17	12,887.93	11,971.65	11,955.46	7.80	0.14	5.59	0.44
B18	26,397.76	25,278.35	25,278.33	4.43	0.00	4.03	0.75
B19	52,795.52	51,575.10	51,575.10	2.37	0.00	2.31	0.52
B20	105,591.04	104,361.48	104, 361.48	1.18	0.00	1.18	0.27

decomposition values that are more homogeneous across legs, which leads to more consistent marginal value estimates used in policy generation.

6.2.2. The Effect of Reoptimization. To investigate the robustness of the policy DCOMP1, we briefly discuss cases where DCOMP and DCOMP1 are resolved a few times throughout the selling horizon. For this purpose, we consider all problem instances B1–B20. We conduct 5,000 simulations where the policies DCOMP and DCOMP1 are resolved five times over equally spaced time intervals. Note that simulations with resolving take considerably longer to finish because, in each simulation run, we need to resolve each policy five times, which is fairly time consuming. Table 8 reports the results with resolving, together with no resolving. With very few exceptions,

Table 8 The Effect of Resolving for Hub-and-Spoke Network with Two Nonhub Locations

		DCOMP (R)			DCOMP1 (R)		
Case #	DCOMP REV	REV	%-DCOMP (%)	DCOMP1 REV	REV	%-DCOMP1 (%)	%-DCOMP (R) (%)
B1	5,621.55	5,568.54	−0.94	5,775.47	5,760.51	−0.26	3.45
B2	12,739.75	12,664.63	−0.59	13,262.92	13,183.14	−0.60	4.09
B3	25,448.01	25,469.71	0.09	25,456.41	26,115.88	2.59	2.54
B4	50,338.88	51,385.01	2.08	53,946.59	53,812.38	−0.25	4.72
B5	8,027.36	8,079.66	0.65	8,034.27	8,180.23	1.82	1.24
B6	16,372.88	16,634.09	1.60	17,318.40	17,129.15	−1.09	2.98
B7	32,852.85	33,680.46	2.52	35,472.06	35,297.77	−0.49	4.80
B8	65,619.10	68,453.46	4.32	65,618.95	70,739.69	7.80	3.34
B9	9,093.38	9,226.55	1.46	9,269.56	9,356.85	0.94	1.41
B10	19,608.36	20,346.09	3.76	20,521.01	20,576.99	0.27	1.13
B11	39,273.65	41,713.03	6.21	42,471.91	42,542.27	0.17	1.99
B12	82,297.72	84,989.90	3.27	86,841.58	86,913.42	0.08	2.26
B13	11,069.36	11,148.23	0.71	11,107.51	11,188.71	0.73	0.36
B14	23,242.32	23,274.36	0.14	23,268.75	23,326.46	0.25	0.22
B15	47,807.99	47,772.67	−0.07	47,824.97	47,828.07	0.01	0.12
B16	96,987.90	97,030.05	0.04	96,993.79	97,070.21	0.08	0.04
B17	11,832.66	11,876.92	0.37	11,854.02	11,895.88	0.35	0.16
B18	25,248.58	25,250.30	0.01	25,259.70	25,262.40	0.01	0.05
B19	51,589.78	51,552.76	−0.07	51,593.10	51,557.64	−0.07	0.01
B20	104,375.64	104,455.64	0.08	104,376.37	104,457.30	0.08	0.00

Table 9 **Segments in Hub-and-Spoke Network with Four Nonhub Locations**

l	λ_l	N_l	v_{l0}
1	0.10	{1, 2}	2.88
2	0.05	{3, 4}	4.66
3	0.02	{5, 6}	3.91
4	0.10	{7, 8}	3.77
5	0.03	{9, 10}	2.54
6	0.05	{11, 12}	3.12
7	0.01	{13, 14}	3.17
8	0.14	{15, 16}	1.99

