Expected Future Value Decomposition Based Bid Price Generation for Large-Scale Network Revenue Management[1]

L.F. Escudero[2], J.F. Monge[3], D. Romero Morales[4], J. Wang[4]

[2] Departamento de Estadística e Investigación Operativa

Universidad Rey Juan Carlos, Madrid, Spain

e-mail: laureano.escudero@urjc.es

[3] Centro de Investigación Operativa

Universidad Miguel Hernández, Elche (Alicante), Spain

e-mail: monge@umh.es

[4] Saïd Business School

University of Oxford, Oxford, United Kingdom

e-mail: {dolores.romero-morales,jingbo.wang}@sbs.ox.ac.uk

Abstract

This paper studies a multi-stage stochastic programming model for large-scale network revenue management. We solve the model by means of the so-called Expected Future Value (EFV) decomposition via scenario analysis, estimating the impact of the decisions made at a given stage on the objective function value related to the future stages. The EFV curves are used to define bid prices on bundles of resources directly, as opposed to the traditional additive bid prices. Numerical results show that the revenue outcome of our approach is generally comparable to that of state-of-the-art additive approaches, and tends to be better when the network structure is complex. Moreover, our approach requires significantly less computation time than the direct optimization, taking only up to 5 minutes for large-scale problem instances (up to 2.6 million variables and 2.3 million constraints).

Keywords: network revenue management, scenario trees, stochastic dynamic programming, expected future value curves, non-additive bid prices.

[1]This research has been partially supported by the grants URJC-CM-2008-CET-3703 from Comunidad de Madrid and MTM2009-14087-C04-01 from the Ministry of Science and Innovation, Spain

1 Introduction

Revenue management aims to maximize the revenue of selling limited quantities of a set of resources by means of demand management decisions. A resource in revenue management is usually a perishable product/service, such as seats on a single flight leg or hotel rooms for a given date. It is common in revenue management that multiple resources are sold in "bundles". For instance, connecting flight legs are sold on a single ticket and hotel customers may stay multiple nights. In this case, the lack of availability of any resource will prevent sales of the bundle, which creates interdependence among these resources. Consequently, the demand management decisions of these resources must be coordinated, which is generally referred to as the *network revenue management problem* [17].

The state-of-the-art approach to this problem is to use bid-price policies [16], in which a bid price is generated for each bundle, and a request to purchase the bundle is accepted if and only if the associated revenue exceeds the bid price. Bid-price policies are not optimal, in general, but they are very popular because they are intuitive and easy to implement, see [16, 17]. Bid prices can be generated using either an "additive" or a "non-additive" approach [1]. In the additive approach, a bid price is generated for each resource, and the bid price of a bundle is defined as the sum of the bid prices of all resources used by the bundle. On the other hand, in the non-additive approach a bid price is generated directly for each bundle.

A number of optimization models have been proposed for generating additive bid prices, mainly differing in the way uncertainty in demand is incorporated. The simplest and most popular model is the so-called Deterministic Linear Programming (DLP) due to D'Sylva [6], Glover et *al.* [7], and Wong, Koppelman and Daskin [19]. The demand for each bundle, which is stochastic by nature, is replaced by the mean value. The objective of the optimization model is to maximize the revenue such that sales are bounded by the mean demand and the capacities on the resources are not violated. The bid prices of the resources are calculated as the dual variables of the capacity constraints [7, 19]. Talluri and van Ryzin [16] showed that DLP is asymptotically optimal as capacity and demand increase linearly and with the same rate. The main drawback of DLP is that it considers only the mean demand and ignores all the uncertainty. As a result, it could suffer when the variance of demand is high.

A natural extension of DLP is to replace the mean demand by the demand distribution, which is called the Probabilistic Non-Linear Programming (PNLP) model. A Linear Programming (LP) version of PNLP was proposed by Wollmer in [18] by using discrete scenarios to represent the demand distribution. In the same manner as in DLP, the bid prices of the resources are calculated as the dual variables of the capacity constraints. Higle and Sen [8] converted the PNLP into a Stochastic Programming (SP) simple recourse model.

The Randomized Linear Programming (RLP) method was first proposed by Smith and Penn [14], and was then further investigated by Talluri and van Ryzin [16]. Basically, RLP considers a finite set of demand scenarios with their corresponding weights, and then solves a different DLP for each scenario. The bid prices of the resources are calculated as the weighted average of the Lagrange multipliers corresponding to the DLPs using scenario demand rather than mean demand. Topaloglu [13] has recently showed that RLP is asymptotically optimal under the abovementioned conditions for DLP.

PNLP and RLP consider different scenarios for the total demand over the whole booking horizon and ignore the potential inter-temporal uncertainty of demand along the booking horizon. In order to account for these dynamics, Higle and Sen [8] studied a two-period SP model, in which the initial allocation of resources is revised based on the demand observed in the first period. Numerical results were provided for small and medium networks. Chen and Hommem-de-Mello [4] also studied a two-period SP model and proposed to handle the multi-period problem by solving a sequence of two-period models. Möller, Römisch and Weber [10, 11] were the first to study a multi-period SP model, however their approach was only tested on single-resource instances. DeMiguel and Mishra [5] studied another multi-period SP model, where protection levels are not modeled. They focus on examining different methods for generating the scenarios trees, and numerical results are provided for small and medium sized networks.

Additive bid prices are easy to implement and popular in practice, however non-additive bid prices could provide a more accurate reflection of the opportunity cost of bundles. As argued in [15], the opportunity cost of a bundle is basically determined by the most constraining resource used by the bundle, and therefore additive bid prices could be restrictive compared to non-additive ones. Bertsimas and Popescu [1] propose a framework for generating non-additive bid prices. Given a model for network revenue management, the bid price of a bundle is calculated as the change on the expected revenue when the capacity on each resource used by the bundle is reduced by one unit. However this approach can be computationally expensive, since a different optimization problem needs to be solved for each bundle.

In this study, we propose an SP based method for generating non-additive bid prices. Noticing that the high computational complexity of multi-period SP models has prevented their application in large networks, we use the Expected Future Value (EFV) decomposition algorithm proposed in [3]. Adopting a Stochastic Dynamic Programming (SDP) approach [12], the EFV decomposition algorithm first combines time periods into stages, then divides the SP problem into intra-stage subproblems that are linked to each other, and finally solves them iteratively. The crucial ingredient of the algorithm is the family of so-called EFV curves, estimating the impact of the decisions to be made at a given stage on the objective function value related to the future stages. The EFV curves are used to define non-additive bid prices on bundles directly.

We apply the EFV decomposition algorithm to the two multi-period SP models proposed in [11] and [5] respectively. Numerical results show that the EFV decomposition algorithm requires significantly less computation time than the direct optimization of the corresponding SP model, taking only up to 5 minutes for large-scale problem instances (up to 2.6 million variables and 2.3 million constraints). The revenue performance of the non-additive bid prices from EFV is tested in a rolling horizon simulation in two test networks. Results show that, in general, the non-additive bid prices from EFV lead to comparable revenues to those from additive SP models. Moreover, when bundles containing a large number of resources are present in the network, the non-additive bid prices from EFV can improve the revenue of additive SP models.

The remainder of the paper is organized as follows. Section 2 introduces the general SP formulation and reviews the EFV decomposition algorithm. The network revenue management problem and the two multi-period SP models are introduced in Section 3. In Section 4, we explain how the EFV decomposition algorithm can be used to generate non-additive bid prices. Section 5 is devoted to the computational experience, in which we examine the solution accuracy, computation time and revenue performance of our approach. Finally, we conclude the paper and discuss future research directions in Section 6.

2 The EFV decomposition algorithm

In this section, we familiarize the reader with the EFV decomposition algorithm for solving stochastic programs proposed in [3]. Section 2.1 presents the general SP formulation. The algorithmic framework of the EFV decomposition algorithm is introduced in Section 2.2. The crucial ingredient of the algorithm, i.e., the EFV curves, is explained in detail in Section 2.3.

