Lancaster University Management School
Working Paper
2011/014

Dynamic Simultaneous Fare Proration for Large-Scale Network Revenue Management

Philipp Kemmer, Arne Karsten Strauss and Thomas Winter

The Department of Management Science
Lancaster University Management School
Lancaster LA1 4YX
UK

Dynamic Simultaneous Fare Proration for Large-Scale Network Revenue Management

Philipp Kemmer[*], Arne Strauss[†], Thomas Winter[*]

April 5, 2011

Abstract

Network revenue management is concerned with managing demand for products that require inventory from one or several resources by controlling product availability and/or prices in order to maximize expected revenues subject to the available resource capacities. One can tackle this problem by decomposing it into resource-level subproblems that can be solved efficiently, e.g. by dynamic programming (DP).

We propose a new dynamic fare proration method specifically having large-scale applications in mind. It decomposes the network problem by fare proration and solves the resource-level dynamic programs simultaneously using simple, endogenously obtained dynamic marginal capacity value estimates to update fare prorations over time. An extensive numerical simulation study demonstrates that the method results in tightened upper bounds on the optimal expected revenue, and that the obtained policies are very effective with regard to achieved revenues and required runtime.

Keywords: Transport; Revenue Management; Dynamic Programming; Air Transport

Introduction

Many service providers sell products that require inventory on one or several resources, for example, flight tickets that require seats on one or several flight legs, or hotel stays that require rooms over one or several nights, etc. In this context, the network revenue management (RM) problem is to manage demand for such network products by controlling product availability and/or prices in order to maximize expected revenues subject to the available resource capacities. The interdependencies between the various resources cause the identification of an exact solution to this problem to be intractable; instead, we can only solve the problem approximately.

Much work has been devoted to the construction of such approximate solutions (see literature in the following section). A popular approach is to decompose the network problem into easier, independent subproblems, one per resource, by replacing the fare for products requiring inventory from more than one resource with resource-level fare allocations based on some estimate of the marginal capacity value (MCV) of each resource. Recent research confirmed the intuition that consideration of time-dependent estimates (as opposed to static ones) can improve the average revenue performance of policies derived from such methods (e.g. Adelman (2007), Topaloglu (2009)). However, this improved revenue performance comes at the expense of computationally more demanding models.

We propose an approximate solution to the network RM problem that generates and uses time-dependent estimates of the MCVs at virtually the same computational cost as the traditional decomposition technique. The key idea is to solve the resource-level subproblems simultaneously, and to exchange

[*]Lufthansa Systems, Salzufer 8, 10587 Berlin, Germany.

[†]Lancaster University Management School, Lancaster LA1 4YX, United Kingdom. Email: a.strauss@lancaster.ac.uk

information on new estimates of the MCVs between these subproblems so as to obtain new fare allocations to the individual resources at every point on a discrete time scale. We compare this new method with a simplified version of the optimization methodology that is being used by Lufthansa Systems. Their actually used complex methodology includes many additional features such as customer choice modeling, overbooking, cancelations, risk-based capacity optimization etc; we ignore these for the sake of a clearer presentation of the new idea. We emphasize that these features can easily be added to the new model in the same way as for the simplified version with similar increase of computational cost. This justifies our claim for applicability to large-scale networks since their complex methodology has been successfully applied to airline RM networks with more than 1000 flight legs.

We present numerical results on public benchmark data that demonstrates that the new method can obtain significantly higher average revenues than the traditional dynamic programming (DP) decomposition with prorated fares at about the same computational cost. Further, it can be much faster whilst maintaining statistically the same average revenue levels if compared to iterative dynamic programming decomposition via fare proration with static proration factors. We also show a new upper bound relationship, namely that the DP decomposition with prorated fares is a tighter upper bound on the optimal expected revenue than the so-called Deterministic Linear Program.

The work is organized as follows: after a brief literature review, the network RM problem is formally stated along with the required mathematical notation. As an approximate way to solving this problem, we present an iterative dynamic programming decomposition via fare proration that serves us as a benchmark to measure performance of the new approximation proposed in the subsequent sections. We report numerical results in the penultimate section and draw conclusions in the final section.

Related Work

There is a large body of literature on the subject of network revenue management, and we only outline the most related work on fare proration or decomposition techniques. For a general overview, consult the book of Talluri and van Ryzin (2004b) and the survey papers Chiang et al. (2007) and McGill and van Ryzin (1999).

The solution approaches to the network RM problem often take the form of either a mathematical programming formulation, or a decomposition into a collection of single-resource problems, or a combination of both. Any of these approaches eventually leads to estimates of opportunity costs that can be used to construct policies.

Popular mathematical programming approaches include the so-called Deterministic Linear Program (DLP) and the Probabilistic Nonlinear Program (PNLP), see Glover et al. (1982); Williamson (1992). A more recent alternative is due to Adelman (2007): he proposes a time-dependent approximation and shows that upper bounds on the optimal objective value are tightened relative to the DLP, and that the obtained policies perform better in a simulation study. A problem with this formulation is the large number of columns in the linear program; in fact, it grows exponentially in the number of products. The author hence proposed to solve it via column generation, which, however, can be (too) time-consuming for large-scale networks. Farias and Van Roy (2007) introduce a linear programming approach to approximate dynamic programming that depends on both time and inventory levels. The same approximation was independently proposed by Talluri (2008) who studies the relationships of various upper bounds obtained by different methods.

Decomposition approaches aim to simplify the network problem by breaking up the links in the problem structure that cause the resources to be interdependent, so that one eventually obtains a collection of simpler subproblems. Under the assumption of independent demand, these links are the inventory consumption and fare of products that use more than one resource.

Jiang (2008) uses Lagrangian relaxation to break up interdependencies across the capacity constraints in order to solve the aforementioned PNLP. He proposes this technique as an alternative to tackling the PNLP with general non-linear solvers. The relaxed problem decomposes by the products, and he shows that the subproblems can be solved analytically (for fixed Lagrangian multipliers). The latter are being improved using subgradient optimization with the disadvantage of slow convergence. Topaloglu (2009) decomposes the network RM problem by likewise using Lagrangian relaxation of the constraints that link resources via the inventory consumption of network products. He obtains a time- and inventory-level-dependent approximation that allows acceptance or rejection of a product on the individual resources that the product requires. Subgradient optimization finds a feasible solution by penalizing discrepancies of these resource-level accept/deny decisions. The evaluation of a subgradient requires solving a single-dimensional dynamic program for all resources. The convergence of this procedure can be quite slow, requiring hundreds of iterations.

Most decomposition techniques focus on breaking up the network fares. Kunnumkal and Topaloglu (2010) develop an approach where optimal partitions of network fares into resource-level allocations are found using subgradient optimization. Each evaluation of a subgradient requires solution of single-resource dynamic programs. The main advantage over previous fare decomposition methods is that it adjusts the fare allocations from iteration to iteration.

A traditional approach is to combine mathematical programming and decomposition: first, we solve a mathematical program resulting in some estimate of the marginal value of capacity for each resource, and then we use these estimates to define fare allocations to the individual resources. These allocations can be obtained additively or multiplicatively, and we refer to them as displacement-adjusted or prorated fares, respectively. Talluri and van Ryzin (2004a) describe the traditional dynamic programming approach using displacement-adjusted fares based on an initial solution of the DLP. Fare proration was proposed in the Master thesis of Bratu (1998) based on capacity value estimates derived from the Expected Marginal Seat Revenue heuristics of Belobaba (1987, 1986).

Variants of this decomposition method became popular in practice. Recent research aimed at improving the shortcoming that after the initial fare allocation, all network information is lost. While this allows us to solve the subproblems in parallel since they are independent, one would expect revenue improvements if some information would be exchanged. This idea underpins the work of Zhang (2011), who proposes to use the static marginal capacity value estimates obtained from the DLP to decompose the problem as in the traditional approach, however, since we can obtain an upper bound on the optimal expected revenue from each of these subproblems at each time step, he proposes to solve the subproblems in parallel, and to use the tightest bound at each time step as the value function approximation. His work is related to ours in that we also solve the DP subproblems in parallel and exchange information between them, however, the main difference is that we use information from all subproblems to define dynamic estimates of the marginal value of capacity, and use them to obtain new fare allocations by proration to reflect changes of the capacity values over time.

The Network Revenue Management Problem

We face a network with m resources and n products. A product j is a seat on one or several flight legs and has a fixed fare f_j and potentially some fare rules associated with it. The set of all resources and products is denoted by $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$, respectively. The matrix $A \in \{0, 1\}^{m,n}$ with components a_{ij} specifies whether product j requires resource i for all product-resource pairings. We assume that there are no group requests. We write A_j for the jth column of A and A^i for its ith row.

