DYNAMIC PRICING WITH ONLINE LEARNING OF GENERAL RESERVATION PRICE DISTRIBUTION

Tatsiana Levina * **Yuri Levin** * **Jeff McGill** *
Mikhail Nediak *

* *School of Business, Queen's University, 143 Union St., Kingston, ON, K7L 3N6, Canada*

Abstract: We study the problem of dynamic pricing of perishable items when demand intensity and the reservation price distribution are unknown. The company learns consumer demand through successive observations over consecutive planning horizons by aggregating the forecasts generated from a pool of online stochastic prediction strategies. The methodology of this paper is general and independent of the form of the reservation price distribution.

Keywords: Inventory control, uncertainty, learning algorithms.

1. INTRODUCTION

Dynamic pricing is an increasingly common revenue management practice in industry. Its applications are usually characterized by a finite supply of some inventory item, a fixed time span over which sales occur, and price sensitive demand which may vary stochastically. In the dynamic pricing approach to revenue management, the entire supply is usually treated as homogeneous, while the customers are differentiated through prices which vary dynamically over time. There is an extensive literature on revenue management and related practices. The main references include books (Talluri and van Ryzin, 2004; Phillips, 2005). Representative references on dynamic pricing include (Gallego and van Ryzin, 1994; Zhao and Zheng, 2000). These pricing models are based on exact knowledge of demand characteristics such as customer arrival intensity and the reservation price distribution.

To employ any revenue management practice in general, and dynamic pricing in particular, one needs to know the future demand (for example, as a function of time and price). However, the exact value of demand can rarely be known with certainty. Moreover, even stochastic models recognizing the uncertain nature of demand assume some knowledge of demand distributions which cannot be known precisely in practice. This uncertainty in demand characteristics has long been recognized in the economics, marketing/pricing, inventory management, and revenue management literature. Balvers and Cosimano (1990) study a pricing problem with learning of linear demand with no limits on sales due to inventory levels. Petruzzi and Dada (2002) consider a stocking and pricing model with a fixed but unknown perturbation of some given demand function. Recent papers (Bertsimas and Perakis, 2003; Aviv and Pazgal, 2002; Lin, 2005) study the pricing of a fixed stock of items over a finite horizon with demand learning. However, only Bertsimas and Perakis (2003) consider learning of all demand characteristics including the price sensitivity. Their approach assumes a linear demand model with normal perturbations. The latter two papers assume a known reservation price distribution.

In our view, since demand evolves in time, demand learning naturally belongs to the class of *online*

methods. Furthermore, to ensure wide applicability in practice, an effective methodology should be free from specific distributional assumptions. A sufficiently general approach to online learning is the Aggregating Algorithm (AA) (Vovk, 1999) which was originally developed to address the problem of combining expert advice (Vovk, 1990). Similar techniques and variations of AA have been applied to the problem of online portfolio selection since the work of Cover (1991). Our numerical technique is based on the aggregation of Markov switching strategies (a variation of AA), a method applied to online portfolio selection by Levina (2004). Online learning methods are particularly appropriate in situations where historical data are scarce or irrelevant due to changes in environment.

2. PRICING MODEL

We consider a product with limited availability sold by a monopolistic company over several planning horizons, each comprising T decision periods: $\{0, 1, \ldots, T-1\}$. The initial inventory of the product at time 0 is Y, with no replenishment possible during a planning period, and each planning period starts with the same initial inventory. At time T, the product expires and all unsold items are lost.

