A constraint logic programming algorithm for modeling dynamic pricing.

OLIVEIRA, F.S.

2008

A Constraint Logic Programming Algorithm for Modeling Dynamic Pricing

Fernando S. Oliveira
Operational Research and Information Systems, Warwick Business School, Coventry CV4 7AL, U.K.,
Fernando.Oliveira@wbs.ac.uk

We extend Lemke's algorithm in order to solve a dynamic pricing problem. We identify an instance in which Lemke's algorithm fails to converge to the optimal solution (when an optimum does exist) and propose a constraint logic programming solution to this problem. We analyze the complexity of the extended-Lemke's algorithm. Our analysis shows that, in the short-term, dynamic pricing can be used to improve resource management efficiency. It is also shown that dynamic pricing can be used to manage the long-term behavior of demand.

Key words: analysis of algorithms, artificial intelligence, constraint logic programming, dynamic pricing, revenue management, simulation.

1. Introduction

Since Kincaid and Darling's (1963) seminal work, dynamic pricing (a business strategy that adjusts prices in real time in order to increase profits) has become one of the most prolific research topics in operations research and management science. Dynamic pricing has been successfully applied in many service industries (e.g., McGill and van Ryzin 1999, Kimes 1989), such as airlines (e.g., Ladany and Arbel 1991, You 1999) and hotels (e.g., Bitran and Mondschein 1995), in which demand fluctuates heavily from day to day and month to month and capacity of a given resource is fixed, in order to increase profits by extracting value from the varying degree of different customers' willingness to pay. Moreover, dynamic pricing has also been used in manufacturing for improving the management of perishable (e.g., Weatherford and Bodily 1992) and non-perishable (e.g., Federgruen and Heching 1999) inventories.

In order to solve the dynamic pricing problem two main approaches have been used so far, dynamic programming (e.g., Bellman 1957, Bertsekas and Tsitsiklis 1996) and reinforcement learning (e.g., Sutton and Barto 1998, Kaelbling et al. 1996). In this paper we represent the dynamic pricing model as a nonlinear program which we solve using Lemke's algorithm (Lemke 1965).

We show that under certain conditions the Lemke's algorithm fails to converge to the correct solution. We develop an extended-Lemke's algorithm, showing that it always terminates for the dynamic pricing model presented in this paper. We analyze the complexity of the algorithm, showing that the number of visited states (iterations of the algorithm) before the algorithm terminates is a linear function of the number of periods in the planning horizon.

The extended-Lemke's algorithms is developed using constraint logic programming, which enables a straightforward generation of new optimal solutions by first imposing the set of constraints defining the optimum and then looking for solutions that verify this set of constraints. We have implemented our constraint logic programming algorithm using clp(r,q), see Holzbaur (1995), available under different Prolog programs such as SICStus Prolog, ECLiPSe Prolog, YAP prolog, among others.

The challenge of this project was to identify how dynamic pricing can be used to improve resource management in the telecommunications industry (however, all the results presented in this paper can be generalized to other industries). The computational model developed in this paper aims to tackle this problem by illustrating how demand behavior, pricing policies and resource management interact in order to extract additional value from scarce resources. In other words, we aim to analyze the effects of market segmentation and price differentiation on resource management, by: implementing different prices at different times; reducing congestion times and charging the opportunity cost of using a given resource; analyzing how the interaction between a firm's pricing strategies and the expected long-term behavior of demand; simulating the product life cycles and analyzing how prices of a given product change during its life time.

Next, in section 2 we review some of the models of dynamic pricing in the literature and present the life cycle *vs*. weekly revenue problem. In section 3 we present the model developed in this paper for revenue management. In sections 4 and 5 we present the extended-Lemke's algorithm and the simulation results, respectively. Finally, section 6 concludes the paper.

2. Models of Revenue Management

Revenue management (e.g., Gallego and van Ryzin 1997, McGill and van Ryzin 1999, Ciancimino et al. 1999, Botimer and Belobaba 1999, Valkov and Secomandi 2000) uses market segmentation and price discrimination as a tool to extract the highest possible income from customers (Wynter 2004, applies revenue management to the telecommunications industry, using proportionally fair pricing). The increasingly dynamic nature of electronic commerce has produced a shift away from fixed pricing and toward dynamic pricing (Bichler et al. 2002). Bitchler defines flexible pricing as including both differential pricing (in which different buyers may receive different prices based on expected valuations) and dynamic-pricing (in which prices and conditions are based on bids by market participants). Dynamic pricing, together with a better knowledge of demand, and at the same time a good management of supply contracts (Davenport and Kalagnanam 2001, Eso et al. 2001) is essential to improve cost efficiency.

A case in which dynamic pricing can play an important role is when demand fluctuates cyclically throughout a year, a week, or even a day (for example, in electricity retail markets it is possible to buy electricity at different prices for peak and low demand prices).

A short-term application, within the telecommunications industry, is the management of the workforce which serves customers by maintaining networks and repairing faults. The typical behavior of repair volumes during a week is presented in Figure 2.1.