2.1 The general SP formulation

Consider the following dynamic multi-linking constraint deterministic program, in which decisions are taken over a time horizon $\mathcal{T}$ with T periods:

$$\text{maximize} \sum_{t\in\mathcal{T}} c_t x_t \tag{2.1}$$

s.t.

$$\sum_{t\in\{\tau-1,\tau\}} A^t_\tau x_t = b_\tau \qquad \forall \tau \in \mathcal{T} \tag{2.2}$$

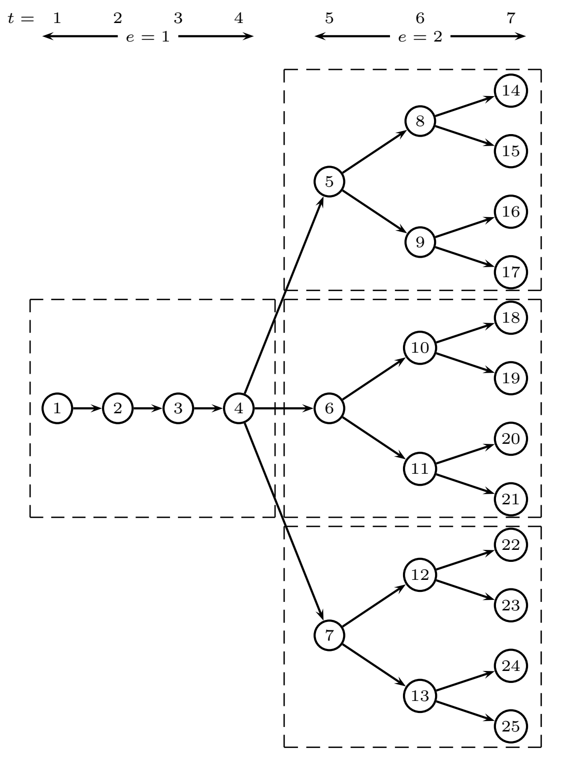
$$x_t \in \mathcal{X}_t \qquad \forall t \in \mathcal{T}, \tag{2.3}$$

where x_t is the vector of variables related to time period t, c_t is the row vector of the objective function coefficients associated with x_t, A^t_τ is the coefficient matrix in the constraints related to time period τ for x_t, and b_τ is the right-hand-side (*rhs*) vector for the constraint related to time period τ, for $\tau \in \mathcal{T}$. Additional constraints on x_t, such as non-negativity and 0–1 constraints, are modeled in $\mathcal{X}_t$. All vectors and matrices have the appropriate dimensions. Hereafter, components of x_t that have non-zero coefficients in A^t_{t+1}, will be referred as "linking" variables, since they will affect the decisions in period $t+1$. Note that in this formulation, and for the sake of clarity, variables in a given period will affect the ones in the next period, but not further periods into the future. As we will see in Section 4, this assumption is satisfied by the network revenue management problem.

The stochasticity of the objective function and the *rhs* vectors is modeled by means of a scenario tree, such as the one illustrated in Figure 1. Each node in the tree represents a point in time where a decision will be made. Once a decision is made, some contingencies can arise and information related to these contingencies is available at the beginning of the period. Each root-to-leaf path in the tree represents one specific scenario and corresponds to one realization of the whole set of uncertain parameters. Each node in the tree can be associated with a scenario group, such that two scenarios belong to the same group from a given time period provided that they have the same realizations of the uncertain parameters up to that period. Accordingly with the non-anticipativity principle, for example, see [2], scenarios belonging to the same group at a given time period t should have the same value for variables x_τ with $\tau \leq t$, for all $t \in \mathcal{T}$.

In the following we introduce the notation for describing the elements of the scenario tree:

Ω, set of scenarios, consecutively numbered.



$\mathcal{T} = \{1, \dots, 7\}$

$\Omega = \Omega_1 = \{14, 15, \dots, 25\}$

Scenario 14 = path $1, 2, \dots, 5, 8, 14$

$t(12) = 6$; $\rho(12) = 7$

$\mathcal{G}^2 = \{5, \dots, 25\}$

$\mathcal{A}^2 = \{5, 6, 7\}$

$\mathcal{C}_5 = \{5, 8, 9, 14, \dots, 17\}$

$\mathcal{S}_3 = \{3, 4, 5, 6, 7, \dots, 24, 25\}$

$\mathcal{P}_1 = \{4\}$

Figure 1: An example of a scenario tree.

$\mathcal{G}$, set of scenario groups, consecutively numbered.

w^g, likelihood associated with scenario group g, for $g \in \mathcal{G}$.

$t(g)$, time period for scenario group g, for $g \in \mathcal{G}$.

Ω_g, set of scenarios in group g, such that the scenarios that belong to the same group are identical in all realizations of the uncertain parameters up to period $t(g)$, for $g \in \mathcal{G}$. Notice that $\Omega_g \subseteq \Omega$. From now on we will without distinction use nodes in the scenario tree and scenario groups.

$\rho(g)$, immediate ancestor node of node g, for $g \in \mathcal{G}$. (A dummy node acts as the ancestor of the root node.)

$\mathcal{S}_g$, set of nodes in the subtree whose root is node g, for $g \in \mathcal{G}$.

In order to present the stochastic version of Model (2.1)-(2.3), let the following notation be used for the variables and the uncertain parameters.

Variables:

x^g, vector of variables under scenario group g, for $g \in \mathcal{G}$. It replaces the vector $x_{t(g)}$ in the deterministic model.

Uncertain parameters:

c^g, vector of the objective function coefficients for the variables x^g under scenario group g, for $g \in \mathcal{G}$. It replaces the vector $c_{t(g)}$ in the deterministic model.

b^g, *rhs* of the constraints under scenario group g, for $g \in \mathcal{G}$. It replaces the vector $b_{t(g)}$ in the deterministic model.

The *Deterministic Equivalent Model (DEM)* of the stochastic program with complete recourse for maximizing the expected objective function value over the set of scenarios has the following so-called *compact* representation as an alternative to Model (2.1)-(2.3),

$$\text{maximize} \sum_{g \in \mathcal{G}} w^g c^g x^g \tag{2.4}$$

s.t.

$$\sum_{\delta \in \{\rho(g), g\}} A_{t(g)}^{t(\delta)} x^\delta = b^g \qquad \forall g \in \mathcal{G} \tag{2.5}$$

$$x^g \in \mathcal{X}_{t(g)} \qquad \forall g \in \mathcal{G}. \tag{2.6}$$

2.2 The EFV decomposition algorithm

The EFV decomposition algorithm adopts a stochastic dynamic programming approach [12]. It aims to approximate the solution value of DEM through an iterative procedure, which stops if the relative change of solution value between two consecutive iterations is below a tolerance parameter $\epsilon > 0$.

The EFV decomposition algorithm combines consecutive time periods into stages, as shown in Figure 1. The following notation describing the stages will be used throughout the paper:

$\mathcal{E}$, set of stages in the time horizon.

$\mathcal{G}^e$, set of scenario groups from stage e, for $e \in \mathcal{E}$.

$\mathcal{A}^e$, set of scenario groups associated with the root nodes from stage e, for $e \in \mathcal{E}$. Notice that $\mathcal{A}^e \subseteq \mathcal{G}^e$.

$\mathcal{C}_a$, set of nodes in the subtree rooted at node a with nodes in $\mathcal{G}^e$, for $a \in \mathcal{A}^e, e \in \mathcal{E}$.

$\mathcal{P}_a$, set of leaf nodes in $\mathcal{C}_a$, for $a \in \mathcal{A}^e$.

Once the time horizon has been split into stages, the DEM can be divided into subproblems, which are connected by the linking variables. For each $e \in \mathcal{E}$ and $a \in \mathcal{A}^e$, we associate a subproblem with the subtree defined by node set $\mathcal{C}_a$. In Figure 1, $\mathcal{C}_5 = \{5, 8, 9, 14, \ldots, 17\}$ defines a subtree/subproblem in stage 2 with node 5 as the root node. In this example, there are in total four subtrees/subproblems marked by dashed boxes.

The subproblem defined by node set $\mathcal{C}_a$, $a \in \mathcal{A}^e$, $e \in \mathcal{E}$, can be written as:

$$\sigma^a := \text{maximize} \sum_{g \in \mathcal{C}_a} w^g c^g x^g + \sum_{g \in \mathcal{P}_a} \lambda^g(x^g) \tag{2.7}$$

s.t.

$$\sum_{\delta \in \{\rho(g), g\}} A_{t(g)}^{t(\delta)} x^\delta = b^g \qquad \forall g \in \mathcal{C}_a \tag{2.8}$$

$$x^g \in \mathcal{X}_{t(g)} \qquad \forall g \in \mathcal{C}_a \tag{2.9}$$

$$x^{\rho(a)} = \overline{x}^{\rho(a)} \tag{2.10}$$

where $\overline{x}^{\rho(a)}$ is the vector of variables the subproblem receives from its ancestor node $\rho(a)$, in which only the linking components will be actually used in the subproblem, and $\lambda^g(x^g)$ represents the expected future objective function value under the set of scenarios Ω_g, for each leaf node $g \in \mathcal{P}_a$. The function $\lambda^g(x^g)$ is therefore called the "Expected Future Value" (EFV) curve of node g. Since the exact EFV curves are generally difficult to compute, the EFV decomposition algorithm proposes to approximate them by piecewise linear and concave functions.

In short, each iteration of the EFV decomposition algorithm consists of a *front-to-back* scheme, followed by a *back-to-front* scheme. The front-to-back scheme solves subproblems from stage 1 to stage $|\mathcal{E}|$, passing the obtained values of linking variables onto the subproblems in the next stage. The back-to-front scheme goes from stage $|\mathcal{E}|$ to stage 1. In each stage, it first refines the approximations of the EFV curves and then solves the associated subproblems based on those refined approximations.