Table 10 **Products in Hub-and-Spoke Network with Four Nonhub Locations**

j	f_j	v_j	A_j^T
1	1,083.22	2.54	[1 0 0 0]
2	1,171.19	3.22	[1 0 0 0]
3	1,119.20	5.59	[0 1 0 0]
4	885.98	3.73	[0 1 0 0]
5	112.61	3.55	[0 0 1 0]
6	449.49	4.27	[0 0 1 0]
7	225.33	1.71	[0 0 0 1]
8	705.00	5.83	[0 0 0 1]
9	1,136.04	1.84	[1 0 1 0]
10	1,539.65	3.23	[1 0 1 0]
11	1,243.12	1.13	[1 0 0 1]
12	1,782.38	5.11	[1 0 0 1]
13	1,170.22	2.97	[0 1 1 0]
14	1,268.69	3.36	[0 1 1 0]
15	1,277.30	1.95	[0 1 0 1]
16	1,511.43	2.03	[0 1 0 1]

the relative error (measured by half width of 95% confidence interval divided by the mean) is less than 0.3%, with a maximum relative error of 0.6%. The results show that the performance of DCOMP1 is rather robust. In fact, resolving does not improve revenue in all problem instances. In cases where resolving helped, the revenue gain is minimal. In addition, DCOMP1 with resolving beats DCOMP with resolving.

6.2.3. Hub-and-Spoke Network with Four Nonhub Locations. We next consider a larger hub-and-spoke network with four nonhub locations. The instance is generated similar to instance A1–A16. There are four local itineraries and four through itineraries. There are two fare products for each itinerary. So, in total there are 16 fare products. The definition of segments and products are described in Tables 9 and 10. We then vary the number of periods τ in the set {100, 200, 400, 800}. We also vary the capacity per leg to generate cases with different load factors. In all, we consider 20 cases with different time horizons and load factors, which are labeled C1 to C20. Table 11 reports the results for the 20 cases considered in our simulation. The revenues reported are averages over 20,000 simulations. For all statistics reported, the relative error (measured by half width of 95% confidence interval divided by the mean) is less than 0.3%.

The results show that DCOMP1 outperforms CDLP10 and DCOMP in the majority of test cases. The performance gains compared with both CDLP10 and DCOMP are smaller than for case B1–B20, but are still quite significant. Averaged over the 20 cases, it generates revenue gains of 0.63% compared with CDLP10 and 1.07% compared with DCOMP. We note that in cases with low load factors, the performance gaps are quite small, which is consistent with our

Table 11 **Simulation Results for Hub-and-Spoke Network with Four Nonhub Locations**