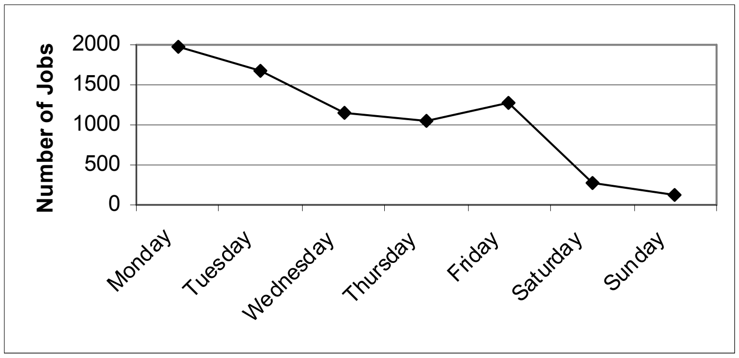
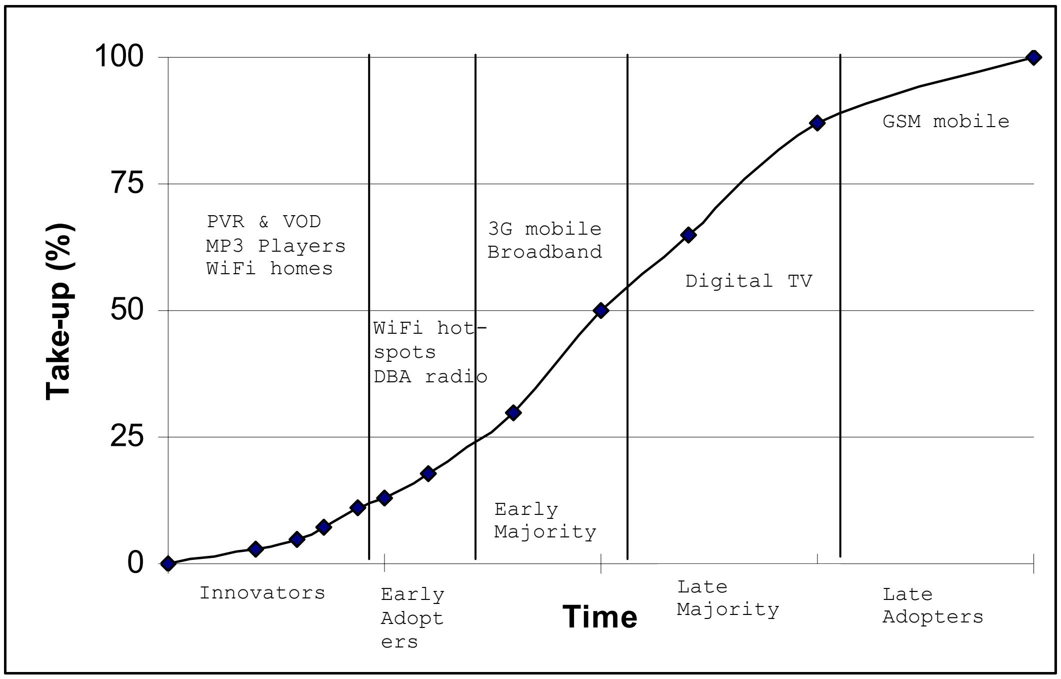


Figure 2.1: Repair Volumes during a Typical Week

Another application is to model the entire life-cycle of a product. The life-cycle of a product is a representation of the number of units of the product sold at each unit of time (usually a year): a typical life-cycle is presented in Figure 2.2.



Source: Ofcom

Figure 2.2: Life Cycle in the Telecommunications Industry

The life-cycle analysis can be applied, for example, to several products in the telecommunications industry, as illustrated in Figure 2.2. The problem faced by a firm is the management of demand over time, as the first customers will tend to be more willing to pay a higher price (innovation or product driven) and the customers buying the product at the later stages of the product's life cycle will have a lower willingness to pay (price driven). Therefore, the firm needs to choose between short-term and long-term profits.

3. Modeling Dynamic Pricing

In this section we describe the short and long-term models, including the equations and the process of parameterization. We use the following notation: *N,* the number of periods in the planning horizon; *t*, any given period in the planning horizon; Q_t, number of units produced at period *t*; P_t, average price at period *t*; C_t, cost of producing one extra unit at period *t*; Π, total profit during the entire planning horizon.

The goal of a firm is to maximize the profit received during the planning period (the weekly, monthly or yearly profit). The total profit earned during the planning horizon is represented by equation (3.1), in which C_tQ_t represents the variable cost in *t* and P_tQ_t represents the total revenue in period *t*.

$$\Pi = \sum_{t=1}^{N} \left(P_t Q_t - C_t Q_t \right) \tag{3.1}$$

The price in period *t*, which is chosen by the firm, depends on the total demand at that period (which is the classical model) and, at the same time, it depends on the prices and demand of the other periods within the planning horizon (it is assumed that customers are price takers). Therefore, the demand for units in each one of the *t* periods can be represented by function (3.2): for products in which a higher price decreases demand $b_{tt} < 0$. Each one of the parameters b_{jt} represents the impact of a price change at time *j* on the quantity demanded of a given product at time *t*. This equation shows that prices of the different periods interact to define the demand level at a given time *t*: by increasing (or decreasing) the price at any other period the number of customers searching for a given product or service at period *t* will also change.

$$Q_t = b_{0t} + b_{1t}P_1 + b_{2t}P_2 + \ldots + b_{tt}P_t + \ldots + b_{Nt}P_N \qquad t = 1,\ldots,N \tag{3.2}$$

Demand is therefore assumed exogenous: this corresponds to an assumption that the firm has some degree of market power, for example resulting from product differentiation, in the services of products modeled. (Different *bs* are associated to different degrees of competition.) In the limit this may correspond to a monopoly. (In this paper we consider a firm that has the monopoly in the products modeled. However, this demand function would also work with imperfectly competitive markets, for as long as the parameters are stationary within the time period modeled).

We tend to solve the system in the inverted form, in which prices are represented as independent variables. Hence, we use the system of equations in (3.3) to represent the inverted demand in the planning horizon: for products in which a higher price decreases demand $a_{tt} < 0$.

$$P_t = a_{0t} + a_{1t}Q_1 + a_{2t}Q_2 + \dots + a_{tt}Q_t + \dots + a_{Nt}Q_N \qquad t = 1,\dots,N \tag{3.3}$$

Each one of the parameters a_{jt} represents the impact of a demand change at time j on the price at time t. This equation shows that demand changes in any period leads to price changes in period t. Equation (3.3) can be represented more compactly by equation (3.4).

$$P_t = a_{0t} + \sum_{j=1}^{N} a_{jt}Q_j \qquad t = 1,\dots,N \tag{3.4}$$

Plugging equation (3.4) into equation (3.1) we can re-write the profit function as equation (3.5) which can be used in the optimization problem.

$$\Pi = \sum_{t=1}^{N}\left[a_{0t}Q_t + \sum_{j=1}^{N} a_{jt}Q_jQ_t - C_tQ_t \right] \tag{3.5}$$

Before proceeding to describe the rest of the model we would like to look more carefully at the inverted demand function, equation (3.4), by describing how, through its parameterization, we can use this model to capture very different types of dynamic pricing problems, i.e., by discussing the typical choice of parameters for a_{jt} when $j \neq t$.

In the typical case of weekly demand, for example, one would expect the increase in production (and a decrease in price) in a given day to decrease the number of units sold during the rest of the week. In this case, a_{jt} will be in the range $-1 \leq a_{jt} \leq 0$.