2.3 Approximating the EFV curves

In this section, we will explain how the approximations of the EFV curves are obtained and refined in the back-to-front scheme.

Consider a leaf node g in stage $e, g \in \mathcal{P}_a, a \in \mathcal{A}^e, e \in \mathcal{E}$. The descendent subproblems of node g in stage $e+1$ are given by node sets $\mathcal{C}_a$, $\forall a \in \mathcal{S}_g \cap \mathcal{A}^{e+1}$. We can express $\lambda^g(x^g)$ as the sum of the optimal objective function value of these subproblems:

$$\lambda^g(x^g) = \sum_{a \in \mathcal{S}_g \cap \mathcal{A}^{e+1}} w^a \sigma^a. \tag{2.11}$$

Let $\overline{x}^g$ be the values of x^g obtained in the previous front-to-back scheme. We obtain an approximation of the EFV curve $\lambda^g(x^g)$ through an ad-hoc sensitivity analysis of σ^a against x^g around the values of $\overline{x}^g$. These values of x^g around $\overline{x}^g$ are called *reference levels.*

Let $\mathcal{Z}^g$ denote the set of the reference levels for x^g. For each reference level $z \in \mathcal{Z}^g$, we solve the subproblem defined by $\mathcal{C}_a$ by fixing the values of variables of x^g at z. Let σ^{az} be the solution value of the subproblem and π^{az} be the dual variables of constraints (2.10). Due to the concavity of this subproblem, we have:

$$\sigma^a \quad \leq \quad \sigma^{az} + \pi^{az}(x^g - z), \tag{2.12}$$

and together with Formula (2.11), we obtain:

$$\lambda^g(x^g) \quad \leq \quad \mu^{gz}_{e+1} + \pi^{gz}_{e+1} x^g, \tag{2.13}$$

where $\mu^{gz}_{e+1} := \sum_{a \in \mathcal{S}_g \cap \mathcal{A}^{e+1}} w^a(\sigma^{az} - \pi^{az} z)$ and $\pi^{gz}_{e+1} := \sum_{a \in \mathcal{S}_g \cap \mathcal{A}^{e+1}} w^a \pi^{az}$ are constant terms. Combining Inequality (2.13) obtained for multiple reference levels will give us an upper envelope of a finite family of linear functions, continuous and concave but not differentiable everywhere, see Figure 2 for an example.

We use this envelope as an approximation of the EFV curve $\lambda^g(x^g)$, which means that constraints (2.13) will be added to subproblem (2.7)-(2.10), for all reference levels $z \in \mathcal{Z}^g$, where $\lambda^g(x^g)$ will be replaced by λ^g. Note that since the non-linking components of x^g do not affect the objective function value of future subproblems, they have zero coefficient in π^{az}, and subsequently zero coefficient in π^{gz}_{e+1}.

The approximations of the EFV curves are refined in each back-to-front scheme by generating new reference levels.

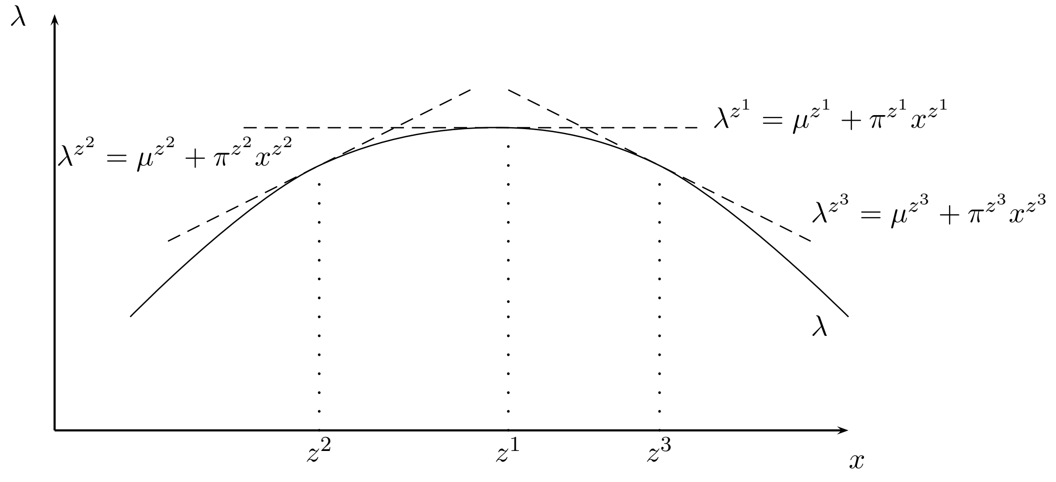

λ
$\lambda^{z^1} = \mu^{z^1} + \pi^{z^1} x^{z^1}$
$\lambda^{z^2} = \mu^{z^2} + \pi^{z^2} x^{z^2}$
$\lambda^{z^3} = \mu^{z^3} + \pi^{z^3} x^{z^3}$
λ
z^2
z^1
z^3
x

Figure 2: An example of an EFV curve and its approximation.

3 The network revenue management problem

As discussed in the introduction, two multi-period SP models have been proposed for network revenue management in [11] and [5] respectively. In the following we give the formulations of both models.

The following notation is used to describe the network revenue management problem:

Sets:

$\mathcal{L}$, set of resources (with size L).

$\mathcal{I}$, set of bundles (with size I).

$\mathcal{J}$, set of fare classes (with size J).

$\mathcal{I}_l$, set of bundles using resource l, for $l \in \mathcal{L}$.

Deterministic parameters:

f_{ij}, fare of bundle-class ij, for $i \in \mathcal{I}$, $j \in \mathcal{J}$.

C_l, capacity on resource l, for $l \in \mathcal{L}$.

Uncertain parameters:

d_{ij}^g, demand for bundle-class ij in period $t(g)$ at node g, for $i \in \mathcal{I}$, $j \in \mathcal{J}$, $g \in \mathcal{G}$.

Variables:

b_{ij}^{g}, number of accepted bookings for bundle-class ij in period $t(g)$ at node g, for $i \in \mathcal{I}$, $j \in \mathcal{J}$, $g \in \mathcal{G}$.

B_{ij}^{g}, cumulative number of accepted bookings of bundle-class ij along the path from the root to node g, for $i \in \mathcal{I}$, $j \in \mathcal{J}$, $g \in \mathcal{G}$.

$P_{ij}^{\rho(g)}$, protection level of bundle-class ij set at node $\rho(g)$ for cumulative accepted bookings along the path from the root to node g, for $i \in \mathcal{I}$, $j \in \mathcal{J}$, $g \in \mathcal{G}$. (Notice that all the nodes with the same immediate ancestor share the same protection level.)

3.1 The model with Protection Levels

A stochastic programming model with protection levels and satisfying the non-anticipativity constraints was proposed in [10] and [11]. The DEM formulation of the model is the following:

$$\text{maximize} \sum_{g \in \mathcal{G}} w^g \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} f_{ij} b_{ij}^{g} \tag{3.1}$$

s.t.

$$B_{ij}^{g} = B_{ij}^{\rho(g)} + b_{ij}^{g} \qquad \forall g \in \mathcal{G}, i \in \mathcal{I}, j \in \mathcal{J} \tag{3.2}$$

$$B_{ij}^{g} \le P_{ij}^{\rho(g)} \qquad \forall g \in \mathcal{G}, i \in \mathcal{I}, j \in \mathcal{J} \tag{3.3}$$

$$\sum_{i \in \mathcal{I}_l} \sum_{j \in \mathcal{J}} P_{ij}^{\rho(g)} \le C_l \qquad \forall g \in \mathcal{G}_T, l \in \mathcal{L} \tag{3.4}$$

$$0 \le b_{ij}^{g} \le d_{ij}^{g} \qquad \forall g \in \mathcal{G}, i \in \mathcal{I}, j \in \mathcal{J}, \tag{3.5}$$

where $\mathcal{G}_T$ is the set of nodes in the last period of the time horizon, T. Constraints (3.2) define the booking balance equations and constraints (3.3) ensure that the cumulative number of accepted bookings along the path from the root to node g cannot exceed the protection level set at the ancestor node $\rho(g)$. The protection levels across bundles and class fares are then bounded by the capacity on the resources in constraints (3.4). Constraints (3.5) reflect that the number of accepted bookings should be not greater than the demand. Notice that the non-anticipativity constraints are satisfied by construction. We will refer to this model as DEM$^{\mathrm{P}}$.

Note that in DEM$^{\mathrm{P}}$, the only linking variables are B_{ij}^{g}, since they determine the remaining capacity on the resources. It is easy to see that B_{ij}^{g} is only present in constraints associated with its own period, $t(g)$, and the next one.