The notation $i \in A_j$ ($j \in A^i$) represents resources i that are used by product j (products j that use resource i).

Customers arrive continuously over time while decisions on which products to offer are made at discrete points in time such that the time intervals are small enough to have a negligible probability that two or more arrivals occur. The decision time periods are indexed with t, starting at time $t = 1$ until all flights depart at time $t = T + 1$. The index t can also refer to the time interval between decisions at t and $t + 1$ and will be clear from the context. In practice, the time grid can differ from one resource to another since we are interested in adjusting the grid for expected demand for this resource on the one hand, but not letting the grids become unnecessarily fine and thereby increasing computational time on the other.

Given that we offer a set $S \subseteq J$ of products at time t, a customer purchases product $j \in S$ with probability p_{jt} and does not purchase with probability $p_{0t} = 1 - \sum_{j \in S} p_{jt}$. We assume that all customers show up and do not cancel so that no overbooking is required.

Each resource i initially has capacity c_i, and the state vector $x \in \mathbb{N}_0^m$ indicates how much inventory is still available. Although x is clearly time-dependent, we do not use a subscript t because it will be clear from the context. The remaining inventory also affects which products can be offered; since we exclude overbooking, we require that sufficient inventory must be available to provide a product. The set of all feasible products is then $J(x) = \{j \in J : A_j \leq x\}$.

Let us denote the optimal expected revenue obtainable from time t until the end of the booking horizon given remaining capacity x by $v_t(x)$, usually referred to as the value function. A common assumption in recent work on this kind of network RM problem is that we can offer any combination of products at any time; subject to sufficient remaining inventory. Under this assumption, $v_t(x)$ can be written as follows:

$$\begin{aligned} v_t(x) &= \max_{S \subseteq J(x)} \sum_{j \in S} p_{jt} \Big[f_j + v_{t+1}(x - A_j) \Big] + p_{0t} v_{t+1}(x) \\ &= \max_{S \subseteq J(x)} \left\{ \sum_{j \in S} p_{jt} \Big[f_j - \big(v_{t+1}(x) - v_{t+1}(x - A_j) \big) \Big] \right\} + v_{t+1}(x), \\ &= \sum_{j \in J(x)} p_{jt} \Big[f_j - \big(v_{t+1}(x) - v_{t+1}(x - A_j) \big) \Big]^+ + v_{t+1}(x), \qquad \forall\, t, x, \end{aligned} \tag{1}$$

where $[z]^+$ is defined as $\max\{z, 0\}$. The boundary conditions are given by $v_{T+1}(x) = 0$ for all inventory states x. Note that the expression $\big(v_{t+1}(x) - v_{t+1}(x - A_j)\big)$ represents the opportunity cost of selling product j. Computation of $v_t(x)$ for all time periods t and all inventory vectors $x \in \bigotimes_{i \in I} \{0, \ldots, c_i\}$ is not tractable, so instead we are looking for a good approximation of $v_t(x)$ that we can use as an approximation of the opportunity cost. The resulting policy is to offer only those products for which fare minus approximate opportunity cost is positive.

In the next section, we introduce the iterative fare proration method that serves us as a benchmark in our numerical study.

Iterative Dynamic Programming Decomposition via Fare Proration

Fare proration means the allocation of portions of any network fare f_j to the resources that product j uses by defining a vector of so-called proration factors $\zeta \in [0, \infty)^m$ that is used to obtain allocations $f_j^i := f_j \zeta_i / \sum_{k \in A_j} \zeta_k$ for each resource i if $\sum_{k \in A_j} \zeta_k \neq 0$. In the event that the denominator is zero for a product j, we distribute the revenue equally across all resources via $f_j^i := f_j/|A_j|$ for all $i \in A_j$, where $|A_j|$ denotes the number of nonzero components of A_j. If we substitute the fare f_j with the fare allocation f_j^i for a fixed resource i in the dynamic program (1), then we obtain a collection of easy-to-solve dynamic programs defined over a single resource:

$$v_{t,i}^{\zeta}(x_i) = \sum_{j \in J(x_{-i})} p_{jt} \Big[f_j^i - \big(v_{t+1,i}^{\zeta}(x_i) - v_{t+1,i}^{\zeta}(x_i - a_{ij}) \big) \Big]^+ + v_{t+1,i}^{\zeta}(x_i), \qquad \forall\, t, x_i \in \{0, \dots, c_i\}, \tag{2}$$

with $x_{-i} := [c_1, \dots, x_i, \dots, c_m]$ and boundary conditions $v_{t,i}^{\zeta}(0) = v_{T+1,i}^{\zeta}(x_i) = 0$ for all t and x_i. Having solved all resource-level value functions $v_{t,i}^{\zeta}(x_i)$, we can approximate $v_t(x)$ with $\sum_i v_{t,i}^{\zeta}(x_i)$.

Let us consider the construction of proration factors. A classic approach is to use the optimal dual values of the capacity constraints of the Deterministic Linear Program (DLP) as proration factors. The DLP considers demand for product j as deterministic and being equal to its expected value $\sum_t p_{jt}$, and is often found in industry solutions owing to its simplicity and relatively good performance. It is given by

$$\begin{aligned} z_{\text{DLP}} = &\max_y \langle f, y \rangle \\ \text{(DLP)} \quad &\sum_{j \in J} a_{ij} y_j \leq c_i, \qquad \forall\, i \in I, \\ &0 \leq y_j \leq \sum_t p_{jt}, \qquad \forall\, j \in J, \end{aligned} \tag{3}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product. The variable y_j is an allocation of capacity to product j, and the constraints ensure that the available capacity is not exceeded. The optimal dual variables π of the capacity constraints (3) represent a (static) estimate of the marginal value of capacity. Therefore, using $\zeta_i := \pi_i$ for all resources i allocates fares to resources proportional to the value of each respective resource.

An intriguing variant of this process is to run it once, to use the resulting information to obtain updated estimates of the marginal value of capacity and thus new proration factors, and then to restart the process using this new proration until some stopping criterion is met. Although there are typically neither guarantees that the proration factors will become better nor that they converge, still this approach has turned out successful in practice at Lufthansa Systems. We define this method in Algorithm 1, where we obtain the initial proration factors from the DLP in line 1, define the denominator d_j^h of the proration term for product j at iteration h in line 4, prorate the fare between lines 5 and 10, and solve the dynamic programs (2) with $\zeta := \pi$ in line 12. Accordingly, we denote these single-resource value functions as $v_{t,i}^{\pi,h}(x_i)$. After this first iteration we check a stopping criterion, and if it is not met, we define new proration factors $\zeta_i^{h+1} := v_{1,i}^{\pi,h}(c_i) - v_{1,i}^{\pi,h}(c_i - 1)$ and re-evaluate the dynamic programs. The stopping criterions are discussed in the penultimate section. We use Algorithm 1 as a benchmark method in our numerical experiments.

Properties

An attractive feature of decomposition by fare proration is that we are always guaranteed to obtain an upper bound on the optimal expected revenue given by $v_t(x)$ as expressed by the following Lemma:

Lemma 1. *Let there be a vector α such that $\sum_{i \in I} \alpha_{ijt} = f_j$ for all products j and all time periods t. Then $v_t(x) \leq \sum_{i \in I} v_{t,i}(x_i|\alpha)$, where $v_{t,i}(x_i|\alpha)$ is the value function obtained by substituting the prorated revenues f_j^i in (2) with α_{ijt}.*

Proof. This result can be shown by induction over time and is stated in Kunnumkal and Topaloglu (2010). □

Algorithm 1 Benchmark: Iterative Fare Proration

```
1: obtain initial marginal capacity value estimates $\zeta^h \leftarrow \pi$ from DLP for iteration $h = 1$
2: boundary conditions: $v^{\pi,h}_{T+1,i}(x_i) = 0$ for all $x_i, i$. $v^{\pi,h}_{t,i}(0) = 0$ for all $t, i$
3: loop
4:     $d^h_j \leftarrow \langle A_j, \zeta^h \rangle$ for all products $j$
5:     for all products $j$ with $d^h_j > 0$ do
6:         $f^i_j \leftarrow \zeta^h_i f_j / d^h_j$ for all resources $i \in A_j$
7:     end for
8:     for all products $j$ with $d^h_j = 0$ do
9:         $f^i_j \leftarrow f_j / |A_j|$ for all resources $i \in A_j$
10:    end for
11:    for all $i = 1 : m$ do
12:        solve $v^{\pi,h}_{t,i}(x_i) = \sum_{j \in J(x_{-i})} p_{jt} \left[ f^i_j - \left( v^{\pi,h}_{t+1,i}(x_i) - v^{\pi,h}_{t+1,i}(x_i - a_{ij}) \right) \right]^+ + v^{\pi,h}_{t+1,i}(x_i)$   for all $t, x_i$
13:    end for
14:    update proration factors: $\zeta^{h+1}_i \leftarrow v^{\pi,h}_{1,i}(c_i) - v^{\pi,h}_{1,i}(c_i - 1)$ for all $i$
15:    if stopping criterion satisfied then
16:        break
17:    else
18:        $h \leftarrow h + 1$
19:    end if
20: end loop
21: return value function approximations $v^i$ for all $i$
```