However, in the case of yearly demand, for example, when analyzing the life cycle of a given product, one would expect that an increase in production (and a decrease in price) in the early years of the life of a product, may lead to a higher growth of the consumption of that product in the future. This effect may also be observed at a monthly level, for example, and it is one of the main reasons for promotions. In this case, a_{jt} will tend to be non-negative, $a_{jt} \geq 0$.

Additionally, we define a set of constraints which the firm follows when defining its pricing policy. We analyze two of the most common constraints: available capacity (number of workers or machinery) and price caps. In both of these constraints we have considered a low and an upper bound: a) there is a lower bound in the case in which management wants to keep a minimum level o production per day, or want to ensure that the resources are used every day, even if this implies incurring a loss; b) for a similar reason, management may feel that the price should not fall below a given threshold, as this may signal low value to its customers.

Let M_t and K_t represent, respectively, the lower and the upper bound for capacity constraint. In this case, equations (3.6) represent the lower and the upper bounds.

$$Q_t \geq M_t \qquad Q_t \leq K_t \qquad t = 1,\ldots,N \tag{3.6}$$

Moreover, let $\underline{P}_t$ and $\overline{P}_t$ represent, respectively, the lower and the upper cap for the price. In this case, equations (3.7) represent the lower and upper price caps.

$$P_t \geq \underline{P}_t \qquad P_t \leq \overline{P}_t \qquad t = 1,\ldots,N \tag{3.7}$$

This is quite a hard problem to solve, as equation (3.5) is nonlinear. We need to use a specific algorithm to optimize the total profit: we use Lemke's algorithm (Lemke, 1965) to solve this problem.

However, as shown in section 4, for some parameters Lemke's algorithm fails to converge. Therefore, in order to solve this problem, we have extended this algorithm to handle multi-stage problems with price and capacity constraints. Next, section 4 describes the basic Lemke's algorithm. We prove that it may fail to converge and we describe how we have extended it in order to tackle this problem.

4. Extending Lemke's Algorithm

In order to solve this model we start by computing the Lagrangian and deriving the set of equations to be solved.

4.1 Deriving the Optimality Conditions

In order to derive the optimality conditions we need to define the slack variables: S_t^M, $t = 1,\ldots, N$, slack variables for the minimum capacity constraint; S_t^K, $t = 1,\ldots, N$, slack variables for the maximum capacity constraint; $S_t^{\underline{P}}$, $t = 1,\ldots, N$, slack variables for the minimum price constraint; $S_t^{\overline{P}}$, $t = 1,\ldots, N$, slack variables for the maximum price constraint.

Moreover, each one of the slack variables has an associated shadow price, representing the value of a unit of the scarce resources: λ_t^M , *t = 1,…, N*, shadow prices for the minimum capacity constraint; λ_t^K , *t = 1,…, N*, shadow prices for the maximum capacity constraint; $\lambda_t^{\underline{P}}$, *t = 1,…, N*, shadow prices for the minimum price constraint; $\lambda_t^{\overline{P}}$, *t = 1,…, N*, shadow prices for the maximum price constraint. Equations (4.1) and (4.2) redefine the capacity constraints and the price caps by using the respective slack variables.

$$M_t + S_t^M - Q_t = 0 \qquad K_t - S_t^K - Q_t = 0 \qquad t = 1,\dots,N \tag{4.1}$$

$$S_t^{\underline{P}} = a_{0t} + \sum_{j=1}^{N} a_{jt} Q_j - \underline{P}_t \qquad t = 1,\dots,N \tag{4.2}$$

$$S_t^{\overline{P}} = \overline{P}_t - a_{0t} - \sum_{j=1}^{N} a_{jt} Q_j \qquad t = 1,\dots,N$$

By computing the derivative of the Lagrangian in order for each one of the quantities we obtain equations (4.3).

For *t = 1,…,N*:

$$a_{0t} - C_t + 2.a_{tt} Q_t + \sum_{\substack{j=1 \\ j \neq t}}^{N} a_{jt} Q_j + \sum_{\substack{j=1 \\ j \neq t}}^{N} a_{tj} Q_j + \lambda_t^M - \lambda_t^K + \sum_{j=1}^{N} a_{tj} \lambda_j^{\underline{P}} - \sum_{j=1}^{N} a_{tj} \lambda_j^{\overline{P}} = 0 \tag{4.3}$$

Furthermore, in order to ensure that every solution of this system of equations is optimal, we need to add the *complementarity conditions*, which establish the relationship between the slack variables and the shadow prices, see equations (4.4).

For *t = 1,…,N*:

$$\lambda_t^M S_t^M = 0 \qquad \lambda_t^K S_t^K = 0 \tag{4.4}$$

$$\lambda_t^{\underline{P}} S_t^{\underline{P}} = 0 \qquad \lambda_t^{\overline{P}} S_t^{\overline{P}} = 0$$

Finally, in order for the system to be complete, we need to add the nonnegativity constraints, equations (4.5). For *t = 1,…,N*:

$$Q_t \geq 0$$

$$\lambda_t^M \geq 0 ,\ \lambda_t^K \geq 0 ,\ \lambda_t^{\underline{P}} \geq 0 ,\ \lambda_t^{\overline{P}} \geq 0 \tag{4.5}$$

$$S_t^M \geq 0 ,\ S_t^K \geq 0 ,\ S_t^{\underline{P}} \geq 0 ,\ S_t^{\overline{P}} \geq 0$$

In conclusion, in order to optimize the profit represented by equation (3.5) subject to the constraints (3.6) and (3.7) we need to solve the system of equations represented by equations (4.1) to (4.5). As the complementarity conditions, i.e. equations (4.4), are nonlinear, we still have a system of nonlinear equations. In order to solve this system by using Lemke's algorithm we first need to relax the system of equations.

4.2 Relaxing the System of Equations

We use a version of Lemke's algorithm that relies on the concept of *homotopy*. We start by finding a feasible solution of a relaxed system of equations. We then iteratively modify and resolve the system until a solution of the original system is reached, or this system is classified as having no solution.

The relaxation process implies the modification of equations (4.1) to (4.3), the definition of a dummy variable Z, the addition of equations (4.6) to our system of equations, and the introduction of an additional dummy variable B, defined in equation (4.7), which is used to relax equations (4.2) and (4.3).

$$Z \geq 0 \qquad\qquad Z \leq 1 \tag{4.6}$$

The dummy variable B is positive and can be computed by the system of equations (4.7). In reality the algorithm will work as long as we choose a B such that there is an initial solution for the system of equations. This is just one of the possible procedures to obtain B. However, the intuitive idea of fixing B to a large value may not work as the value chosen may not be large enough in certain cases and as the solution of the system may have problems with numbers that are too large. This happens if one uses constraint logic programming to solve the system of equations.