3.2 The model without Protection Levels

A model partially violating the non-anticipativity principle was proposed in [5]. The DEM formulation of the model is the following:

$$\text{maximize} \sum_{g\in\mathcal{G}} w^g \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} f_{ij} b_{ij}^g \tag{3.6}$$

s.t.

$$B_{ij}^g = B_{ij}^{\rho(g)} + b_{ij}^g \qquad \forall g\in\mathcal{G}, i\in\mathcal{I}, j\in\mathcal{J} \tag{3.7}$$

$$\sum_{i\in\mathcal{I}_l} \sum_{j\in\mathcal{J}} B_{ij}^g \le C_l \qquad \forall g\in\mathcal{G}_T, l\in\mathcal{L} \tag{3.8}$$

$$0 \le b_{ij}^g \le d_{ij}^g \qquad \forall g\in\mathcal{G}, i\in\mathcal{I}, j\in\mathcal{J}. \tag{3.9}$$

We will refer to this model as DEM$^{\mathrm{N}}$. Notice that here constraints (3.8), imposing that the total number of accepted bookings over the whole booking horizon is restricted by the capacity on the resources, replace constraints (3.3) and (3.4) in DEM$^{\mathrm{P}}$. As a consequence, DEM$^{\mathrm{N}}$ has fewer variables and constraints than DEM$^{\mathrm{P}}$.

Contrary to DEM$^{\mathrm{P}}$, in which the same protection level is chosen for all successor nodes, in DEM$^{\mathrm{N}}$, different allocations can be chosen for different successors. In other words, DEM$^{\mathrm{N}}$ assumes perfect information on which realization of demand will happen in the next period.

4 EFV based non-additive bid prices

In this section, we propose to use the EFV decomposition algorithm introduced in Section 2 to solve both DEM$^{\mathrm{P}}$ and DEM$^{\mathrm{N}}$, as well as derive non-additive bid prices. We denote the two resulting algorithms by EFV$^{\mathrm{P}}$ and EFV$^{\mathrm{N}}$ respectively.

In the last back-to-front step of EFV$^{\mathrm{P}}$ and EFV$^{\mathrm{N}}$, we solve the unique subproblem in stage 1 and derive the non-additive bid prices. In the following, we write down this subproblem for both cases.

Let us first look at the subproblem in stage 1 in algorithm EFV$^{\mathrm{P}}$. We have that

$$\text{maximize} \sum_{g\in\mathcal{G}^1} w^g \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} f_{ij} b_{ij}^g + \sum_{g\in\mathcal{P}_1} \lambda^g \tag{4.1}$$

s.t.

$$B_{ij}^{g} = B_{ij}^{\rho(g)} + b_{ij}^{g} \qquad \forall g \in \mathcal{G}^1, i \in \mathcal{I}, j \in \mathcal{J} \tag{4.2}$$

$$B_{ij}^{g} \leq P_{ij}^{\rho(g)} \qquad \forall g \in \mathcal{G}^1, i \in \mathcal{I}, j \in \mathcal{J} \tag{4.3}$$

$$\sum_{i \in \mathcal{I}_l} \sum_{j \in \mathcal{J}} P_{ij}^{\rho(g)} \leq C_l \qquad \forall g \in \mathcal{P}_1, l \in \mathcal{L} \tag{4.4}$$

$$0 \leq b_{ij}^{g} \leq d_{ij}^{g} \qquad \forall g \in \mathcal{G}^1, i \in \mathcal{I}, j \in \mathcal{J} \tag{4.5}$$

$$\lambda^g \leq \mu_2^{gz} + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \pi_{2,ij}^{gz} B_{ij}^{g} \qquad \forall g \in \mathcal{P}_1, z \in \mathcal{Z}^g. \tag{4.6}$$

Hereafter, we will refer to constraints (4.6) as the EFV curve constraints of leaf nodes in stage 1.

Similarly, in algorithm EFV$^{\text{N}}$, the subproblem in stage 1 reads as follows

$$\text{maximize} \sum_{g \in \mathcal{G}^1} w^g \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} f_{ij} b_{ij}^{g} + \sum_{g \in \mathcal{P}_1} \lambda^g \tag{4.7}$$

s.t.

$$B_{ij}^{g} = B_{ij}^{\rho(g)} + b_{ij}^{g} \qquad \forall g \in \mathcal{G}^1, i \in \mathcal{I}, j \in \mathcal{J} \tag{4.8}$$

$$\sum_{i \in \mathcal{I}_l} \sum_{j \in \mathcal{J}} B_{ij}^{g} \leq C_l \qquad \forall g \in \mathcal{P}_1, l \in \mathcal{L} \tag{4.9}$$

$$0 \leq b_{ij}^{g} \leq d_{ij}^{g} \qquad \forall g \in \mathcal{G}^1, i \in \mathcal{I}, j \in \mathcal{J} \tag{4.10}$$

$$\lambda^g \leq \mu_2^{gz} + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \pi_{2,ij}^{gz} B_{ij}^{g} \qquad \forall g \in \mathcal{P}_1, z \in \mathcal{Z}^g. \tag{4.11}$$

Similarly, we will use the EFV curve constraints term when referring to (4.11).

Before the bid prices are defined, there are two useful observations to be made. For a given EFV curve constraint, the coefficient $\pi_{2,ij}^{gz}$ of variable B_{ij}^{g} can be seen as an estimation of the future effect in revenue of a unit increase of the cumulated bookings B_{ij}^{g}. Let $\Gamma^{gz} \geq 0$ be the dual variable of the EFV curve constraint of $g \in \mathcal{P}_1, z \in \mathcal{Z}^g$. Using the Duality Theory, we know that $\sum_{z \in \mathcal{Z}^g} \Gamma^{gz} = 1$, while Γ^{gz} will be equal to zero when the EFV curve constraint is not binding. Therefore, for a given g, Γ^{gz} can be seen as a measure of the importance of the binding EFV curve constraints.

In both formulations, once the corresponding stage 1 subproblem is solved, the bid price for bundle i is defined as:

$$\max \Big\{ \sum_{g \in \mathcal{P}_1} \sum_{z \in \mathcal{Z}^g} \Gamma^{gz} \pi_{2,ij}^{gz} \; : \; j \in \mathcal{J} \Big\}, \tag{4.12}$$

where we combine the slopes $\pi_{2,ij}^{gz}$ given by all the EFV curve constraints of leaf nodes in stage 1, using the dual variables Γ^{gz}.

Using the definition of $\pi^{gz}_{2,ij}$ given in Section 2.3, $\pi^{gz}_{2,ij} := \sum_{a \in \mathcal{S}_g \cap \mathcal{A}^{e+1}} w^a \pi^{az}_{ij}$, the term being maximized in (4.12) can be rewritten as

$$\begin{aligned}
\sum_{g \in \mathcal{P}_1} \sum_{z \in \mathcal{Z}^g} \Gamma^{gz} \pi^{gz}_{2,ij} &= \sum_{g \in \mathcal{P}_1} \sum_{z \in \mathcal{Z}^g} \Gamma^{gz} \sum_{a \in \mathcal{S}_g \cap \mathcal{A}^{e+1}} w^a \pi^{az}_{ij} \\
&= \sum_{g \in \mathcal{P}_1} \sum_{a \in \mathcal{S}_g \cap \mathcal{A}^{e+1}} w^a \sum_{z \in \mathcal{Z}^g} \Gamma^{gz} \pi^{az}_{ij} \\
&= \sum_{g \in \mathcal{P}_1} \sum_{a \in \mathcal{S}_g \cap \mathcal{A}^{e+1}} \sum_{z \in \mathcal{Z}^g} w^a \, \Gamma^{gz} \pi^{az}_{ij},
\end{aligned}$$

which is a weighted average of the slopes π^{az}_{ij}, since $\sum_{g \in \mathcal{P}_1} \sum_{a \in \mathcal{S}_g \cap \mathcal{A}^{e+1}} \sum_{z \in \mathcal{Z}^g} w^a \, \Gamma^{gz} = 1$.

Finally, the bid prices derived from DEM are additive, while with our approach the bid prices are calculated directly for each bundle, and therefore they are not additive, in general.

5 Computational experience

5.1 Overview

Our computational experience consists of two parts. The first part focuses on the performance of EFV as a decomposition algorithm, which includes testing its solution accuracy and computation time compared to DEM. The second part focuses on the revenue performance of the non-additive bid prices generated by EFV, which consists of running a rolling horizon simulation [5, 15] and comparing the revenues generated by EFV with DEM and state-of-the-art additive approaches.

Recall that the EFV decomposition algorithm will stop if the relative change of solution value between two consecutive iterations is below a tolerance parameter $\epsilon > 0$. Throughout our numerical experiments, we set $\epsilon = 0.1\%$. We use the optimization engine CPLEX v11.0 [9] for solving the LP problems arising. Our experiments were conducted on a PC with a 2.33 GHz Intel Xeon dual core processor, 8.5 Gb of RAM and the operating system was LINUX Debian 4.0.