Note that this holds regardless of the way that the allocations are made. On the other hand, we are also interested in methods that result in tight bounds since these can be expected to generally produce better average revenue results. Indeed, for fare proration based on the dual values of the DLP with subsequent solution of the dynamic programs (2), we can show that we achieve a tighter bound after one DP iteration than the one given by the optimal objective z_{DLP} of DLP:

Proposition 2. $v_1(c) \leq \sum_{i \in I} v^{\pi,1}_{1,i}(c_i) \leq z_{\text{DLP}}$.

The proof can be found in the Appendix. Note that this result only holds when we use the DLP dual values as initial prorating factors; it is not clear whether the bound is tighter after more than one iteration of Algorithm 1. However, numerical results in the penultimate section indicate that iterations can tighten the bound with respect to z_{DLP}.

Practical Considerations

Iterative dynamic programming decomposition via fare proration is a proven concept that delivers good revenue and run time performance at Lufthansa Systems. Even for large-scale networks with more than 1000 flight legs, the method can run within the time limitations given the hardware equipment typically found at large airlines (even with all additional features such as customer choice models etc) because the single-resource DPs are independent from each other and can consequently be solved in parallel. The number of DP iterations typically averages around five, though that of course varies with network structure and demand.

Dynamic Simultaneous Fare Proration

Although the iterative DP decomposition via fare proration based on the DLP has appealing theoretical and practical properties, there still appears to be scope for improvement: various researchers have recently

Algorithm 2 Dynamic Simultaneous Fare Proration

```
1:  define boundary conditions: $v^{\xi}_{T+1,i}(x_i) = 0$ for all $x_i, i$. $v^{\xi}_{t,i}(0) = 0$ for all $t, i$.
2:  for all $t = T : -1 : 1$ do
3:      for all $i \in I$ do
4:          update $\xi_i \leftarrow v^{\xi}_{t+1,i}(c_i)/c_i$
5:          $d_j \leftarrow \langle A_j, \xi \rangle$ for all products $j$
6:          for all products $j \in A^i$ with $d_j > 0$ do
7:              $f^i_j \leftarrow \xi_i f_j / d_j$
8:          end for
9:          for all products $j \in A^i$ with $d_j = 0$ do
10:             $f^i_j := f_j / |A_j|$
11:         end for
12:         solve $v^{\xi}_{t,i}(x_i) = \sum_{j \in J(x_{-i})} p_{jt} \Big[ f^i_j - \big( v^{\xi}_{t+1,i}(x_i) - v^{\xi}_{t+1,i}(x_i - a_{ij}) \big) \Big]^+ + v^{\xi}_{t+1,i}(x_i)$
13:     end for
14: end for
15: return value function approximations $v^i$ for all $i$
```

demonstrated the benefit of accounting for time-dependence of estimates of the marginal capacity values (see literature review), yet the approach as outlined above is based on the static dual solution of the DLP. Typically, the corresponding approaches become difficult to solve for large-scale networks since they are based on slow-converging strategies such as column generation or subgradient optimization.

In this section, we investigate a new DP decomposition approach to the network RM problem that exploits simultaneous solution of the DPs and endogenously creates time-dependent proration factors without need for DP iterations. It uses the knowledge of the resource-level value functions as soon as it becomes available.

We call this new method Dynamic Simultaneous Proration (DSP) and outline it in Algorithm 2. More specifically, we initially allocate equal fare proportions f^i_j for any network product j to all the resources i that it uses. This allocation is motivated by the fact that all MCV are zero at departure. In line 4, the estimated MCV of resource i is defined as the MCV averaged over all inventory levels l: $\sum_{l=1}^{c_i} \big(v^{\xi}_{t+1,i}(l) - v^i_{t+1}(l-1) \big) / c_i = v^{\xi}_{t+1,i}(c_i)/c_i$. We compute new prorated fares (lines 5–11) and use these in the calculation of the value function at time t (line 12). At each time step, the value functions for all resources need to be evaluated (in other words, we solve the DPs simultaneously).

This approach has several interesting features: Firstly, it does not require any initial prorating factors, and afterwards we can use the previously computed MCV estimates from the resource-level value functions themselves to obtain new MCVs (i.e. new proration factors). Therefore, all MCVs are obtained endogenously from the model. Secondly, these endogenous updates of the MCV within the DP backwards recursion make iterative proration superfluous. In contrast, the iterative fare proration in the benchmark method attempts to correct the static nature of the prorating factors by updating them after each DP iteration. Thirdly, the proration updates are very simple to compute and incur virtually no additional computational cost as compared to Algorithm 1.

Properties

We know from Lemma 1 that $\sum_{i \in I} v^{\xi}_{1,i}(c_i)$ is an upper bound on the optimal expected revenue $v_1(c)$, and intuitively one would expect that the dynamic DP decomposition would result in a bound that is tighter than the benchmark $\sum_{i \in I} v^{\pi,1}_{1,i}(c_i)$. Unfortunately, this is in general not the case. In fact, we show with an example that $\sum_{i \in I} v^{\xi}_{1,i}(c_i)$ is not even guaranteed to be tighter than the optimal objective of (DLP).

Proposition 3. *In general, z_{DLP} and $\sum_{i \in I} v^{\xi}_{1,i}(c_i)$ as defined in Algorithm 2 do not dominate each* *other.*

Proof. Consider a network with two products and two flights. The fares are given by $f = [100, 50]$, product 1 uses both flight legs and product 2 uses only the first flight leg. The time horizon is $T = 50$, and $p_t = [0.1, 0.1]$ for all t. Let the initial capacity be $c = [10, 1]$. It is easy to see that the DLP objective is 350 since we would allocate the sales of 1 unit of product 1 and 5 units of product 2. However, $\sum_i v^{\pi,1}_{1,i}(c_i) \approx 349 \leq z_{\text{DLP}} = 350 < \sum_{i\in I} v^{\xi}_{1,i}(c_i) \approx 395$. On the other hand, in the penultimate section we report examples where $\sum_{i\in I} v^{\xi}_{1,i}(c_i) \leq \sum_i v^{\pi,1}_{1,i}(c_i) \leq z_{\text{DLP}}$. □

The example above illustrates how the new method can go wrong: the dual solution to (D) for this network is $\pi = [0, 100]$, whereas Algorithm 2 starts with an initial guess of $\xi_T = [0, 0]$. This behavior arises from the absence of information concerning the MCVs of the resource with zero capacity, whereas the static dual solution π reflects that this resource is the bottleneck. We could smooth this effect by constructing proration factors using a convex combination of prior information such as π obtained from the DLP as well as the dynamic endogenous estimates.

However, we observe in our extensive numerical study that the revenue and upper bound performance of the new approach is very good across all test instances without using any prior information. The method is computationally very attractive since it does not require the expensive up-front computation of time-dependent MCV. We further discuss the MCV estimates obtained with this method in the numerical results section.

Practical Considerations

As run time performance is crucial for any real-life application, we need to make sure that the exchange of information between parallel single-resource DPs does not slow the entire system down. Moreover, the time grids in real applications can differ from resource to resource, so that it is not immediately clear how to apply this concept. On the other hand, in the real system there are a number of time points (usually about 20) at which new forecasting data becomes available, and at these so-called data collection points the time grids are synchronous. Therefore we suggest to perform the update of proration factors at these data collection points; this will also result in a low amount of information that needs to be exchanged. In the numerical experiments that we describe in the following section, we address this issue by testing both updates at every time step (under the assumption of synchronous time grids) and at only 20 fixed points in time.