For $t = 1,\ldots,N$:

$$B_t = -2\min(a_{tt},0).K_t - \sum_{\substack{j=1\\ j\neq t}}^{N}\min(a_{jt},0)K_j - \sum_{\substack{j=1\\ j\neq t}}^{N}\min(a_{tj},0)K_j + 1 \tag{4.7}$$

then

$$B = \max\left\{B_1,\ldots,B_N, \sum_{j=1}^{N} a_{jt}Q_j, \sum_{j=1}^{N} - a_{jt}Q_j\right\}.$$

First we modify equations (4.1) by multiplying the minimum quantity by Z, to ensure that there is always a solution for this equation. We then obtain a system of equations (4.8).

$$M_t Z + S_t^M - Q_t = 0 \quad K_t - S_t^K - Q_t = 0 \qquad t = 1,\dots,N \tag{4.8}$$

Then, we relax equations (4.2) using the dummy variables *Z* and *B* to obtain a system of equations (4.9) which is always true.

$$S_t^{\underline{P}} = B(1-Z) + (a_{0t} - \underline{P}_t)Z + \sum_{j=1}^{N} a_{jt} Q_j \qquad t = 1,\dots,N \tag{4.9}$$

$$S_t^{\overline{P}} = B(1-Z) + (\overline{P}_t - a_{0t})Z - \sum_{j=1}^{N} a_{jt} Q_j$$

The last set of equations to be relaxed by using the dummy variables *Z* and *B* is (4.3). The new system of equations (4.10) always has a solution.

For $t = 1,\dots,N$: (4.10)

$$B(1-Z) + (a_{0t} - C_t)Z + 2a_{tt}Q_t + \sum_{\substack{j=1\\ j\neq t}}^{N} a_{jt} Q_j + \sum_{\substack{j=1\\ j\neq t}}^{N} a_{tj} Q_j + \lambda_t^M - \lambda_t^K + \sum_{j=1}^{N} a_{tj}\lambda_j^{\underline{P}} - \sum_{j=1}^{N} a_{tj}\lambda_j^{\overline{P}} = 0$$

We now have a set of relaxed equations which under certain initial conditions can always be solved. These equations are (4.4), (4.5), (4.6), (4.8), (4.9) and (4.10).

4.3 From Lemke's to Extended-Lemke's algorithm

As we have shown, the system of equations (4.4) is nonlinear. In order to solve these equations we need to replace the nonlinear by linear equations by fixing the slack variables or the shadow prices to zero. This is the basic idea of Lemke's algorithm: to define which variables are fixed to zero so that the relaxed system has an initial solution.

The Lemke's algorithm proceeds as follows:

Step 1: Given our relaxation of the original equations, an initial procedure to solve the relaxed system is to:

a) Fix the production quantities to the maximum

For $t = 1,\dots,N$: $$Q_t = K_t \tag{4.11}$$

b) Fix to zero the shadow prices for the minimum capacity, and for the maximum and minimum prices

For $t = 1,\dots,N$: $$\lambda_t^M = 0,\ \lambda_t^{\underline{P}} = 0,\ \lambda_t^{\overline{P}} = 0 \tag{4.12}$$

c) Fix to zero the slack variables for the maximum capacity

For $t = 1,\dots,N$: $$S_t^K = 0 \tag{4.13}$$

Step 2: Solve the system of equations having *Z* as an independent variable and choose the value of the non instantiated variables in order to maximize *Z*.

Step 3: Decide termination. If $Z = 1$ the algorithm terminates, as the optimum has been found and returns the value of each one of the Q_t and P_t as a solution. However, if $Z < 1$ the algorithm proceeds to Step 4.

Step 4: If the algorithm does not terminate, then for $t = 1,\dots,N$ and for each pair *j* of shadow price and slack variable, λ_t^j and S_t^j, such that both $\lambda_t^j = 0$ and $S_t^j = 0$, we change to zero the variable that, in Step 2, was not instantiated (and we leave non-instantiated the variable that, in Step 2, was equal to zero). Go to Step 2.

In the case of our model, we can be more precise in the description of Step 4. For $t = 1,\dots,N$, and for each λ_t^M, $\lambda_t^{\underline{P}}$, $\lambda_t^{\overline{P}}$, if $\lambda_t^M > 0$, $\lambda_t^{\underline{P}} > 0$, $\lambda_t^{\overline{P}} > 0$ these variables and the corresponding slack variables $S_t^M = 0$, $\lambda_t^{\underline{P}} = 0$, $\lambda_t^{\overline{P}} = 0$ are not changed. However, if for any $j = M, \underline{P}, \overline{P}$ we have $\lambda_t^j = 0$ and $S_t^j = 0$, then we set S_t^j to zero and leave λ_t^j not instantiated.

In the case of the maximum capacity constraint: for $t = 1,\dots,N$, each shadow price $\lambda_t^K > 0$ and the correspondent slack variables $S_t^K = 0$ are not changed. However, if $\lambda_t^K = 0$ and $S_t^K = 0$, then we set λ_t^K to zero and leave S_t^K not instantiated.

We can now show that, under certain parameters, Lemke's algorithm fails to converge to an optimal solution even when such solution exists. By analyzing equation (4.10) and the initial conditions (4.11) and (4.13) we can see that there is a failure when λ_t^K $\left(\lambda_t^M\right)$and $\lambda_t^{\overline{P}}$ $\left(\lambda_t^{\underline{P}}\right)$ are not instantiated, even though there may be a solution. Proposition 4.1 formalizes this result.