The remainder of the section is organized as follows. We will describe the test networks and the demand model in Section 5.2. The results on solution accuracy and computation time will be given in Section 5.3. Finally, we present the results on revenue performance in Section 5.4.

5.2 The test networks and demand model

In this numerical study, we consider a medium network and a large one. Next, we will introduce first the dimensions and then the structure of our test networks. Finally, we will discuss the fare structure and the demand model.

Recall that the dimensions of a network are given by its number of bundles I, number of fare classes J and number of resources L. We also include here the capacity on the resources C_l. The dimensions of our test networks can be found below:

Medium network: $I = 18$, $J = 2$, $L = 10$, $C_l = 200\ \forall l \in \mathcal{L}$.

Large network: $I = 200$, $J = 2$, $L = 100$, $C_l = 200\ \forall l \in \mathcal{L}$.

In terms of network structure, we follow the standard practice in the literature to use randomly generated networks with a hub-and-spoke structure [1, 5, 15], resembling networks seen in hub-based airlines.

Among the resources, the first half are spoke-to-hub flight legs and the second half are hub-to-spoke flight legs. The bundles are generated as follows. In the medium network, the first 10 bundles each use one of the 10 legs, and the remaining 8 each use two random legs, spoke1-to-hub and hub-to-spoke2. In the large network, the first 100 bundles each use one of the 100 legs, the next 75 each use two random legs, spoke1-to-hub and hub-to-spoke2, and finally the last 25 each use four random legs spoke1-to-hub, hub-to-spoke2, spoke2-to-hub and hub-to-spoke1.

We now present the way the fares and the demands have been generated. In the following, we will denote by $N^{tr}(\mu, \sigma, a, b)$ a normal distribution with parameters μ and σ, and truncated within the interval $[a, b]$.

The class 1 fare of a single-resource bundle is generated from $N^{tr}(100, 40, 20, 180)$. The class 1 fare of a multi-resource bundle is the summation of the class 1 fares of the single-resource bundles associated with its resources. For all bundles, the fare of class 2 is twice that of class 1.

The demand for bundle-class ij in period t is generated from $N^{tr}(\mu_{ijt}, \upsilon\mu_{ijt}, 0, 2\mu_{ijt})$, where υ controls the level of variation in demand relative to the mean demand. We will refer to υ as the "variation coefficient". Each μ_{ijt} is generated from $N^{tr}(\overline{\mu}, 0.2\overline{\mu}, 0.6\overline{\mu}, 1.4\overline{\mu})$, where $\overline{\mu} = \lambda \frac{\sum_{l \in \mathcal{L}} C_l}{I \times J \times T}$. Since $\frac{\sum_{l \in \mathcal{L}} C_l}{I \times J \times T}$ is a constant, the higher λ the higher the demand. We refer to λ as the "load factor" of demand.

5.3 Results on solution accuracy and computation time

In this section, we will compare the solution accuracy and computation time of the EFV approach with the DEM approach. We have generated 10 and 8 problem instances for the medium and large network respectively. The scenario tree is generated through random period-by-period demand sampling based on our demand model.

Table 1 describes the scenario tree and gives the dimensions of both DEM formulations for each problem instance. The first column of the table assigns an identifier to each problem instance. The following four columns focus on the scenario tree, reporting the predefined structure "Tree", the number of periods T, the number of scenarios $|\Omega|$, and the number of nodes $|\mathcal{G}|$. Column "Tree" displays the structure of the tree in the form $A_1^{B_1} A_2^{B_2} \dots$, where A_i denotes the number of children each node in stage i has and B_i denotes the number of periods in stage i. For instance, the structure $3^2 3^2 3^2 3^2$ means the tree has 4 stages, 2 periods in each stage, and each node has 3 children. The rest of the columns show the dimensions of DEM$^{\mathrm{P}}$ and DEM$^{\mathrm{N}}$, including the number of variables n, number of constraints m, number of nonzero elements in the constraint matrix nel, and constraint matrix density $dens := \frac{nel}{n \times m}$ (in %).

Table 2 shows the solution values and computation time of the four approaches, namely DEM$^{\mathrm{P}}$, DEM$^{\mathrm{N}}$, EFV$^{\mathrm{P}}$ and EFV$^{\mathrm{N}}$. The headings are as follows: *solution*, solution value; t_{LP}, elapsed time (in seconds) to solve the problem (where a maximum of 3600 seconds has been imposed); N_{iter}, number of iterations used by the EFV decomposition algorithm; N_z, total number of reference levels generated for each vector x^g by the EFV decomposition algorithm; N_{LP}, total number of LP subproblems solved by the EFV decomposition algorithm; *error*, relative error of the solution value of the EFV decomposition algorithm (in %). For each test network, the last row shows average figures across all problem instances.

The problem instances generated in this paper are challenging because of their large scale, especially in the protection levels formulation. In the largest problem instance in the medium network, problem instance 10, DEM$^{\mathrm{P}}$ contains roughly 1.42 million variables and 1.16 million constraints, while DEM$^{\mathrm{N}}$ contains roughly 1.06 million variables and 0.20 million constraints. In the largest problem instance in the large network, problem instance 18, DEM$^{\mathrm{P}}$ contains roughly 2.62 million variables and 2.30 million constraints, while DEM$^{\mathrm{N}}$ contains roughly 1.97 million variables and 0.66 million constraints. Nevertheless, the EFV decomposition algorithm only takes a few minutes. The longest running time of EFV$^{\mathrm{P}}$ in Table 2 is 251 seconds in problem instance 10, while the longest running time of EFV$^{\mathrm{N}}$ is 159 seconds in problem instance 8.

Compared with DEM, the EFV decomposition algorithm trades solution accuracy for computation time. From Table 2, we have that the average *error* of EFV^P and EFV^N in the medium network is 2.05% and 2.46% respectively. In the large network, the corresponding figures are 1.71% and 2.11%. Given the scale of the instances, these results suggest that the EFV decomposition algorithm generally achieves good solution accuracy.

The EFV decomposition algorithm requires significantly less computation time than DEM. On average, EFV^P and EFV^N reduce the computation time of DEM^P and DEM^N by $353/31 = 11.4$ times and $163/28 = 5.8$ times in the medium network, respectively. The average reduction in the large network is $1112/24 = 46.3$ times and $924/31 = 29.8$ times, for EFV^P and EFV^N respectively. Furthermore, the reduction of computation time is more significant when the problem instance is larger. For example, the reduction in problem instance 10 is $3125/251 = 12.4$ times and $1531/96 = 15.9$ times, for EFV^P and EFV^N respectively. In problem instance 18, the largest problem instance in the large network, both DEM models are stopped after 3600 seconds. The reduction in problem instance 18 is at least $3705/71 = 52.2$ times and at least $3656/44 = 83.1$ times, for EFV^P and EFV^N respectively.

5.4 Results on revenue performance

In this section, we present results on the revenue performance of the four approaches discussed in this paper, namely DEM^P, DEM^N, EFV^P and EFV^N. The popular single-period models, DLP [19], RLP [16] and the perfect hindsight (PH) model are tested as well. PH consists of solving a DLP using the actual demand instead of mean demand and its optimal value is used as an upper bound of the achievable revenue, see [16, 5].

As usual in the literature, we use a rolling horizon simulation to test the revenues of the different approaches, see e.g. [5, 15]. Next, we describe the details on the simulation. We end the section with the discussion on the reported revenues.

5.4.1 Rolling horizon simulation

The input data to the simulation is the length of the simulation horizon, denoted by T^s, and the sequence of booking requests. These booking requests are put into a simulated sales process controlled by a bid-price policy. The simulation is carried out on a rolling-horizon basis. At the beginning of period $t^s \in \{1, \dots, T^s\}$, bid prices are calculated using the corresponding model, which are then used to decide which booking requests to accept in period t^s. At the end of period t^s, the remaining capacity is updated. This process is repeated for $t^s = 1, \dots, T^s$.

In our experiments, we have set $T^s = 8$ and $T^s = 6$ for the medium and large network respectively. The booking requests are generated as follows. Using our demand model, we generate the vector of (actual) demands over the simulation horizon. For each period, the total number of booking requests for each bundle-class combination is equal to the corresponding actual demand, while the bookings for different bundle-class combinations arrive in random order.

In terms of the input data used by each model, in simulation period t^s, PH uses the actual demand in period t^s, while DLP uses the (theoretical) demand mean μ_{ijt^s}. For the rest of tested models, demand scenarios are needed. At the beginning of each simulation period t^s, a scenario tree with time horizon equal to the remaining simulation horizon, i.e. $T = T^s + 1 - t^s$, and 2 children per node is generated. The demands in the scenario tree are generated through random period-by-period demand sampling based on our demand model. To ensure a fair comparison, DEM^{P}, DEM^{N}, EFV^{P} and DEM^{N} use the same scenario tree, while RLP uses the demand scenarios defined by the scenario tree.