Numerical Experiments

Network Instances

We investigate the numerical behavior of various policies on the 48 single hub network instances provided by Topaloglu (2011), available for download online. Associated with each one of the ν spokes, there are two flight legs, one of which is to the hub and the other one is from the hub. For each origin-destination pair, there is one high-fare and one low-fare product. Consequently, there are 2ν flight legs and 4ν products for direct flights, and $2\nu(\nu - 1)$ products itineraries using two flight legs. Each of the 48 instances is characterized by (T, ν, α, κ), that is, the number of time periods $T \in \{200, 600\}$, the number of spokes $\nu \in \{4, 5, 6, 8\}$, tightness of leg capacities $\alpha \in \{1.0, 1.2, 1.6\}$, and the ratio between low and high class fares $\kappa \in \{4, 8\}$. By "tightness of leg capacity" we refer to the ratio of total expected demand for the capacity on all flights over total initial network capacity, that means

$$\alpha := \frac{\sum_{t=1}^{T} \sum_{i \in I} \sum_{j \in J} p_{jt} a_{ij}}{\sum_{i \in I} c_i}.$$

We define 20 time points on each time grid as "data collection points", i.e. $\{200, 190, 180, \ldots, 10\}$ for the scenarios with $T = 200$, and $\{600, 570, \ldots, 30\}$ for those with $T = 600$. At these points in time there is no actual data collection taking place (since the data does not provide us with new demand information); we merely introduce this notion to reflect that in realistic implementations the time grids are only synchronous at about 20 data collection points where we perform proration factor updates and re-solves.

Policies & Implementation

The iterative proration benchmark defined in Algorithm 1 requires some stopping criterion that we have yet to define. It is not clear whether the method converges in some sense, be that convergence of the proration factors or of the prorated fares themselves. We experimented with the following stopping criteria:

Factor Stop after iteration h if $\|\xi^h - \xi^{h+1}\|_\infty \leq 5$.

Fare Let us denote the prorated fare for product j on resource i under proration factors ξ by $f_j^i(\xi)$. Define the number *nConverged* of "converged" prorated fares as the number of such fares for which $|f_j^i(\xi^h) - f_j^i(\xi^{h+1})| \leq 5$. Let *nProratedFares* denote the total number of prorated fares. Finally, the scalar *meanFareDiff* denotes the fare difference $|f_j^i(\xi^h) - f_j^i(\xi^{h+1})|$ averaged over all prorated fares. The stopping criterion can be stated as follows:

```
if nConverged / nProratedFares ≥ 90%
-   if nConverged / nProratedFares == 1, stop, endif
-   if meanFareDiff ≤ 5, stop, endif
endif
```

We allow at most 10 iterations.

1Iter As a third alternative, we stop always after the first iteration.

On average, the stopping criterion **Factor** turned out to produce smaller revenues and to require slightly more iterations so that we confine ourselves to reporting results only for **Fare** and **1Iter**.

We tested the following policies, all of which are based on 20 re-solves of the corresponding underlying method at equally spaced time points over the entire time horizon:

IPBMK The Iterative Proration Benchmark as outlined in Algorithm 1. We use the stopping criterion **Fare**. Each call to the algorithm provides us with a collection of single-leg value function approximations v_i^ζ, which we use to accept a product at time t if there is sufficient remaining inventory and $f_j - \sum_{i \in I} a_{ij} \Delta v_{t+1,i}^\zeta(x_i) \geq 0$.

BMK1 This is the same benchmark method as IPBMK except that we always stop after the first DP iteration (i.e., the IPBMK with stopping criterion **1Iter**).

DSPt We run the Dynamic Simultaneous Proration as outlined in Algorithm 2 with prorating factor updates at every time step t.

DSP Same as DSPt, except with only 20 prorating factor updates.

DLP We re-optimize the Deterministic Linear Program at each each data collection point to obtain a static estimate of the MCV by means of the dual solution π to the capacity constraints, and offer all products j for which there is sufficient inventory ($x_i \geq a_{ij} \forall i \in A^j$) and positive contribution $f_j - \sum_i a_{ij}\pi_i \geq 0$.

Results

Upper Bounds

We begin with an investigation of the upper bounds on the optimal expected revenue obtainable from the various methods and report the results for all 48 test instances in Table 1 and 2. We find that the DLP bound is between 2.2% to 10.2% weaker than the IPBMK bound (on average +5.3% over all instances). BMK1 results in bounds that are between +0% to +2.7% weaker than IPBMK (on average +0.5%), while DSPt and DSP improve the IPBMK bound by up to -2.2% and -2.2%, respectively. In the worst cases, DSPt and DSP results in 0.4% and 0.5% weaker bounds (on average -0.1% and -0.0%, respectively).

Table 1: Upper bounds

(T,ν,α,κ)	IPBMK	BMK1	DSPt	DSP	DLP
(200,4,1.0,4)	20894	20930	20429	20442	21531
(200,4,1.0,8)	33348	33857	33250	33265	34571
(200,4,1.2,4)	18887	18887	18879	18897	19882
(200,4,1.2,8)	31640	31640	31641	31659	32922
(200,4,1.6,4)	16530	16534	16543	16569	17530
(200,4,1.6,8)	29243	29257	29248	29274	30570
(200,5,1.0,4)	21358	21556	21320	21325	22144
(200,5,1.0,8)	34421	34671	34384	34389	35387
(200,5,1.2,4)	20187	20343	20115	20121	21263
(200,5,1.2,8)	33134	33302	33052	33059	34495
(200,5,1.6,4)	17644	17644	17679	17695	18870
(200,5,1.6,8)	30484	30486	30491	30507	32081
(200,6,1.0,4)	21114	21372	21118	21128	22300
(200,6,1.0,8)	34104	34425	34106	34118	35544
(200,6,1.2,4)	19708	19730	19663	19686	20932
(200,6,1.2,8)	32597	32619	32544	32568	34172
(200,6,1.6,4)	17263	17263	17308	17338	18592
(200,6,1.6,8)	30073	30073	30101	30126	31824
(200,8,1.0,4)	18798	19302	18757	18789	20052
(200,8,1.0,8)	30277	30931	30253	30290	31835
(200,8,1.2,4)	17454	17463	17484	17530	18952
(200,8,1.2,8)	28850	28862	28865	28910	30727
(200,8,1.6,4)	15276	15292	15276	15323	16833
(200,8,1.6,8)	26526	26548	26525	26574	28608

We observe that all methods result in significantly strengthened bounds as compared to DLP. In particular, IPBMK appears to be already a very strong method as far as the generation of tight bounds is concerned, but can nevertheless be improved in some cases by dynamic simultaneous proration. Next, we investigate the revenue performance of the various policies.

Revenue Performance

We conduct an extensive numerical study of the revenue performance of all policies involving 2000 simulations for each scenario and each method. We report the average revenues and load factors in

Table 2: Upper bounds

(T, ν, α, κ)	IPBMK	BMK1	DSPt	DSP	DLP
(600,4,1.0,4)	31181	31482	30986	31005	32409
(600,4,1.0,8)	50613	50985	50391	50413	52086
(600,4,1.2,4)	28610	28610	28620	28652	29852
(600,4,1.2,8)	47923	47923	47945	47978	49529
(600,4,1.6,4)	25071	25073	25094	25141	26324
(600,4,1.6,8)	44334	44350	44345	44392	46001
(600,5,1.0,4)	32357	32424	32289	32296	33299
(600,5,1.0,8)	52156	52239	52068	52075	53285
(600,5,1.2,4)	30658	30834	30557	30568	31943
(600,5,1.2,8)	50277	50464	50165	50177	51904
(600,5,1.6,4)	26829	26829	26929	26963	28343
(600,5,1.6,8)	46309	46310	46373	46404	48283
(600,6,1.0,4)	25663	26221	25550	25561	26873
(600,6,1.0,8)	41355	42054	41258	41272	42865
(600,6,1.2,4)	23787	23820	23718	23747	25184
(600,6,1.2,8)	39363	39398	39289	39319	41166
(600,6,1.6,4)	20756	20756	20825	20864	22274
(600,6,1.6,8)	36239	36242	36297	36330	38252
(600,8,1.0,4)	22879	23340	22743	22782	24167
(600,8,1.0,8)	36804	37378	36659	36706	38395
(600,8,1.2,4)	21079	21098	21098	21161	22755
(600,8,1.2,8)	34862	34891	34872	34933	36976
(600,8,1.6,4)	18491	18509	18486	18545	20228
(600,8,1.6,8)	32117	32146	32112	32173	34449

Tables 3 and 4, along with standard deviations in Table 5. To facilitate reading of the revenue results, we depict the revenues of all policies relative to the revenue levels achieved by IPBMK in Figure 1. The relative percentage error of the reported results is about 0.2%–0.4%.