Proposition 4.1 *The Lemke's algorithm fails to solve the dynamic pricing model presented in sections 4.1 and 4.2 when, in a given iteration of the algorithm, a capacity constraint and a price cap are both binding.*

Proof: In equations (4.10) the variables λ_t^K $\left(\lambda_t^M\right)$ and $\lambda_t^{\overline{P}}$ $\left(\lambda_t^{\underline{P}}\right)$ have the same sign. In Step 1 all $\lambda_t^M = 0$, $\lambda_t^{\underline{P}} = 0$, $\lambda_t^{\overline{P}} = 0$, and $\lambda_t^K \geq 0$. If a solution is not reached then, in Step 4:

a) any $\lambda_t^K > 0$ will be kept non-instantiated. If simultaneously, for at least one of the periods *j*, the slack variable for the price cap is equal to zero, then, in the next iteration $\lambda_j^{\bar{P}}$ is non-instantiated. Next, in Step 2 the algorithm fails to instantiate all the non-instantiated variables, as we have a system with more variables than equations. Finally, if *Z* is not equal to zero the algorithm fails in Step 4 as some of the variables are non-instantiated.

b) any $S_t^M = 0$ will be kept instantiated to zero and the respective shadow price λ_t^M will be kept non-instantiated. If simultaneously, for at least one of the periods *j*, the slack variable for the price bottom is equal to zero, then, in the next iteration $\lambda_j^{\underline{P}}$ is non-instantiated. Again, in Step 2 the algorithm fails to instantiate all the non-instantiated variables, as we have a system with more variables than equations. Finally, if *Z* is not equal to zero the algorithm fails in Step 4 as some of the variables are non-instantiated. Q.E.D.

As two shadow prices are non-instantiated, when we maximize *Z* both of them should be driven to zero, but since they have the same sign the system fails to instantiate these variables.

This failure of the Lemke's algorithm is a very important one as in several industries (and not only in telecommunications), such as electricity markets, for example, we have products for which capacity is limited and prices regulated. Moreover, firms may also have a minimum level of resource utilization they need to maintain during a day of work (for instance in order to keep up workers' moral) and at the same time they may not wish to lower the product price below a certain level (as it may reduce the perceived value of the product). In both these cases, the dynamic pricing model cannot be solved using Lemke's algorithm.

This failure leads us to look into ways of improving this algorithm. In order to solve the problem we need to identify which one of the non-instantiated variables can be fixed to zero. We implement a step-by-step procedure which sets each one of these variables to zero (one at a time) and tests if the system has a solution. If at least one solution exists then we fix that variable to zero and proceed to Step 2. The procedure only fails if none of the non-instantiated variables can be set to zero, which means that there is no solution (see Proposition 4.3). Table 4.1 summarizes the extended-Lemke's algorithm for our dynamic pricing model.

Table 4.1: Extended-Lemke's Algorithm for Dynamic Pricing

Step 1: Initialization.

1.a) Define the dummy variable B

For $t = 1,\dots,N$: $$B_t = -2\min(a_{tt},0).K_t - \sum_{\substack{j=1 \\ j\neq t}}^{N} \min(a_{jt},0)K_j - \sum_{\substack{j=1 \\ j\neq t}}^{N} \min(a_{tj},0)K_j + 1$$

$$B = \max\left\{B_1,\dots,B_N, \sum_{j=1}^{N} a_{jt}Q_j, \sum_{j=1}^{N} -a_{jt}Q_j\right\}.$$

1.b) Define the system of equations

$Z \geq 0$, $Z \leq 1$

For $t = 1,\dots,N$:

$M_t Z + S_t^M - Q_t = 0$, $K_t - S_t^K - Q_t = 0$

$$S_t^{\underline{P}} = B(1-Z) + (a_{0t} - \underline{P}_t)Z + \sum_{j=1}^{N} a_{jt}Q_j$$

$$S_t^{\overline{P}} = B(1-Z) + (\overline{P}_t - a_{0t})Z - \sum_{j=1}^{N} a_{jt}Q_j$$

$$B(1-Z) + (a_{0t} - C_t)Z + 2.a_{tt}Q_t + \sum_{\substack{j=1 \\ j\neq t}}^{N} a_{jt}Q_j + \sum_{\substack{j=1 \\ j\neq t}}^{N} a_{tj}Q_j + \lambda_t^M - \lambda_t^K + \sum_{j=1}^{N} a_{tj}\lambda_j^{\underline{P}} - \sum_{j=1}^{N} a_{tj}\lambda_j^{\overline{P}} = 0$$

$\lambda_t^M.S_t^M = 0$, $\lambda_t^K.S_t^K = 0$, $\lambda_t^{\underline{P}}.S_t^{\underline{P}} = 0$, $\lambda_t^{\overline{P}}.S_t^{\overline{P}} = 0$

$Q_t \geq 0$, $\lambda_t^M \geq 0$, $\lambda_t^K \geq 0$, $\lambda_t^{\underline{P}} \geq 0$, $\lambda_t^{\overline{P}} \geq 0$, $S_t^M \geq 0$, $S_t^K \geq 0$, $S_t^{\underline{P}} \geq 0$, $S_t^{\overline{P}} \geq 0$

1.c) Define the initial conditions

For $t = 1,\dots,N$: $Q_t = K_t$, $\lambda_t^M = 0$, $\lambda_t^{\underline{P}} = 0$, $\lambda_t^{\overline{P}} = 0$, $S_t^K = 0$, Z = -1

Step 2: Maximize Z subject to the equations in 1.b)

2.a) For all t and $j = M, \underline{P}, \overline{P}$, such that $\lambda_t^j = 0$ and $S_t^j = 0$, set S_t^j to zero and leave λ_t^j not instantiated.

2.b) For all t if $\lambda_t^K = 0$ and $S_t^K = 0$, set λ_t^K to zero and leave S_t^K not instantiated.

Step 3:

3.a) If $Z = 1$ terminates and goes to step 6, with Z =1.

3.b) If Z<1 proceed to Step 4.

Step 4:

4.a) If Z increased in the last iteration, go to Step 2

4.b) Otherwise, a cycle is found. Proceed to Step 5.

Step 5:

5.a) Start from the first non-instantiated shadow prices for the minimum production, λ_t^M, and then repeated for the non-instantiated shadow prices for the maximum production, λ_t^K :

For all non-instantiated shadow prices for the production (λ_t^M and λ_t^K) :

- Set the shadow price to zero
- If there is no solution, leave the variable not instantiated
- Otherwise, if there is a solution fix that shadow price to zero
- Go to the next non-instantiated shadow price

5.b) If at least one of the non-instantiated variables in Step 5.a) was fixed to zero, go to Step 2, keeping equal to zero the variables instantiated in 5.a).

5.c) If no non-instantiated variable was fixed to zero, the algorithm terminates. Go to Step 6 with $Z < 1$.

Step 6:

If Z = 1: Return for $t = 1,\ldots, N$, Q_t and P_t

If Z < 1: Return "There is no solution."