										DCOMP1 revenue gains (%)		
Case #	τ	Load factor	Capacity per leg	CDLP REV	CDLP10 REV	DCOMP REV	DCOMP1 REV	DCOMP1 bound	OPT-GAP (%)	%CDLP	%CDLP10	%DCOMP
C1	100	1.99	6	14,655.87	16,719.25	16,778.72	16,795.66	17,693.73	−5.08	14.60	0.46	0.10
C2	200	1.99	12	22,926.70	34,359.44	34,570.95	35,028.96	35,974.27	−2.63	52.79	1.95	1.32
C3	400	1.99	24	62,453.81	70,242.02	70,141.74	70,163.84	72,596.09	−3.35	12.35	−0.11	0.03
C4	800	1.99	48	94,494.93	141,995.52	142,983.63	143,921.85	145,870.04	−1.34	52.31	1.36	0.66
C5	100	1.49	8	15,120.42	21,491.92	21,439.80	21,860.34	22,826.37	−4.23	44.57	1.71	1.96
C6	200	1.49	16	39,560.47	44,406.50	43,998.33	45,171.83	46,348.02	−2.54	14.18	1.72	2.67
C7	400	1.49	32	68,626.96	90,591.24	89,258.58	88,532.95	93,473.47	−5.29	29.01	−2.27	−0.81
C8	800	1.49	64	182,554.55	183,513.87	184,072.16	184,410.85	187,772.44	−1.79	1.02	0.49	0.18
C9	100	1.19	10	25,022.94	25,891.34	25,587.59	26,270.03	27,652.19	−5.00	4.98	1.46	2.67
C10	200	1.19	20	52,323.04	53,646.57	52,357.71	54,509.68	56,207.01	−3.02	4.18	1.61	4.11
C11	400	1.19	40	107,809.05	109,858.78	106,787.04	111,520.14	113,554.19	−1.79	3.44	1.51	4.43
C12	800	1.19	80	143,922.78	221,795.74	219,300.70	226,059.91	228,447.48	−1.05	57.07	1.92	3.08
C13	100	1.00	12	28,833.02	29,473.86	29,146.85	29,208.18	30,744.93	−5.00	1.30	−0.90	0.21
C14	200	1.00	24	59,871.37	61,047.52	61,127.65	61,175.75	63,083.58	−3.02	2.18	0.21	0.08
C15	400	1.00	48	118,310.00	124,934.81	125,855.86	125,854.79	128,148.98	−1.79	6.38	0.74	0.00
C16	800	1.00	96	249,599.46	253,971.64	256,396.02	256,236.11	258,811.53	−1.00	2.66	0.89	−0.06
C17	100	0.85	14	31,275.43	32,318.08	31,916.88	32,057.27	32,895.98	−2.55	2.50	−0.81	0.44
C18	200	0.85	28	64,561.58	66,549.39	66,393.98	66,527.87	67,387.86	−1.28	3.05	−0.03	0.20
C19	400	0.85	56	131,630.39	135,543.70	135,824.75	135,897.71	136,599.06	−0.51	3.24	0.26	0.05
C20	800	0.85	112	266,275.54	273,692.34	274,770.81	274,817.89	275,280.60	−0.17	3.21	0.41	0.02

Table 12 Bounds for Hub-and-Spoke Network with Four Nonhub Locations

				Bound improvement (%)		% Difference across legs (%)	
Case #	CDLP bound	DCOMP bound	DCOMP1 bound	%-CDLP	%-DCOMP	DCOMP	DCOMP1
C1	18,313.87	17,714.14	17,693.73	3.39	0.12	3.10	0.40
C2	36,627.74	35,998.76	35,974.27	1.78	0.07	1.69	0.24
C3	73,255.49	72,623.62	72,596.09	0.90	0.04	0.87	0.12
C4	146,510.97	145,899.88	145,870.04	0.44	0.02	0.42	0.05
C5	23,581.37	22,826.37	22,826.37	3.20	0.00	3.27	0.30
C6	47,162.74	46,348.02	46,348.02	1.73	0.00	1.76	0.09
C7	94,325.47	93,473.47	93,473.47	0.90	0.00	0.91	0.01
C8	188,650.94	187,772.44	187,772.44	0.47	0.00	0.47	0.00
C9	28,758.99	27,652.19	27,652.19	3.85	0.00	4.00	1.52
C10	57,517.98	56,207.01	56,207.01	2.28	0.00	2.33	0.96
C11	115,035.95	113,554.19	113,554.19	1.29	0.00	1.30	0.57
C12	230,071.90	228,447.48	228,447.48	0.71	0.00	0.71	0.33
C13	32,853.46	31,293.97	30,744.93	6.42	1.75	4.98	0.00
C14	65,706.92	63,710.63	63,083.58	3.99	0.98	3.13	0.00
C15	131,413.85	128,887.47	128,148.98	2.48	0.57	1.96	0.00
C16	262,827.70	259,693.57	258,811.53	1.53	0.34	1.21	0.00
C17	34,707.96	33,107.58	32,895.98	5.22	0.64	4.83	0.00
C18	69,415.93	67,636.88	67,387.86	2.92	0.37	2.63	0.00
C19	138,831.85	136,946.88	136,599.06	1.61	0.25	1.38	0.00
C20	277,663.71	275,760.68	275,280.60	0.86	0.17	0.69	0.00

expectation. For cases with medium load factors (C5 to C12), the performance gains are quite substantial. Overall, the results for C1–C20 are consistent with our earlier observations. Table 6 shows that with very few exceptions the empirical load factors for DCOMP1 are higher than DCOMP and CDLP10. Table 12 shows the performance of various bounds for the test cases C1–C20. Overall, the bound performance for these cases are quite consistent to that of cases B1–B20.