5.4.2 Revenue performance

As discussed in Section 5.2, there are two demand parameters: the variation coefficient υ and the load factor λ. In terms of demand variation, we have tested three values, namely $\upsilon = 0.1, 0.3$ and 0.5. In terms of demand load, extreme values will yield very similar revenues for all models. If demand is too low, all methods will simply accept all demand requests, while if demand is too high all methods will accept demand requests for the very profitable bundles only. To cover an interesting range of load factors, we have chosen $\lambda \in \{0.8, 0.9, \dots, 2.8\}$.

For each combination of the two parameters υ and λ, we perform 100 simulation runs, each run with a different random realization of the demand. For each method, we calculate the total revenue across the 100 runs, and report its relative total revenue against that of DLP. For each value of υ, we plot the relative revenues against λ. See Figures 3–5 for the medium network and Figures 6–8 for the large network. To help with the discussion on revenue performance, Table 3 shows the average relative revenue across the entire range of λ, denoted by "AvgRev", for each network and each value of υ.

Throughout all six plots, the following two observations hold. First, the curve of PH cannot be fully shown, since it is much higher than the rest, confirming that PH gives a very loose upper bound of the achievable revenue. Second, the relative revenues of all methods, except for DLP, are above 1, and thus DLP is outperformed by the other six methods, for most of the values of λ. Moreover, in terms of AvgRev, this dominance always holds in our experiments. Therefore, PH and DLP will not be included in our discussion below. The conclusions on relative performance of the remaining five methods are similar in the two networks.

In general, we can say that EFV^{N} is consistently the best method in terms of AvgRev, followed by RLP, DEM^{N}, EFV^{P} and lastly DEM^{P}, for $\upsilon = 0.1$, 0.3 and 0.5. (This only fails for the large network with $\upsilon = 0.5$, where for RLP the AvgRev is equal to 1.0124, while for EFV^{N} is equal to 1.0123.) With respect to the four models discussed in this paper, we have that DEM^{P} and EFV^{P} produce lower AvgRev than DEM^{N} and EFV^{N}, respectively. For instance, in the medium network and when $\upsilon = 0.1$, the respective AvgRev of DEM^{N} and EFV^{N} is 1.0049 and 1.0057, while the respective AvgRev of DEM^{P} and EFV^{P} is 1.0027 and 1.0034. This agrees with the observations in [5], where they argue that the non-anticipativity constraints prevent an adaptive response to the realization of demand.

We also observe that EFV^{P} and EFV^{N} produce higher relative revenues than DEM^{P} and DEM^{N} respectively. The advantage of the EFV approach is more pronounced in the large network, for middle values of λ. For instance, when $\upsilon = 0.3$ and $\lambda = 2$, the respective relative revenue of DEM^{P} and DEM^{N} is 1.00863 and 1.01258, while the respective AvgRev of EFV^{P} and EFV^{N} is 1.01637 and 1.01849. The more pronounced effect in the large network is mostly due to the difference in the structure of the two networks. Recall that the medium network only has single-resource and two-resource bundles, whereas the large network also has four-resource bundles. As discussed above, the bid price of a bundle, if calculated in an additive manner, could deviate from its real opportunity cost. Such deviation gets more serious when the bundle contains a large number of resources, which explains why EFV^{P} and EFV^{N} are more competitive in the large network, where four-resource bundles are present.

6 Conclusions

The computation of bid prices for network revenue management along a time horizon has been considered in this paper. The uncertainty in demand is modeled by means of scenarios yielding an SP formulation. We proposed to solve two existing SP models from the literature using the EFV decomposition algorithm, which brings two main advantages. First, the EFV decomposition algorithm requires remarkably less computation time than DEM. Instances with up to 400 pairs of bundle-fare classes have been solved in less than 5 minutes, while the DEM model with protection levels has up to 2.6 million variables and 2.3 million constraints. In fact, the bigger the problem instance the bigger the difference in computation time. Such difference in computation time makes EFV more useful than DEM in practice, since bid prices usually need to be updated on a daily basis in real world revenue management systems, if not more frequently. Second, contrary to the traditional additive bid prices, the EFV approach is able to define non-additive bid prices on bundles directly. Numerical results based on a rolling horizon simulation in two test networks show that in general, the non-additive EFV bid prices give comparable revenues to those obtained from DEM and RLP. Further, when bundles containing a large number of resources are present in a network, the non-additive bid prices from EFV can improve the revenue of additive bid prices from DEM and RLP.

Two future research directions are worth considering. First, it will interesting to investigate the effect of the network structure on the revenue of non-additive and additive bid prices, and different methods for generating them. As discussed above, the relative performance of bid prices from different methods is quite different for our two test networks. If we can understand exactly how network structure affects the relative performance of different methods then, in practice, we will be able to choose the method that fits the network structure the best. Second, a parallel implementation of the EFV decomposition approach, in which the subproblems in each stage are solved in parallel, would allow handling larger sets of reference levels, ensuring better solution values.

References

[1] D. Bertsimas and I. Popescu. Revenue management in a dynamic network environment. *Transportation Science*, 37:257-277, 2003.

[2] J. Birge and F.V. Louveaux. Introduction to Stochastic Programming, Springer, 1997.

[3] M.P. Cristóbal, L.F. Escudero and J.F. Monge. On stochastic dynamic programming for solving large-scale planning problems under uncertainty. *Computers & Operations Research*, 36:2418-2428, 2009.

[4] L. Chen and T. Homem-de-Mello. Re-solving stochastic programming models for air revenue management. Working paper. Dept. of Industrial Engineering and Management Sciences, Northwestern University, 2006. To appear in *Annals of Operations Research.*

[5] V. DeMiguel and N. Mishra. What multistage stochastic programming can do for network revenue management. Working paper. London Business School, 2008.

[6] E. D'Sylva. OD seat assignment to maximize expected revenue. Technical report. Boeing Commercial Airplance Company, Seattle, WA, 1982.

[7] F. Glover, R. Glover, J. Lorenzo and C. McMillan. The passenger mix problem in the scheduling airlines. *Interfaces*, 12:73-79, 1982.

[8] J.L. Higle and S. Sen. A stochastic programming model for network resource utilization in the presence of multiclass demand uncertainty. In S.W. Wallace and W.T. Ziemba (eds.). Applications of Stochastic Programming. MPS-SIAM series on Optimization, pp. 299-313, 2005.

[9] ILOG. CPLEX 11.0. *http://www.ilog.com/products/cplex*, 2007.

[10] A. Möller, W. Römisch and K. Weber. A new approach to O-D revenue management based on scenario trees. *Journal of Revenue and Pricing Management*, 3:265-276, 2004.

[11] A. Möller, W. Römisch and K. Weber. Airline network revenue management by multi-stage stochastic programming. *Computational Management Science*, 5:355-377, 2008.

[12] S. M. Ross. Introduction to Stochastic Dynamic Programming, Academic Press, 1995.

[13] H. Topaloglu. On the asymptotic optimality of the randomized linear programming method for network revenue management. *European Journal of Operational Rersearch*, 197:884-896, 2009.

[14] B.C. Smith and C.W. Penn. Analysis of alternative origin-destination control strategies. In Proceedings of the Twenty Eight Annual AGIFORS Symposium, New Seabury, MA, 1988.

[15] K.T. Talluri and G.J. van Ryzin. A randomized linear programming method for computing network bid prices. *Transportation Science*, 33:207-216, 1999.

[16] K.T. Talluri and G.J. van Ryzin. An analysis of bid-price controls for network revenue management. *Management Science*, 44:1577-1593, 1998.

[17] K.T. Talluri and G.J. van Ryzin. The Theory and Practice of Revenue Management. Springer, 2004.

[18] R.D. Wollmer. An airline seat management model for a single leg route when lower fare class book first. *Operations Research*, 40:26-37, 1992.

[19] J.T. Wong, F.S. Koppelman and M.S. Daskin. Flexible assignment approach to itinerary seat allocation. *Transportation Research Part B: Methodological*, 27:33-48, 1993.