The main difference between the extended and the simple Lemke's algorithm is in Step 5, by treating the instance of the problem in which two or more shadow prices are simultaneously non-

instantiated. The use of constraint logic programming is crucial for this step of the algorithm, as we iteratively assert values to the non-instantiated variables until we find a feasible solution.
We now prove that the extended-Lemke's algorithm converges to the solution of the dynamic pricing model if there is one (Proposition 4.2), and that it correctly returns that there is no solution (Proposition 4.3).

Proposition 4.2 *When there is a solution of the dynamic pricing model, if the Lemke's algorithm fails to converge for the reasons in Proposition 4.1, then the extended-Lemke's algorithm will converge to the solution of the dynamic pricing model.*

Proof: If there is a solution then there is a set of variables $Q_t \geq 0$, $\lambda_t^M \geq 0$, $\lambda_t^K \geq 0$, $\lambda_t^{\underline{P}} \geq 0$, $\lambda_t^{\overline{P}} \geq 0$, $S_t^M \geq 0$, $S_t^K \geq 0$, $S_t^{\underline{P}} \geq 0$, $S_t^{\overline{P}} \geq 0$, for $t = 1,\ldots, N$, which solves equations (4.10) for $Z = 1$. When in a given iteration of the Lemke's algorithm we have $Z<1$ and two of the shadow prices are non-instantiated (for example λ_t^K and $\lambda_t^{\overline{P}}$) then, as the price and capacity constraints are orthogonal, at least one of these shadow prices is equal to zero. Therefore, by iteratively instantiating these shadow prices a new solution for the system is found. Hence, when there is a solution for the problem and the Lemke's algorithm fails for the reasons in Proposition 4.1, the extended Lemke's algorithm succeeds. Q.E.D.

We now prove that the algorithm returns the correct answer when there is no solution.

Proposition 4.3 *When there is no solution for the dynamic pricing model, the extended-Lemke's algorithm returns 'no solution'.*

Proof: We need to prove that when there is no solution the algorithm always reaches Step 5.c). Since there is no solution *Z* is always less than 1. Therefore, the algorithm follows from Step 2 to Step 4. Now we need to prove that eventually a cycle is reached. There are 4*N* shadow prices in the dynamic pricing problem. This means that the pivoting process in Step 2 has a finite number of possible points to examine. Since a solution will never be found, it will take a finite number of iterations for a cycle to be reached. If in step 5.a) an improvement is reached, then the new Z can change but, as there is no solution, a new cycle will eventually be reached such that, after a finite number of iterations. We are now in step 5.c): the algorithm terminates and returns 'no solution'. Q.E.D.

5. Simulations and Complexity

In this section we use the extended-Lemke's algorithm to simulate the dynamic pricing model, analyzing its complexity. In order to analyze the complexity of the extended-Lemke's algorithm we have simulated one hundred and twenty thousand different problems which were randomly generated. The results are presented in section 5.1.

5.1 Complexity of the extended-Lemke's algorithm

In these simulations we assume a linear demand, $P_t = 1500\text{-}Q_t$, for every period t. We have tested several different planning horizons (seven, twelve, twenty-four, thirty-six, forty-eight and sixty periods), for each one of these time periods we have generated ten thousand different problems. We have tested two different types of problems, one in which a problem randomly generated is likely not to have a solution, and a second one in which the problems always have a solution. We have therefore generated a total of one hundred and twenty thousand problems.

In the first set of experiments (which is presented in Figures 5.1 and 5.2.a)) we have randomly generated the following problems that are not likely to have a solution: a) maximum production capacity was generated between 500 and 1,500 units; b) minimum generation capacity was set between zero and 500 units; c) maximum prices were generated between £500/unit and £1,500/unit; d) the marginal costs were randomly generated between £0/unit and £200/unit.

In the second set of experiments (represented in Figure 5.2.b)) we have relaxed some of the constraints in order to ensure that the problems would always have a solution. We have randomly generated problems as follows: a) maximum production capacity was set between 1,000 and 1,500 units; b) minimum generation capacity was generated to £0/unit; c) maximum prices were generated between £500/unit and £1,500/unit; d) the marginal costs were set to £0/unit.

In Figure 5.1, in order to illustrate the outcome of these simulations, we present the frequency distribution for the experiments with a planning horizon of seven periods, for the type of problems in which a solution is very unlikely to exist (this figure represents the frequency distribution for the number of states visited in 10,000 randomly generated experiments with seven periods each). In this case, the distribution for the number of states (i.e., a state is defined by the shadow prices that are set equal to zero) visited before the algorithm terminates has a mean of 8.1 states and a standard deviation of 1.8 states, with a minimum of four states and a maximum of fourteen states.

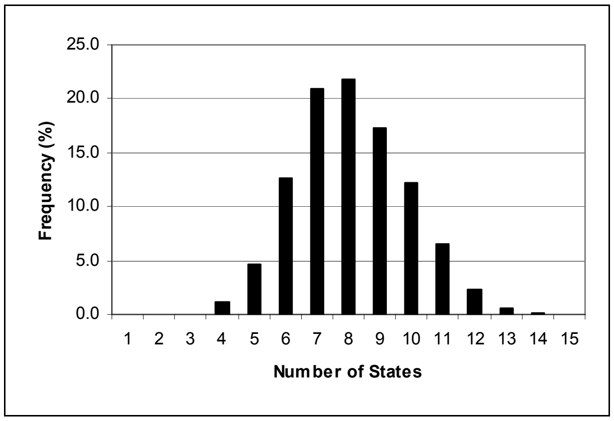


Figure 5.1: Frequency Distribution for the Number of States Visited

The results for all these experiments are summarized in Figure 5.2. The mean number of states visited until the algorithm terminates is a linear function of the number of periods in the planning horizon. (The same conclusion is true for the worst and best cases.) In experiments 5.2.b), in which a solution always exists, the mean number of visited states is higher than in experiments 5.2.a)