Figure 1 illustrates the strength of the methods IPBMK, BMK1, DSPt and DSP by the considerable improvements that they yield over DLP. In particular, for fixed (T, ν, α), a big difference between high and low fares ($\kappa = 8$) always results in significantly improved performances relative to DLP as compared to the improvements gained if $\kappa = 4$. Similarly, higher capacity tightness scenarios tend to show higher improvements over DLP. One would have expected these outcomes since big fare differences and relatively high demand should create more difficult decision problems so that the improved bid prices determined by the policies IPBMK, BMK1, DSPt and DSP have more impact.

In the same figure, we observe that IPBMK is always better than BMK1. This confirms the intuition that using iterative proration in conjunction with dynamic programming decomposition has a significant revenue impact. Averaged over all scenarios, IPBMK achieves a statistically significant +0.5% improvement of revenues relative to BMK1.

DSP generates in all but three scenarios the same revenue as IPBMK (deviations statistically insignificant). It is interesting to observe that DSPt and DSP are always very close; in fact, DSPt improves on DSP only about 0.1%. While we would expect DSP to generate less revenue than DSPt since it uses much fewer proration factor updates, we would have expected the difference to be larger. This shows that DSP can already realize most of the dynamic proration potential so that updates of proration factors at 20 data collection points are sufficient to maintain revenue performance on the level of IPBMK despite the considerably reduced computational cost. In order to provide some intuition concerning the small difference between DSPt and DSP, we plot the trajectories of the proration factors (note that we can interpret them as estimates of the marginal value of capacity) used by both methods in Figure 2 and

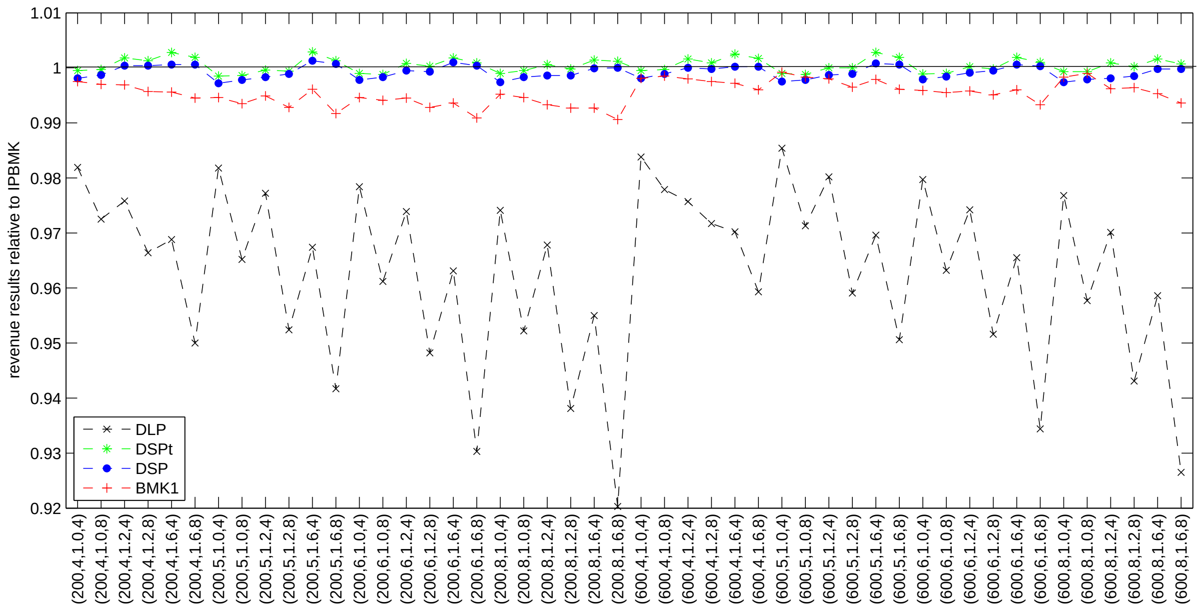

Figure 1: Average revenue performance of DLP, DSP and DSPt relative to IPBMK based on 2000 simulations.

Figure 3.

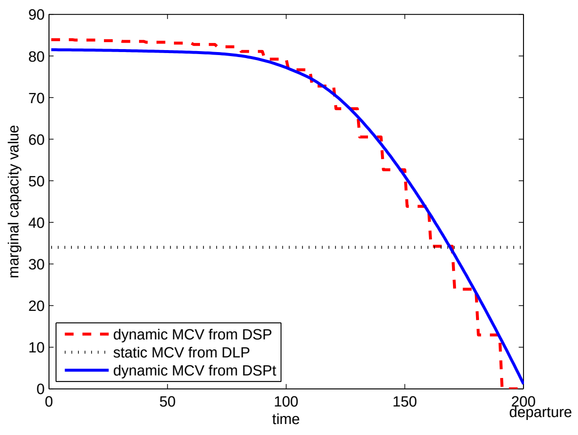

Figure 2: Marginal capacity value (MCV) estimates based on DLP, DSP and DSPt.

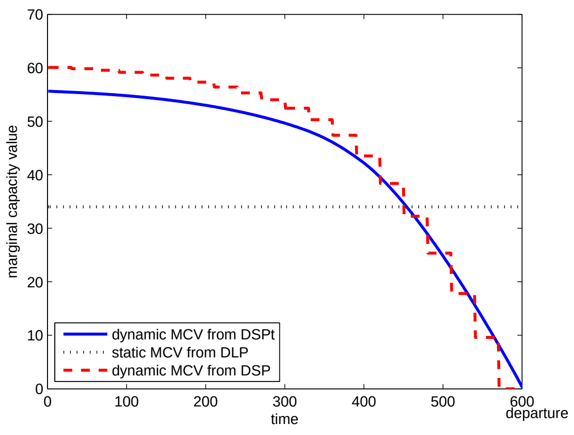

Figure 3: Marginal capacity value (MCV) estimates based on DLP, DSP and DSPt.

The proration factors in DSP are very close to those generated in DSPt. The approximation becomes a bit less accurate when we increase the time horizon whilst keeping the number of data collection points fixed at 20 (as one would expect), but is still a reasonably close fit. This helps to explain why the revenue results of DSPt and DSP are so similar. Furthermore, 600 time periods for a leg-based time grid are already close realistic grid sizes and therefore the revenue results look promising for real-world implementation.

Runtime Performance

The tests were executed on the High Performance Cluster at Lancaster University since each method is being re-solved 20 times over the time horizon of each simulation, thereby requiring a considerable

Table 3: Average revenue results and load factors based on 2000 simulations.

(T,ν,α,κ)	IPBMK	Load	BMK1	Load	DSPt	Load	DSP	Load	DLP	Load
(200,4,1.0,4)	20190	0.9	20139	0.9	20179	0.9	20151	0.91	19824	0.9
(200,4,1.0,8)	33025	0.88	32925	0.89	33015	0.89	32981	0.89	32118	0.89
(200,4,1.2,4)	18565	0.92	18507	0.92	18599	0.91	18572	0.91	18115	0.91
(200,4,1.2,8)	31337	0.89	31201	0.9	31379	0.88	31349	0.88	30285	0.9
(200,4,1.6,4)	16221	0.9	16150	0.91	16266	0.9	16231	0.9	15715	0.9
(200,4,1.6,8)	28925	0.87	28765	0.88	28980	0.86	28942	0.86	27480	0.9
(200,5,1.0,4)	21077	0.86	20963	0.86	21046	0.86	21018	0.86	20693	0.88
(200,5,1.0,8)	34120	0.84	33898	0.85	34071	0.84	34046	0.85	32934	0.89
(200,5,1.2,4)	19762	0.89	19662	0.89	19754	0.89	19729	0.89	19311	0.9
(200,5,1.2,8)	32653	0.86	32417	0.87	32635	0.86	32618	0.86	31098	0.91
(200,5,1.6,4)	17224	0.89	17156	0.9	17274	0.88	17246	0.88	16663	0.9
(200,5,1.6,8)	30020	0.85	29770	0.87	30059	0.85	30041	0.84	28269	0.9
(200,6,1.0,4)	20817	0.86	20704	0.87	20796	0.86	20772	0.86	20368	0.88
(200,6,1.0,8)	33792	0.84	33593	0.85	33752	0.84	33736	0.84	32480	0.88
(200,6,1.2,4)	19248	0.88	19143	0.89	19263	0.88	19238	0.88	18746	0.89
(200,6,1.2,8)	32118	0.85	31887	0.86	32127	0.85	32094	0.85	30453	0.9
(200,6,1.6,4)	16839	0.88	16732	0.89	16870	0.87	16855	0.87	16217	0.9
(200,6,1.6,8)	29628	0.83	29358	0.85	29654	0.83	29640	0.82	27562	0.9
(200,8,1.0,4)	18378	0.84	18289	0.85	18359	0.84	18330	0.84	17902	0.86
(200,8,1.0,8)	29820	0.82	29660	0.83	29805	0.82	29768	0.82	28395	0.86
(200,8,1.2,4)	17008	0.86	16894	0.86	17018	0.85	16985	0.85	16461	0.88
(200,8,1.2,8)	28350	0.82	28142	0.83	28345	0.82	28309	0.82	26596	0.88
(200,8,1.6,4)	14782	0.86	14674	0.87	14802	0.85	14781	0.85	14117	0.88
(200,8,1.6,8)	25956	0.81	25712	0.82	25987	0.8	25956	0.8	23888	0.88

Table 4: Average revenue results and load factors based on 2000 simulations.