We assume that customers arrive according to a discrete-time counting process $N(t)$ with at most one arrival per time period and the probability of arrival λ in each period. Such an arrival process is a discrete-time approximation of the Poisson process, which is a standard model in the dynamic pricing literature (Gallego and van Ryzin, 1994; Zhao and Zheng, 2000). We assume that the company wishes to reassess its pricing after each sale or offer of sale, and the sales process unfolds in an 'orderly' fashion; that is the company presents items for sale sequentially, one per time period. Each arriving customer compares a directly unobservable quantity, his or her *reservation price*, to the current price p, and makes a purchase if the reservation price exceeds p. The reservation prices of arriving customers are modelled as independent continuous random variables with the distribution $F_t(\cdot)$ for a customer arriving at time t. Given the current price p, the probability of a sale occurring in the current time interval is $\Lambda(t,p) = \lambda(1 - F_t(p))$. Gallego and van Ryzin (1994) considered a case when $F_t(p)$ is *stationary*, *i.e.* independent of t, while Zhao and Zheng (2000) studied a general, non-stationary case. In practice, both the arrival probability λ and the reservation price distribution $F_t(p)$ are usually not known precisely and have to be learned. The learning problem unfolds within, as well as between the consecutive planning horizons. Within each planning horizon, the company attempts to price its product optimally, taking into account the state of its current knowledge of the market.

All pricing policies considered in this paper are of the *state-feedback* form, and the possible values of the price belong to a given set Π. If all model parameters are known then the state of the system at time t can be described in terms of remaining inventory y (see, for example Gallego and van Ryzin (1994)). If parameters are not known precisely, the optimal decision has to be based on the *complete* state description which includes entire histories of the sales and price processes. This is because the pricing policy depends on predictions of future demand intensity and reservation prices which, in turn, are dependent on the information contained in sales and price histories.

From the policy optimization standpoint, it is important to know the purchase probability $\Lambda(t,p) = \lambda(1 - F_t(p))$ as a function of time and price. Thus, we need to specify how current information about $\Lambda(t,p)$ is represented. One possibility is to assume that the initial information about λ is contained in a given prior distribution, while the reservation price distribution $F_t(p)$ is of a fixed parametric form (with initial information about parameters also contained in some given prior distribution). The resulting approach, while quite general, will be parametric. Another possibility, when the set of prices Π is discrete, is to assume a *semi-parametric* model for $\Lambda(t,p)$. In this case, we need to specify the form of $\Lambda(t,p)$ as a function of t (determined by a fixed set of parameters) for each fixed $p \in \Pi$. The initial information about parameters can be described by an appropriate prior distribution. Note that when demand is stationary, this approach results in a completely *nonparametric* demand model. On the other hand, there is no practical nonparametric way to represent time dependence and, for each p, the functional dependence of $\Lambda(t,p)$ on t has to be described by a set of parameters. A modest restriction on the model is that $\Lambda(t,p)$ needs to be nondecreasing in p. In either case, the unknowns which determine market operation can be represented by a single parameter vector $\mathbf{x}$. The sale probability at time t and price p corresponding to parameter values $\mathbf{x}$ is denoted as $\Lambda^{\mathbf{x}}(t,p)$.

The company attempts to set its pricing policy so that the expected revenues are maximized. To compute the optimal policy, it should use the most current knowledge about the parameter vector $\mathbf{x}$. The company starts with some initial (prior) distribution for the value of $\mathbf{x}$ and updates it when additional information becomes available. We let the histories at time t be denoted as $\mathcal{N}_t$ (the list of previous sale times) and $\mathcal{P}_t$ (the list of all previous prices used). At time t, the list $\mathcal{P}_t$ has the length t, and the list $\mathcal{N}_t$ has the length $Y - y$, where y

is the current level of inventory. At the beginning of a planning horizon, the prior distribution for $\mathbf{x}$ results in a random vector denoted as $\mathbf{x}(0, \emptyset, \emptyset)$. A posterior parameter distribution at time t given histories $\mathcal{N}_t, \mathcal{P}_t$ leads to a random vector denoted as $\mathbf{x}(t, \mathcal{N}_t, \mathcal{P}_t)$. The posterior distribution is obtained by multiplying the prior density (distribution of $\mathbf{x}(0, \emptyset, \emptyset)$ at a particular parameter vector $\mathbf{x}$) and the likelihood of the observed sales history (for particular parameter values $\mathbf{x}$), where p_τ denotes the price at time τ:

$$L(t, \mathcal{N}_t, \mathcal{P}_t \mid \mathbf{x}) = \prod_{\tau \in \mathcal{N}_t} \Lambda^{\mathbf{x}}(\tau, p_\tau) \times \prod_{\tau \in \{0,\dots,t-1\} \setminus \mathcal{N}_t} \big(1 - \Lambda^{\mathbf{x}}(\tau, p_\tau)\big). \quad (1)$$