Medium network					DEM$^{\mathrm{P}}$				DEM$^{\mathrm{N}}$			
ID	Tree	T	$\|\mathcal{G}\|$	$\|\Omega\|$	n	m	nel	$dens$(%)	n	m	nel	$dens$(%)
1	$2^2 2^2 2^2$	6	64	32	3420	2588	15488	0.17499	2268	320	9984	1.37566
2	$3^2 3^2 3^2$	6	365	243	17496	14324	91628	0.03656	13104	2430	75816	0.23810
3	$2^3 2^2 2^2$	7	128	64	6876	5212	35584	0.09929	4572	640	23296	0.79615
4	$3^3 3^2 3^2$	7	1094	729	52488	42998	314216	0.01392	39348	7290	265356	0.09251
5	$2^2 2^2 2^2 2^2$	8	256	128	13788	10460	80384	0.05574	9180	1280	53248	0.45316
6	$3^2 3^2 3^2 3^2$	8	3281	2187	157464	129020	1060712	0.00522	118080	21870	909792	0.03523
7	$2^3 2^2 2^2 2^2$	9	512	256	27612	20956	179200	0.03097	18396	2560	119808	0.25440
8	$3^3 3^2 3^2 3^2$	9	9842	6561	472392	387086	3536396	0.00193	354276	65610	3070548	0.01321
9	$2^2 2^2 2^2 2^2 2^2$	10	1024	512	55260	41948	395264	0.01705	36828	5120	266240	0.14120
10	$3^2 3^2 3^2 3^2 3^2$	10	29525	19683	1417176	1161284	11672036	0.00071	1062864	196830	10235160	0.00489

Large network					DEM$^{\mathrm{P}}$				DEM$^{\mathrm{N}}$			
ID	Tree	T	$\|\mathcal{G}\|$	$\|\Omega\|$	n	m	nel	$dens$(%)	n	m	nel	$dens$(%)
11	$2^2 2^2 2^2$	6	64	32	19000	15800	88000	0.02931	12600	3200	67200	0.16667
12	$3^2 3^2 3^2$	6	365	243	97200	85000	516500	0.00625	72800	24300	510300	0.02885
13	$2^3 2^2 2^2$	7	128	64	38200	31800	201600	0.01660	25400	6400	156800	0.09646
14	$3^3 3^2 3^2$	7	1094	729	291600	255100	1767950	0.00238	218600	72900	1786050	0.01121
15	$2^2 2^2 2^2 2^2$	8	256	128	76600	63800	454400	0.00930	51000	12800	358400	0.05490
16	$3^2 3^2 3^2 3^2$	8	3281	2187	874800	765400	5959700	0.00089	656000	218700	6123600	0.00427
17	$2^3 2^2 2^2 2^2$	9	512	256	153400	127800	1011200	0.00516	102200	25600	806400	0.03082
18	$3^3 3^2 3^2 3^2$	9	9842	6561	2624400	2296300	19847150	0.00033	1968200	656100	20667150	0.00160

Table 1: DEM dimensions of problem instances on medium and large networks

Medium network	DEM$^{\mathrm{P}}$		EFV$^{\mathrm{P}}$						DEM$^{\mathrm{N}}$		EFV$^{\mathrm{N}}$					
ID	*solution*	t_{LP}	*solution*	t_{LP}	N_{iter}	N_z	N_{LP}	*error*	*solution*	t_{LP}	*solution*	t_{LP}	N_{iter}	N_z	N_{LP}	*error*
1	357810	0	351488	0	5	21	526	1.77	360907	0	356473	0	4	17	425	1.23
2	357705	1	354861	2	6	25	2797	0.80	361238	0	357460	2	9	37	4150	1.05
3	347835	0	342251	1	6	25	1247	1.61	350239	0	347002	1	8	33	1649	0.92
4	347632	6	345073	10	10	41	13781	0.74	350543	1	347043	7	10	41	13781	1.00
5	345886	0	342464	3	9	37	3874	0.99	348082	0	344807	0	2	9	927	0.94
6	345946	46	343158	30	10	41	41780	0.81	348400	11	344522	4	2	9	9012	1.11
7	359114	1	347885	1	2	9	1851	3.13	361255	0	346526	9	5	21	4374	4.08
8	359143	347	339715	15	2	9	27030	5.41	361464	94	334872	159	8	33	100746	7.36
9	350940	4	337447	2	2	9	3743	3.84	352737	1	341047	3	10	41	17351	3.31
10	350898	3125	345778	251	10	41	376391	1.46	352930	1531	340330	96	9	37	339490	3.57
Average		353		31	6	26	47302	2.05		163		28	6	28	49190	2.46

Large network	DEM$^{\mathrm{P}}$		EFV$^{\mathrm{P}}$						DEM$^{\mathrm{N}}$		EFV$^{\mathrm{N}}$					
ID	*solution*	t_{LP}	*solution*	t_{LP}	N_{iter}	N_z	N_{LP}	*error*	*solution*	t_{LP}	*solution*	t_{LP}	N_{iter}	N_z	N_{LP}	*error*
11	1995191	1	1964167	2	5	21	526	1.55	1995690	0	1941296	2	6	25	627	2.73
12	1994880	21	1966185	18	8	33	3699	1.44	1995507	9	1976605	9	7	29	3248	0.95
13	1996017	4	1974383	8	8	33	1649	1.08	1996317	1	1968961	4	6	25	1247	1.37
14	1995744	1311	1962784	38	7	29	9728	1.65	1996183	91	1962445	44	10	41	13781	1.69
15	1995625	18	1949571	23	10	41	4295	2.31	1995941	3	1942214	32	5	21	2190	2.69
16	1995380	3645*	1949890	28	2	9	9012	2.28	1995809	3617*	1941420	82	8	33	33588	2.73
17	1996546	191	1954316	5	2	9	1851	2.12	1996744	12	1948032	32	10	41	8579	2.44
18	1982240	3705*	1957124	71	2	9	27030	1.27	1996597	3656*	1951099	44	2	9	27030	2.28
Average		1112		24	5	23	7224	1.71		924		31	7	28	11286	2.11

*Time limit: 3600 secs

Table 2: Solution accuracy and computation time of DEM$^{\mathrm{P}}$, DEM$^{\mathrm{N}}$, EFV$^{\mathrm{P}}$ and DEM$^{\mathrm{N}}$

size	υ	PH	DLP	RLP	$\mathrm{DEM}^{\mathrm{P}}$	$\mathrm{DEM}^{\mathrm{N}}$	$\mathrm{EFV}^{\mathrm{P}}$	$\mathrm{EFV}^{\mathrm{N}}$
MEDIUM	0.1	1.0222	1.0000	1.0049	1.0027	1.0049	1.0034	1.0057
MEDIUM	0.3	1.0257	1.0000	1.0084	1.0041	1.0083	1.0050	1.0090
MEDIUM	0.5	1.0307	1.0000	1.0104	1.0052	1.0097	1.0063	1.0105
LARGE	0.1	1.0347	1.0000	1.0036	1.0015	1.0036	1.0024	1.0048
LARGE	0.3	1.0387	1.0000	1.0095	1.0049	1.0093	1.0061	1.0100
LARGE	0.5	1.0444	1.0000	1.0124	1.0070	1.0120	1.0083	1.0123

Table 3: Average relative revenues of seven methods

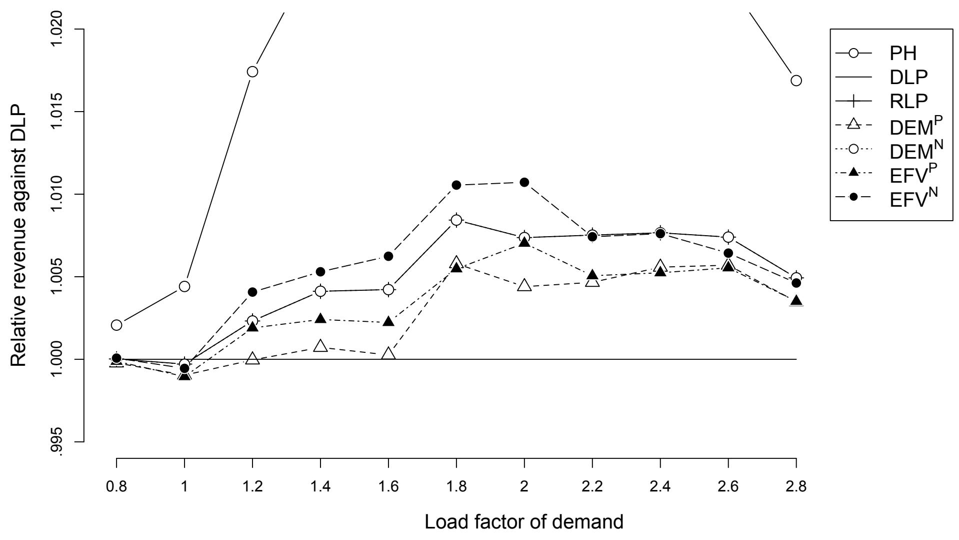


Figure 3: The relative revenues of seven methods in the medium network with $\upsilon = 0.1$