(T,ν,α,κ)	IPBMK	Load	BMK1	Load	DSPt	Load	DSP	Load	DLP	Load
(600,4,1.0,4)	30653	0.92	30595	0.92	30639	0.92	30595	0.92	30156	0.91
(600,4,1.0,8)	49990	0.9	49915	0.91	49976	0.9	49937	0.9	48883	0.91
(600,4,1.2,4)	28181	0.93	28124	0.93	28227	0.92	28180	0.92	27495	0.92
(600,4,1.2,8)	47452	0.91	47333	0.91	47496	0.9	47443	0.9	46109	0.91
(600,4,1.6,4)	24660	0.92	24591	0.92	24722	0.91	24664	0.91	23924	0.92
(600,4,1.6,8)	43851	0.89	43674	0.9	43925	0.88	43861	0.88	42065	0.91
(600,5,1.0,4)	31926	0.88	31899	0.89	31893	0.87	31846	0.87	31459	0.89
(600,5,1.0,8)	51659	0.86	51569	0.87	51597	0.86	51544	0.86	50176	0.9
(600,5,1.2,4)	30091	0.9	30031	0.91	30091	0.9	30051	0.9	29496	0.91
(600,5,1.2,8)	49633	0.88	49459	0.89	49631	0.89	49577	0.89	47601	0.92
(600,5,1.6,4)	26344	0.91	26288	0.92	26417	0.9	26366	0.9	25543	0.92
(600,5,1.6,8)	45726	0.88	45546	0.89	45812	0.87	45755	0.87	43466	0.92
(600,6,1.0,4)	25217	0.87	25113	0.87	25190	0.87	25164	0.87	24706	0.89
(600,6,1.0,8)	40896	0.85	40713	0.85	40856	0.85	40830	0.85	39393	0.89
(600,6,1.2,4)	23274	0.89	23176	0.9	23278	0.89	23254	0.88	22674	0.9
(600,6,1.2,8)	38803	0.86	38614	0.87	38797	0.86	38782	0.86	36923	0.9
(600,6,1.6,4)	20304	0.88	20223	0.89	20342	0.88	20316	0.87	19604	0.9
(600,6,1.6,8)	35744	0.84	35505	0.86	35781	0.83	35753	0.83	33400	0.9
(600,8,1.0,4)	22282	0.85	22244	0.86	22266	0.85	22224	0.85	21766	0.87
(600,8,1.0,8)	36081	0.83	36044	0.84	36057	0.83	36005	0.82	34555	0.87
(600,8,1.2,4)	20519	0.87	20442	0.88	20538	0.86	20481	0.86	19906	0.89
(600,8,1.2,8)	34195	0.83	34072	0.85	34201	0.83	34143	0.83	32251	0.88
(600,8,1.6,4)	17896	0.87	17812	0.87	17925	0.86	17893	0.86	17155	0.89
(600,8,1.6,8)	31394	0.82	31194	0.84	31415	0.81	31389	0.81	29088	0.89

Table 5: Standard deviations for 2000 simulations.

(T, ν, α, κ)	IPBMK	BMK1	DSPt	DSP	DLP	(T, ν, α, κ)	IPBMK	BMK1	DSPt	DSP	DLP
(200,4,1.0,4)	960	968	958	958	967	(600,4,1.0,4)	1623	1556	1597	1578	1543
(200,4,1.0,8)	2054	2067	2043	2040	2069	(600,4,1.0,8)	3502	3393	3464	3437	3209
(200,4,1.2,4)	911	916	918	912	943	(600,4,1.2,4)	1466	1412	1472	1452	1440
(200,4,1.2,8)	2004	2022	2020	2015	2026	(600,4,1.2,8)	3374	3228	3368	3330	3086
(200,4,1.6,4)	891	888	898	886	909	(600,4,1.6,4)	1417	1367	1423	1403	1356
(200,4,1.6,8)	1984	1989	2005	1998	1950	(600,4,1.6,8)	3293	3142	3333	3295	2888
(200,5,1.0,4)	1131	1074	1143	1144	1134	(600,5,1.0,4)	1852	1791	1830	1816	1733
(200,5,1.0,8)	2343	2197	2348	2343	2334	(600,5,1.0,8)	3816	3654	3762	3738	3432
(200,5,1.2,4)	1052	1011	1062	1054	1054	(600,5,1.2,4)	1670	1610	1653	1635	1558
(200,5,1.2,8)	2262	2151	2265	2258	2224	(600,5,1.2,8)	3612	3452	3590	3547	3137
(200,5,1.6,4)	988	949	989	976	1001	(600,5,1.6,4)	1494	1431	1499	1471	1408
(200,5,1.6,8)	2205	2073	2212	2198	2115	(600,5,1.6,8)	3434	3246	3457	3421	2962
(200,6,1.0,4)	1004	1033	1009	1007	1030	(600,6,1.0,4)	1649	1553	1622	1604	1506
(200,6,1.0,8)	2131	2157	2120	2114	2149	(600,6,1.0,8)	3468	3230	3425	3401	3018
(200,6,1.2,4)	958	980	952	946	981	(600,6,1.2,4)	1470	1392	1454	1434	1359
(200,6,1.2,8)	2084	2100	2080	2058	2076	(600,6,1.2,8)	3300	3086	3269	3237	2748
(200,6,1.6,4)	926	946	929	907	922	(600,6,1.6,4)	1360	1305	1373	1342	1239
(200,6,1.6,8)	2059	2078	2067	2041	1964	(600,6,1.6,8)	3173	2951	3187	3156	2548
(200,8,1.0,4)	992	1005	989	984	996	(600,8,1.0,4)	1470	1460	1458	1442	1361
(200,8,1.0,8)	2077	2083	2046	2044	2065	(600,8,1.0,8)	3088	3076	3066	3030	2780
(200,8,1.2,4)	940	949	927	907	931	(600,8,1.2,4)	1355	1320	1335	1302	1229
(200,8,1.2,8)	2028	2033	1994	1964	1940	(600,8,1.2,8)	2982	2926	2956	2908	2531
(200,8,1.6,4)	885	895	881	869	855	(600,8,1.6,4)	1247	1227	1258	1226	1134
(200,8,1.6,8)	1961	1950	1965	1934	1821	(600,8,1.6,8)	2857	2756	2866	2834	2372

volume of computations particularly for IPBMK. We implemented all methods in Matlab R2009b and obtained the runtime results for a single run on each test scenario as reported in Table 6 and Table 7 on a Pentium 4, CPU 3GHz, 1GB RAM. Note that we did not use parallel computing. Of course, runtime of IPBMK increases linearly in the number of iterations it requires, with BMK1 being the time that one iteration requires. DSPt clearly requires a little more time than DSP due to the more frequent prorations, but the difference is negligible. Averaged over all tested instances, IPBMK, BMK1, DSPt and DSP require 5.6, 1.2, 1.6 and 1.5 seconds respectively. The average number of iterations required by IPBMK is 4.6. We conclude that DSP achieves similar revenue levels as IPBMK (as we observed in Figure 1) but in less than a quarter of IPBMK's runtime.

The potential for improvement becomes even clearer when we look at the number of iterations that IPBMK required in the simulations as reported in Table 8, averaged over all data collection points and all 2000 simulations. DSP by definition only requires one iteration for all scenarios at approximately the same cost per iteration whilst achieving similar revenue levels.