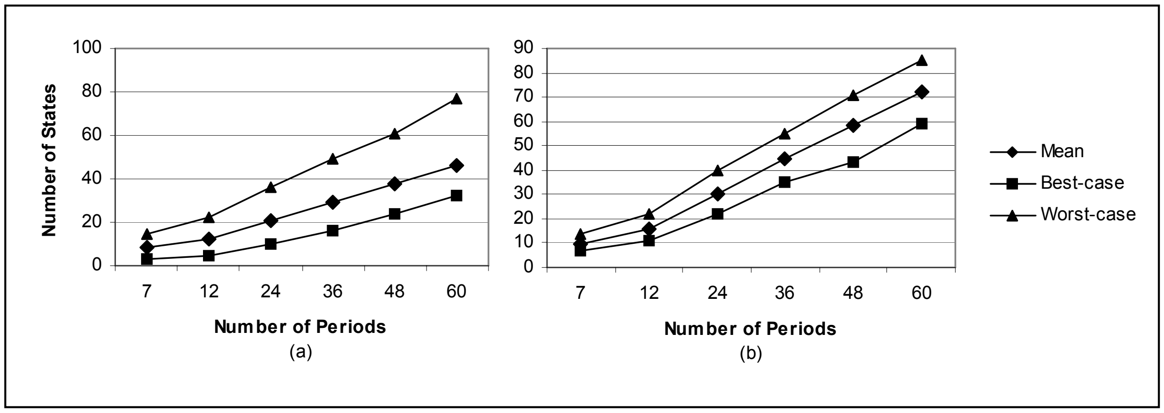


Figure 5.2: Complexity of the Extended-Lemke's Algorithm.

(a) Most or all cases have no solution. (b) All cases have solution.

Next, we illustrate the policies computed by the algorithm for two different dynamic pricing problems.

5.2. Examples of Dynamic Pricing

We have simulated two different sets of experiments. The first set (entitled short-term simulations) presents an application of dynamic pricing to the management of weekly production as an attempt to illustrate how dynamic prices can shift demand between different days of the week. In the second set of experiments (entitled long-term simulations) we apply dynamic pricing to the management of the production over several years. In both sets of experiments we use seven periods.

Let us look at the results of the short-term simulations. In this these simulations we assume that, at any given time: a) Demand decreases by one unit for each pound increase in price. b) An increase in the production in a given day reduces the demand during other days of the week. c) The cost of an additional unit of production was set to zero and the minimum production for each day was also set to zero. d) Demand is higher during the first few days of the week.

In experiment 1: a) maximum production capacity was set to 600 units; b) maximum price was set to £500/unit. In experiment 2: a) maximum production capacity was set to 600 units; b) Maximum price was set to £1,000/unit. In experiment 3: a) maximum production capacity was set to 300 units; b) maximum price was set to £1,000/unit.

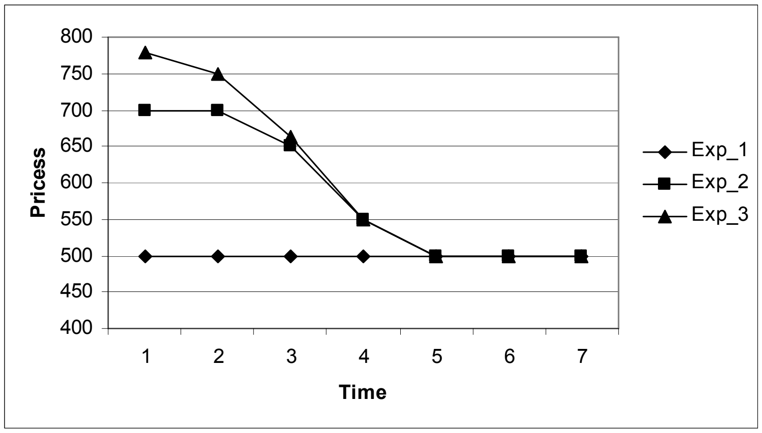


Figure 5.3: Prices per Time Period

As shown in Figure 5.3, by using dynamic pricing, the maximum price increased up to 50% during the first day of the week (assumed to have higher demand). The new prices were never less than the £500/unit which was the maximum price in experiment 1 (Exp_1), and the introduction of a tighter production constraint increased the price by about £50/unit in the first two days of the week.

Dynamic pricing has implications for the level of demand. As shown in Figure 5.4, demand decreases during the first three days and increases during the last three days of the week, as we move from experiment 1, with a price cap of £500/unit, to experiment 2 (with a price cap of £1,000/unit), and experiment 3 (with a price cap of £1,000/unit and a tighter production constraint).

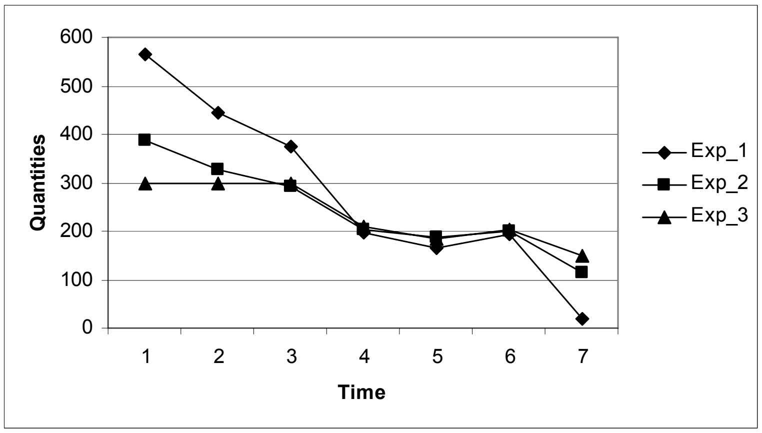


Figure 5.4: Demand (Production) per Time Period

A conclusion from these experiments is that, when it is possible to shift demand from one day to another, dynamic pricing tends to generate production loads that are more even throughout the week, and therefore more efficient. Next we analyze the experiments for the long-term model.

We now look at the results of the long-term simulations. In these simulations we assume that demand in any given year is a negative function of the average price in that year and a positive function of the production during the previous year. More specifically, we assumed that, at any given time: a) demand decreases by one unit for each pound increase in price; b) the firm keeps a given proportion of its customers from the previous year. As in the previous experiments, the cost of an additional unit of production was assumed to be zero, and the minimum production for each year was also assumed to be zero.

In experiment 1: a) The maximum production capacity was set to 3,000 units during the first 4 years and set to 6,000 units during the 3 last years; b) there is no maximum price; c) the firm keeps *50%* of its customers from the previous year. In experiment 2: a) the maximum production capacity was set to 3,000 units during the first 4 years and set to 6,000 units during the 3 last years; b) there is no maximum price; c) the firm keeps *90%* of its customers from the previous year. In experiment 3: a) the maximum production capacity was set to 3,000 units during all the 7 years; b) there is no maximum price; c) the firm keeps *90%* of its customers from the previous year.