7. Summary

We introduce a new dynamic programming decomposition approach to network RM based on a nonlinear nonseparable functional approximation, which leads to a better upper bound on the optimal expected revenue and better heuristic policies than the classical dynamic programming decomposition introduced in the literature. Policy performance from the new approach is tested in an extensive numerical study that shows the strength of the proposed approach. To the extent that the computational cost of the new approach is only slightly higher than the classical version, it is more appealing than several other approaches recently introduced in the literature. We believe the approach has potential for practical implementation.

Functional approximation approaches for dynamic programs are relatively new in the RM literature. Our work confirms, once again, the usefulness of such an approach. We believe that this approach lends itself well to other applications, such as supply chain management, and multiproduct multiperiod inventory allocation problems with substitution (Shumsky and Zhang 2009).

Acknowledgments

This research is partially supported by a research grant from Natural Sciences and Engineering Research Council of Canada (NSERC) and a faculty research grant from Desautels Faculty of Management, McGill University. The author is grateful for helpful comments from Editor Steve Graves, the associate editor, two anonymous referees, Adam Mersereau, and Tamer Boyaci on earlier versions of the paper. The author also benefited from general discussions with Daniel Adelman on relevant topics. The author, as usual, assumes responsibility for any errors.

References

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

Adelman, D., A. J. Mersereau. 2008. Relaxations of weakly coupled stochastic dynamic programs. *Oper. Res.* **56**(3) 712–727.

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

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

Bertsimas, D., I. Popescu. 2003. Revenue management in a dynamic network environment. *Transportation Sci.* **37**(3) 257–277.

Bront, J. M., I. Mendez-Diaz, G. Vulcano. 2009. A column generation algorithm for choice-based network revenue management. *Oper. Res.* **57**(3) 769–784.

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

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

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

Farias, V. F., B. Van Roy. 2007. An approximate dynamic programming approach to network revenue mangement. Working paper, MIT Sloan School of Management, Cambridge, MA.

Gallego, G., G. J. van Ryzin. 1997. A multiproduct dynamic pricing problem and its applications to network yield management. *Oper. Res.* **45**(1) 24–41.

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

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

Kunnumkal, S., H. Topaloglu. 2008a. A new dynamic programming decomposition method for the network revenue management problem with customer choice behavior. Working paper, Cornell University, Ithaca, NY.

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

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

Meissner, J., A. K. Strauss. 2009. Network revenue management with inventory-sensitive bid prices and customer choice. Working paper, Department of Management Science, Lancaster University, Lancaster, UK.

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

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

Secomandi, N. 2008. An analysis of the control-algorithm re-solving issue in inventory and revenue management. *Manufacturing Service Oper. Management* **10**(3) 468–483.

Shumsky, R. A., F. Zhang. 2009. Dynamic capacity management with substitution. *Oper. Res.* **57**(3) 671–684.

Talluri, K. 2008. On bounds for network revenue management. Working paper, Universitat Pompeu Fabra, Spain.

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

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

Talluri, K., G. J. van Ryzin. 2004b. *The Theory and Practice of Revenue Management*. Springer, New York.

van Ryzin, G. J., G. Vulcano. 2008. Simulation-based optimization of virtual nesting controls under consumer choice behavior. *Manufacturing Service Oper. Management* **10**(3) 448–467.

Walczak, D. 2008. A product-focused approach to pricing optimization on a network. Presentation at INFORMS 2008 Annual Meeting, Washington, DC.

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

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

Zhang, D., W. L. Cooper. 2009. Pricing substitutable flights in airline revenue management. *Eur. J. Oper. Res.* **197**(3) 848–861.

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