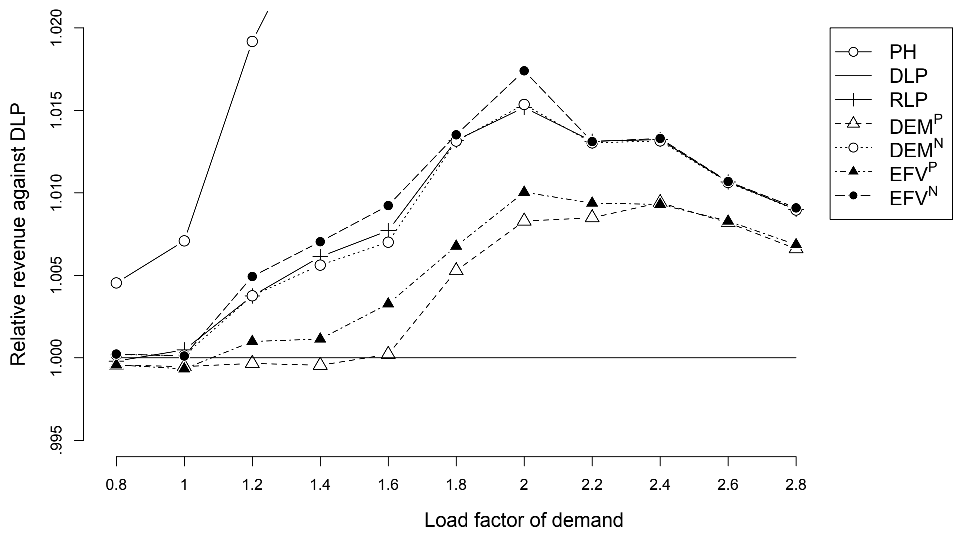


Figure 4: The relative revenues of seven methods in the medium network with $\upsilon = 0.3$

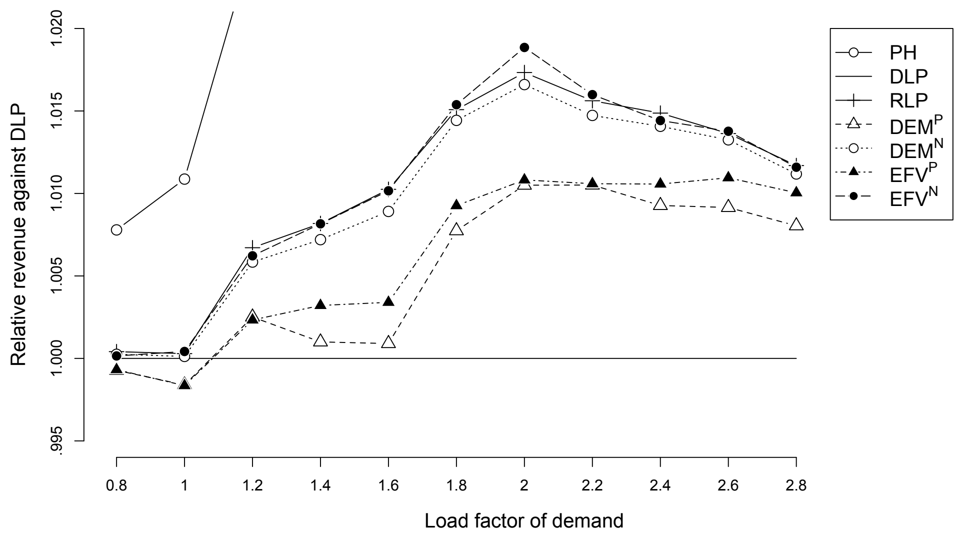


Figure 5: The relative revenues of seven methods in the medium network with $\upsilon = 0.5$

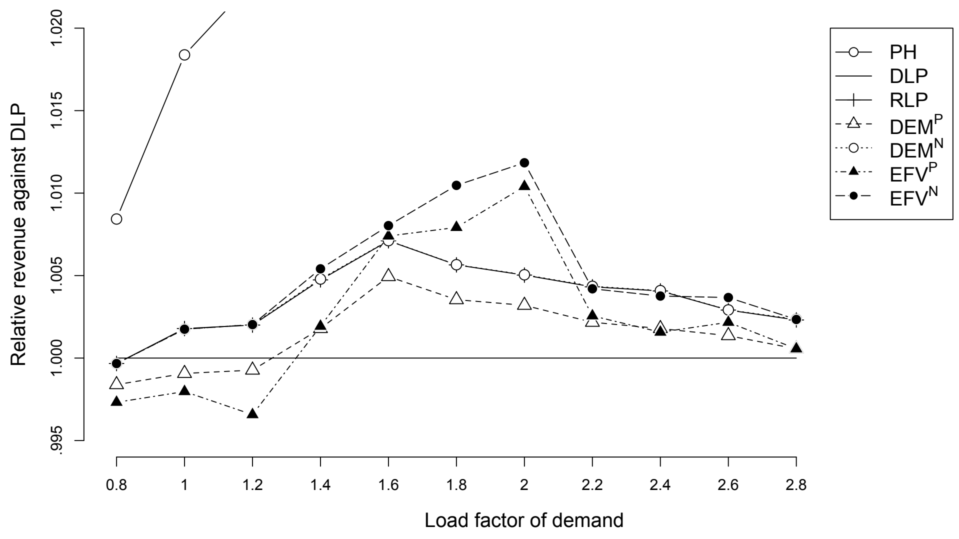


Figure 6: The relative revenues of seven methods in the large network with $\upsilon = 0.1$

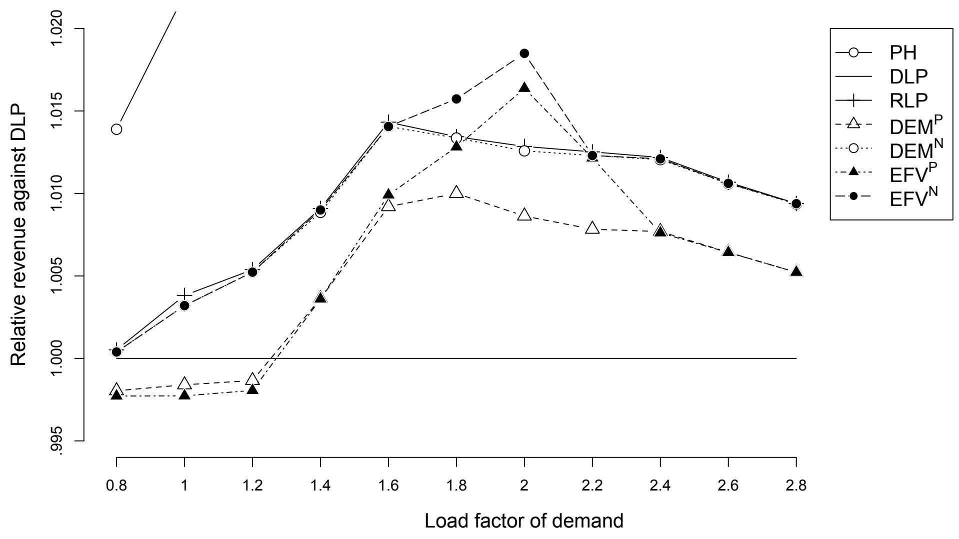


Figure 7: The relative revenues of seven methods in the large network with $\upsilon = 0.3$

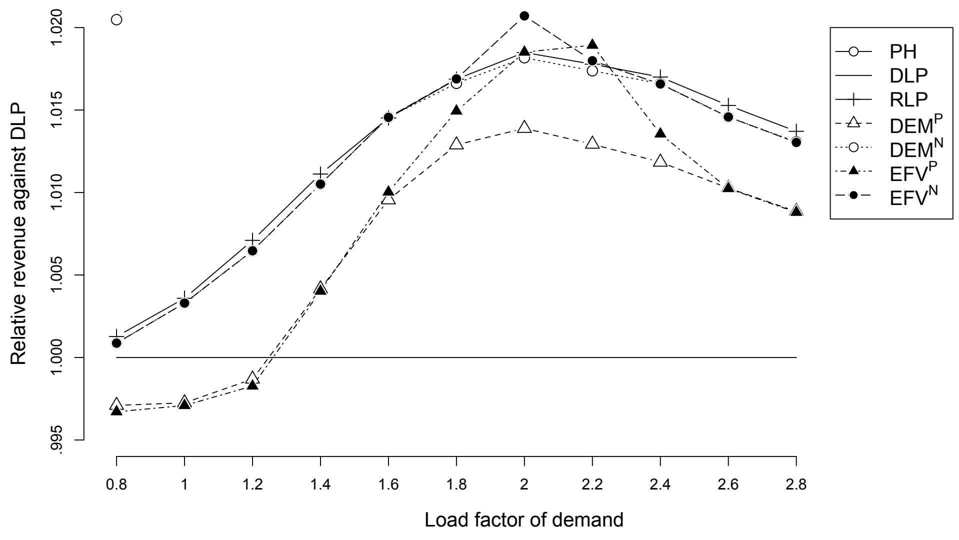


Figure 8: The relative revenues of seven methods in the large network with $\upsilon = 0.5$