Figure 5.5 shows that to have lower prices in order to gain market share and increased future profits can be an optimal policy. Moreover, investment in capacity has important impacts on the pricing policy. Finally, in these experiments, higher capacity leads not only to higher production but also to higher prices, as in period seven of Exp_2. These results are highly dependent on the parameter for customer loyalty.

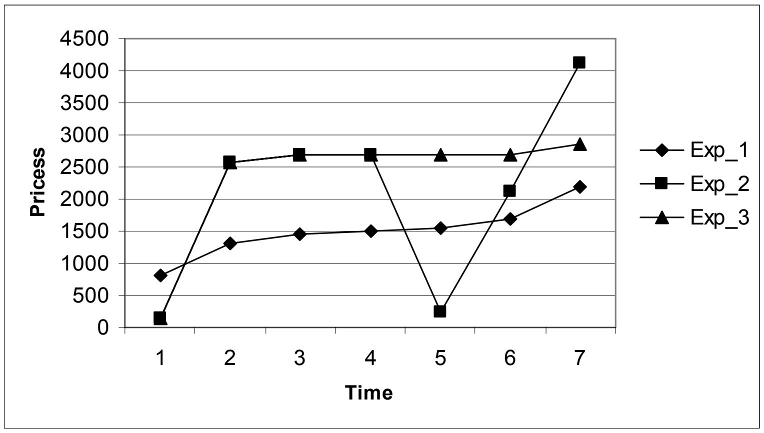


Figure 5.5: Prices per Time Period

6. Conclusions and Discussion

We have shown that dynamic pricing can be used to improve short-term resource management and as a strategic tool, capable of influencing the long-term behavior of customers.

Furthermore, we have shown that the Lemke's algorithm can be used to solve dynamic pricing problems. However, as it fails to converge to the optimal solution under certain conditions, we have extended it, by using constraint logic programming. We have analyzed the complexity of the extended-Lemke's algorithm, showing that its complexity is a linear function of the number of periods in the planning horizon. Furthermore, we solved several examples of the dynamic pricing model, illustrating how capacity constraints, price caps, and the different parameters for the short-term and long-term demand functions influence the optimal pricing policies.

Finally, in the models presented in this paper, we model only one firm which uses dynamic pricing to manage its customers, whose behavior is represented by a static demand function. This corresponds to an assumption that the firm has market power, for example, resulting from product differentiation, in the services and products modeled, and that the demand function is stationary during the planning period. Another important application of Lemke's algorithm can be the solution of dynamic pricing games (as in Kephart et al. 2000), which are not covered in this paper.

References

Bellman, R. 1957. *Dynamic programming*. Princeton University Press

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

Bichler, M., J. Kalagnanam, K. Katircioglu, A. J. King, R. D. Lawrence, H. S. Lee, G. Y. Lin, Y. Lu. 2002. Applications of dynamic pricing in business-to-business electronic commerce. *IBM SYSTEMS JOURNAL* **41 (2)** 287-302.

Bitran, G. R., S. V. Mondschein. 1995. An Application of Yield Management to the Hotel Industry Considering Multiple Day Stays. *Operations Research* **43 (3)** 427-443.

Botimer, T. C., P. P. Belobaba. 1999. Airline Pricing and Fare Product Differentiation: A New Theoretical Framework. *Journal of the Operational Research Society* **50** 1085–1097.

Ciancimino, A., G. Inzerillo, L. Palagi. (1999): A Mathematical Programming Approach for the Solution of the Railway Yield Management Problem. *Transportation Science* **33** 168–181

Davenport, A., J. Kalagnanam. 2001. Price Negotiations for Direct Procurement. Research ReportRC22078, IBMResearch, Yorktown Heights, NY 10598.

Eso, M., S. Ghosh, J. R. Kalagnanam, L. Ladanyi. 2001. Bid Evaluation in Procurement Auctions with Piecewise Linear Supply Curves. Research ReportRC22219, IBMResearch, Yorktown, Heights, NY, 10598.

Federgruen, A., A. Heching. 1999. Combined Pricing and Inventory Control under Uncertainty. *Operations Research* **47 (3)** 454-475.

Gallego, G., G. J. van Ryzin. 1997. A Multiple Product Dynamic Pricing Problem with Applications to Network Yield Management. *Operations Research* **45 (1)** 24–41.

Holzbaur, C. 1995. OFAI clp(q,r) Manual. Edition 1.3.3, Austrian Research Institute for Artificial Intelligence, Vienna, TR-95-09, http://www.ai.univie.ac.at/clpqr/.

Kaelbling, L. P., M. L. Littman, A. W. Moore. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* **4** 237–285.

Kephart, J. O., J. E. Hanson, A. R. Greenwald. 2000. Dynamic Pricing by Software Agents. *Computer Networks* **32 (6)** 731-752.

Kimes, S. E. 1989. A Tool for Capacity-Constrained Service Firms. *Journal of Operations Management* **8 (4)** 348-363.

Kincaid, W. M., D. Darling. 1963. An Inventory Pricing Problem. *Journal of Mathematical Analysis and Applications* **7** 183-208.

Ladany, S. P., A. Arbel. 1991. Optimal cruise-liner passenger cabin pricing policy. *European Journal of Operational Research* **55** 136-147.

Lemke, C. E. 1965. Bimatrix equilibrium points and mathematical programming. *Management Science* **11** 681-689.

McGill, J. I., G. J. van Ryzin. 1999. Revenue Management: Research Overview and Prospects. *Transportation Science* **33 (2)** 233–256.

Sutton, R. S., A. G. Barto. 1998. Reinforcement Learning: An Introduction. MIT Press.

Valkov, T. V., N. Secomandi. 2000. Revenue Management for the Natural Gas Industry. *Energy Industry Management* **1 (1)**.

Weatherford, L., S. Bodily. 1992. A Taxonomy and Overview of Perishable-Asset Revenue Management: Yield Management, Overbooking, and Pricing. *Operations Research* **40 (5)** 831-844.

Wynter, L. 2004. Optimizing Proportionally Fair Prices. *Telecommunication Systems* **27 (1)** 67-83.

You, P.-S. 1999. Dynamic Pricing in Airline Seat Management for Flights with Multiple Flight Legs. *Transportation Science* **33 (2)** 192-206.