Table 6: Run times in seconds and DP iterations for IPBMK

(T, ν, α, κ)	IPBMK (s)	Iter IPBMK	BMK1 (s)	DSPt (s)	DSP (s)
(200,4,1.0,4)	1.2	2	0.5	0.7	0.7
(200,4,1.0,8)	4.4	10	0.4	0.6	0.6
(200,4,1.2,4)	0.4	1	0.4	0.5	0.6
(200,4,1.2,8)	0.4	1	0.4	0.6	0.6
(200,4,1.6,4)	1.7	5	0.4	0.5	0.5
(200,4,1.6,8)	1.7	5	0.5	0.5	0.5
(200,5,1.0,4)	1.6	3	0.6	0.7	0.6
(200,5,1.0,8)	2.6	5	0.5	0.8	0.7
(200,5,1.2,4)	4.4	9	0.5	0.6	0.6
(200,5,1.2,8)	5.2	10	0.5	0.6	0.6
(200,5,1.6,4)	0.4	1	0.4	0.6	0.5
(200,5,1.6,8)	0.9	2	0.4	0.6	0.6
(200,6,1.0,4)	6.6	10	0.6	0.8	0.8
(200,6,1.0,8)	6.2	10	0.6	0.8	0.8
(200,6,1.2,4)	1.6	3	0.6	0.7	0.7
(200,6,1.2,8)	2.1	4	0.5	0.8	0.7
(200,6,1.6,4)	0.5	1	0.5	0.7	0.7
(200,6,1.6,8)	0.5	1	0.5	0.7	0.6
(200,8,1.0,4)	6.1	8	0.8	1.1	1.0
(200,8,1.0,8)	7.6	10	0.8	1.0	1.0
(200,8,1.2,4)	1.5	2	0.8	0.9	0.9
(200,8,1.2,8)	2.2	3	0.7	1.1	0.9
(200,8,1.6,4)	1.2	2	0.6	0.9	0.9
(200,8,1.6,8)	1.9	3	0.7	0.9	0.8

Table 7: Run times in seconds and DP iterations for IPBMK

(T, ν, α, κ)	IPBMK (s)	Iter IPBMK	BMK1 (s)	DSPt (s)	DSP (s)
(600,4,1.0,4)	12.2	7	1.7	2.2	2.1
(600,4,1.0,8)	12.1	7	1.8	2.2	2.2
(600,4,1.2,4)	1.6	1	1.5	2.0	1.9
(600,4,1.2,8)	1.5	1	1.6	2.0	1.9
(600,4,1.6,4)	6.9	5	1.3	1.8	1.7
(600,4,1.6,8)	6.4	5	1.3	2.0	1.6
(600,5,1.0,4)	4.0	2	2.1	2.7	2.5
(600,5,1.0,8)	4.1	2	2.2	2.7	2.5
(600,5,1.2,4)	16.0	9	1.8	2.4	2.2
(600,5,1.2,8)	17.9	10	1.9	2.3	2.2
(600,5,1.6,4)	1.7	1	1.5	2.0	1.9
(600,5,1.6,8)	3.0	2	1.6	2.1	2.0
(600,6,1.0,4)	12.2	6	2.0	2.6	2.5
(600,6,1.0,8)	17.5	9	2.0	2.6	2.5
(600,6,1.2,4)	5.4	3	1.9	2.4	2.3
(600,6,1.2,8)	8.7	5	1.9	2.4	2.3
(600,6,1.6,4)	1.6	1	1.6	2.1	2.0
(600,6,1.6,8)	7.5	5	1.6	2.1	2.0
(600,8,1.0,4)	17.7	7	2.8	3.3	3.3
(600,8,1.0,8)	25.8	10	2.7	3.4	3.4
(600,8,1.2,4)	4.7	2	2.7	3.2	3.1
(600,8,1.2,8)	7.0	3	2.7	3.3	3.1
(600,8,1.6,4)	4.1	2	2.3	2.9	2.8
(600,8,1.6,8)	6.2	3	2.3	2.9	2.8

Table 8: Average number of DP iterations for IPBMK over 2000 simulations with 20 re-solves each.

(T, ν, α, κ)	Iter	(T, ν, α, κ)	Iter
(200,4,1.0,4)	5.83	(600,4,1.0,4)	5.62
(200,4,1.0,8)	7.66	(600,4,1.0,8)	7.14
(200,4,1.2,4)	3.45	(600,4,1.2,4)	3.19
(200,4,1.2,8)	4.81	(600,4,1.2,8)	4.35
(200,4,1.6,4)	4.02	(600,4,1.6,4)	3.76
(200,4,1.6,8)	5.95	(600,4,1.6,8)	5.44
(200,5,1.0,4)	5.28	(600,5,1.0,4)	5.02
(200,5,1.0,8)	6.84	(600,5,1.0,8)	6.47
(200,5,1.2,4)	5.84	(600,5,1.2,4)	5.58
(200,5,1.2,8)	7.65	(600,5,1.2,8)	7.43
(200,5,1.6,4)	3.22	(600,5,1.6,4)	2.94
(200,5,1.6,8)	4.8	(600,5,1.6,8)	4.33
(200,6,1.0,4)	6.05	(600,6,1.0,4)	5.74
(200,6,1.0,8)	7.45	(600,6,1.0,8)	7.34
(200,6,1.2,4)	4.26	(600,6,1.2,4)	4.08
(200,6,1.2,8)	6.03	(600,6,1.2,8)	5.8
(200,6,1.6,4)	3.23	(600,6,1.6,4)	3.06
(200,6,1.6,8)	5.21	(600,6,1.6,8)	5.07
(200,8,1.0,4)	5.77	(600,8,1.0,4)	5.77
(200,8,1.0,8)	7.12	(600,8,1.0,8)	7.16
(200,8,1.2,4)	4.5	(600,8,1.2,4)	4.38
(200,8,1.2,8)	6.34	(600,8,1.2,8)	6.34
(200,8,1.6,4)	3.72	(600,8,1.6,4)	3.68
(200,8,1.6,8)	5.47	(600,8,1.6,8)	5.36

Conclusion

Network optimization plays a central role in today's state-of-the-art revenue management systems. One can tackle this problem by decomposing it into resource-level subproblems that can be solved efficiently, e.g. by dynamic programming (DP). Nevertheless, the solution of these subproblems is the most time-consuming part of the optimization and therefore critical. Recent research has focussed on accounting for the time-dependence of the marginal value of capacity in order to improve the expected revenues that policies resulting from this process can achieve. While this has resulted in very interesting methods, they tend to be computationally too complex for large-scale applications.

We propose a new dynamic fare proration method designed for this situation. It decomposes the network problem by fare proration and solves the resource-level dynamic programs simultaneously with simple, endogenously obtained dynamic marginal capacity value estimates. An extensive numerical study indicates that the estimates are very effective in that the new method achieves similar revenue levels as iterative DP fare proration, but at a fraction of the computational cost. Compared to using a standard fare proration approach based on static proration factors with a single DP iteration, the new method achieves on average significantly higher revenue levels at comparable computational cost, and is well-suited for large-scale practical applications.

For simplicity's sake, we did not consider customer choice models in this work. However, the concept of dynamic simultaneous fare proration can be applied in the same way to choice-based network revenue management. Since choice-based systems typically require more DP iterations than those under the independent demand assumption, the new method might be even more beneficial in such an environment. More research in this direction is clearly needed.

Acknowledgement—The second author gratefully acknowledges financial support by Lancaster University Management School and the LANCS Initiative.

Appendix

Proof of Proposition 2. The first inequality is due to Lemma 1. It remains to show the last inequality. Let us abbreviate summations over $i \in I$, $k \in I$ and $j \in J$ with only the subscript i, k and j, respectively. We denote the dual of (DLP) by (D):

$$\begin{aligned} z_{\mathrm{DLP}} = \min_{\tilde{\pi},\tilde{\sigma}} & \sum_i c_i \tilde{\pi}_i + \sum_j (\sum_{t=1}^{T} p_{jt}) \tilde{\sigma}_j \\ \text{(D)} \quad \text{s.t.} & \sum_i a_{ij} \tilde{\pi}_i + \tilde{\sigma}_j \geq f_j \qquad \forall j \in J \\ & \tilde{\pi}, \tilde{\sigma} \geq 0. \end{aligned}$$

Let us denote the optimal solution to (D) by (π, σ). We introduce the shorthand notation $\mathbf{I}_j$ for the indicator function $\mathbf{1}\{\sum_k a_{kj}\pi_k > 0\}$, and $\bar{\mathbf{I}}_j$ for $\mathbf{1}\{\sum_k a_{kj}\pi_k = 0\}$. With this notation, we can now decompose the (DLP) objective by the resource:

$$\begin{aligned} z_{\mathrm{DLP}} &= \sum_i c_i\pi_i + \sum_j (\sum_{t=1}^T p_{jt})\sigma_j \left[\frac{\sum_i a_{ij}\pi_i}{\sum_k a_{kj}\pi_k}\mathbf{I}_j + \frac{\sum_i a_{ij}}{\sum_k a_{kj}}\bar{\mathbf{I}}_j \right] \\ &= \sum_i \left[c_i\pi_i + \sum_j \sum_{t=1}^T p_{jt}\Big(\frac{\sigma_j a_{ij}\pi_i}{\sum_k a_{kj}\pi_k}\mathbf{I}_j + \frac{\sigma_j a_{ij}}{\sum_k a_{kj}}\bar{\mathbf{I}}_j \Big) \right] \\ &= \sum_i z_i(c_i, 1), \end{aligned}$$

where $z_i(x_i, t_0) := x_i\pi_i + \sum_j \sum_{t=t_0}^T p_{jt}\Big((\sigma_j a_{ij}\pi_i)/(\sum_k a_{kj}\pi_k)\mathbf{I}_j + (\sigma_j a_{ij})/(\sum_k a_{kj})\bar{\mathbf{I}}_j \Big)$.

We show by induction over time that $v_{t,i}^{\pi,1}(x_i) \leq z_i(x_i, t)$ holds for any resource i, for all time steps t and inventory levels x_i. Let us fix an arbitrary resource i and arbitrary x_i. The following inequalities hold for $t = T$:

$$\begin{aligned} v_{T,i}^{\pi,1}(x_i) &= \max_{S\subseteq J(x_{-i})} \sum_{j\in S} p_{jT}\Big[\frac{f_j\pi_i a_{ij}}{\sum_k a_{kj}\pi_k}\mathbf{I}_j + \frac{f_j a_{ij}}{\sum_k a_{kj}}\bar{\mathbf{I}}_j \Big] && (4) \\ &\leq \sum_{j|a_{ij}\leq x_i} p_{jT}\Big[\frac{(\sum_k a_{kj}\pi_k + \sigma_j)\pi_i a_{ij}}{\sum_k a_{kj}\pi_k}\mathbf{I}_j + \frac{(\sum_k a_{kj}\pi_k + \sigma_j)a_{ij}}{\sum_k a_{kj}}\bar{\mathbf{I}}_j \Big] && (5) \\ &= \sum_{j|a_{ij}\leq x_i} p_{jT}\pi_i a_{ij}\mathbf{I}_j + \sum_{j|a_{ij}\leq x_i} p_{jT}\Big[\frac{\sigma_j\pi_i a_{ij}}{\sum_k a_{kj}\pi_k}\mathbf{I}_j + \frac{\sigma_j a_{ij}}{\sum_k a_{kj}}\bar{\mathbf{I}}_j \Big] && (6) \\ &\leq z_i(x_i, T). && (7) \end{aligned}$$

The first equation (4) is due to the definition of $v_{T,i}^{\pi,1}$ in (2) and of the fare allocation f_j^i. Note that it is optimal to offer the set of all feasible products $\{j|a_{ij} \leq x_i\}$. The inequality (5) stems from the feasibility of the optimal solution (π, σ) to the dual of (DLP). Next, we rearrange and simplify terms to obtain (6). Finally, the last inequality (7) results from $a_{ij} \leq x_i$, $\sum_{j|a_{ij}\leq x_i} p_{jT}\mathbf{I}_j \leq 1$, $p_{jT}\Big((\sigma_j a_{ij}\pi_i)/(\sum_k a_{kj}\pi_k)\mathbf{I}_j + (\sigma_j a_{ij})/(\sum_k a_{kj})\bar{\mathbf{I}}_j\Big) \geq 0$ for all j, and the definition of $z_i(x_i, T)$.
Assume that $v_{t+1,i}^{\pi,1}(x_i) \leq z_i(x_i, t+1)$ holds for arbitrary fixed resource i. Then we have the following:

$$\begin{aligned} v_{t,i}^{\pi,1}(x_i) &\leq \max_{S\subseteq J(x_{-i})} \sum_{j\in S} p_{jt}\Big(\frac{f_j\pi_i a_{ij}}{\sum_k a_{kj}\pi_k}\mathbf{I}_j + \frac{f_j a_{ij}}{\sum_k a_{kj}}\bar{\mathbf{I}}_j + z_i(x_i - 1, t+1) \Big) + (1 - \sum_{j\in S} p_{jt}) z_i(x_i, t+1) && (8) \\ &= \max_{S\subseteq J(x_{-i})} \sum_{j\in S} p_{jt}\Big(\frac{f_j\pi_i a_{ij}}{\sum_k a_{kj}\pi_k}\mathbf{I}_j + \frac{f_j a_{ij}}{\sum_k a_{kj}}\bar{\mathbf{I}}_j - \pi_i \Big) + z_i(x_i, t+1) && (9) \\ &\leq \max_{S\subseteq J(x_{-i})} \sum_{j\in S} p_{jt}\Big(\pi_i a_{ij}\mathbf{I}_j + \frac{\sigma_j\pi_i a_{ij}}{\sum_k a_{kj}\pi_k}\mathbf{I}_j + \frac{\sigma_j a_{ij}}{\sum_k a_{kj}}\bar{\mathbf{I}}_j - \pi_i \Big) + z_i(x_i, t+1) && (10) \\ &\leq \sum_{j\in J} p_{jt}\Big(\frac{\sigma_j\pi_i a_{ij}}{\sum_k a_{kj}\pi_k}\mathbf{I}_j + \frac{\sigma_j a_{ij}}{\sum_k a_{kj}}\bar{\mathbf{I}}_j \Big) + z_i(x_i, t+1) && (11) \\ &= z_i(x_i, t). \end{aligned}$$

The first inequality (8) follows from the definition of $v_{t,i}^{\pi,1}(x_i)$ and from the induction assumption. Equality (9) holds because $z_i(x_i, t+1) - z_i(x_i - 1, t+1) = \pi_i$. As before, we exploit the dual feasibility of (π, σ) to obtain (10). Note that $a_{ij}\mathbf{I}_j \leq 1$ so that we can use $\pi_i a_{ij}\mathbf{I}_j - \pi_i \leq 0$, and since the remaining terms in (10) are all non-negative, we can sum over all products, resulting in inequality (11). The desired result follows. □

References

Adelman, D. (2007). Dynamic bid prices in revenue management. *Operations Research*, 55:647–661.

Belobaba, P. P. (1986). Application of a probabilistic decision model to airline seat inventory control. *Operations Research*, 37:183–197.

Belobaba, P. P. (1987). *Air travel demand and airline seat inventory management*. PhD thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge MA. `http://hdl.handle.net/1721.1/14800`.

Bratu, S. (1998). Network value concept in airline revenue management. Master's thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge MA. `http://hdl.handle.net/1721.1/9939`.

Chiang, W.-C., Chen, J. C. H., and Xu, X. (2007). An overview of research revenue management: current issues and future research. *International Journal of Revenue Management*, 1:97–128.

Farias, V. F. and Van Roy, B. (2007). An approximate dynamic programming approach to network revenue management. Working paper, Dept. of Electrical Engineering, Stanford University, Stanford CA. `http://www.stanford.edu/~bvr/psfiles/adp-rm.pdf`.

Glover, F., Glover, R., Lorenzo, J., and McMillan, C. (1982). The passenger mix problem in the scheduled airlines. *Interfaces*, 12(3):73–79.

Jiang, H. (2008). A Lagrangian relaxation approach for network inventory control of stochastic revenue management with perishable commodities. *Journal of the Operational Research Society*, 59:372–380.

Kunnumkal, S. and Topaloglu, H. (2010). A new dynamic programming decomposition method for the network revenue management problem with customer choice behavior. *Production and Operations Management*, 19:575–590.

McGill, J. and van Ryzin, G. (1999). Revenue management: Research overview and prospects. *Transportation Science*, 33:233–256.

Talluri, K. (2008). On bounds for network revenue management. Working paper, Faculty of Economic and Business Sciences, Universitat Pompeu Fabra, Barcelona, Spain. `http://ssrn.com/abstract=1107171`.

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

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

Topaloglu, H. (2009). Using Lagrangian relaxation to compute capacity-dependent bid prices in network revenue management. *Operations Research*, 57:637–649.

Topaloglu, H. (2011). `http://people.orie.cornell.edu/~huseyin/research/rm_datasets/rm_datasets.html`.

Williamson, E. (1992). *Airline network seat control*. PhD thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA. `http://hdl.handle.net/1721.1/13174`.

Zhang, D. (2011). An improved dynamic programming decomposition approach for network revenue management. *Manufacturing & Service Operations Management*, 13:35–52.