The company could use $(t, \mathcal{N}_t, \mathcal{P}_t)$ as a state description, and try to find the state-feedback pricing policy in the form $p(t, \mathcal{N}_t, \mathcal{P}_t)$. Using the posterior distribution, the transition probability from state $(t, \mathcal{N}_t, \mathcal{P}_t)$ to $(t, \mathcal{N}_t \cup t, \mathcal{P}_t \cup p)$ is

$$d(t, \mathcal{N}_t, \mathcal{P}_t, p) = E_{\mathbf{x}(t, \mathcal{N}_t, \mathcal{P}_t)}[\Lambda^{\mathbf{x}(t, \mathcal{N}_t, \mathcal{P}_t)}(t, p)], \quad (2)$$

where p is the price set by the company. These transition probabilities can be used to construct a dynamic program for finding $p(t, \mathcal{N}_t, \mathcal{P}_t)$ in every state. However, an obvious drawback of such a dynamic program is the state space explosion which makes it computationally intractable except for some very special cases. A possible heuristic solution is to sacrifice the accuracy of the posterior distribution of parameters by only using the price and sales history information in an aggregate form. In the extreme case, we can collapse the entire posterior distribution to a single value, for example, the average over the posterior distribution which, for history $(t, \mathcal{N}_t, \mathcal{P}_t)$, is denoted as $\bar{\mathbf{x}}(t, \mathcal{N}_t, \mathcal{P}_t)$.

Using the average of the distribution, the heuristic policy can be computed under the assumption that the average of the last posterior distribution is the actual value of demand model parameters. Suppose that, at time $\bar{t}$, the observed history is $(\bar{t}, \mathcal{N}_{\bar{t}}, \mathcal{P}_{\bar{t}})$. Then the policy for the rest of the planning horizon is computed using the inventory level as the state description. Let $R(t, y)$ denote the expected future revenue if the inventory level at time t is y and the true value of demand parameters is known. Let $\bar{\mathbf{x}} = \bar{\mathbf{x}}(\bar{t}, \mathcal{N}_{\bar{t}}, \mathcal{P}_{\bar{t}})$, then $p = p(t, y)$ is the maximizer in the dynamic program:

$$R(t, y) = \max_{p \in \Pi} \Big\{ \Lambda^{\bar{\mathbf{x}}}(t, p)\big(p + R(t+1, y-1)\big) + (1 - \Lambda^{\bar{\mathbf{x}}}(t, p)) R(t+1, y) \Big\}, \quad y = 1, \dots, Y - |\mathcal{N}_{\bar{t}}|, \ \bar{t} \le t \le T - 1, \quad (3)$$

with the boundary conditions

$$R(T, y) = 0, \ y = 1, \dots, Y - |\mathcal{N}_{\bar{t}}|, \text{ and} \quad (4)$$

$$R(t, 0) = 0, \ t \in \{\bar{t}, \dots, T\}. \quad (5)$$

The learning process and pricing policy execution proceeds as follows. A planning horizon is split into a given number of subintervals. These subintervals may correspond to the individual decision periods (of length 1) or consist of several periods. Suppose that all subintervals have equal length. In the first subinterval, the company uses a pricing policy corresponding to the average of the prior distribution, $\bar{\mathbf{x}}(0, \emptyset, \emptyset)$. After observing the history (let it be represented by $(\bar{t}, \mathcal{N}_{\bar{t}}, \mathcal{P}_{\bar{t}})$), the company updates the parameter distribution to that of $\mathbf{x}(\bar{t}, \mathcal{N}_{\bar{t}}, \mathcal{P}_{\bar{t}})$ and recomputes the policy using the updated average $\bar{\mathbf{x}}(\bar{t}, \mathcal{N}_{\bar{t}}, \mathcal{P}_{\bar{t}})$. In the second subinterval, it applies the new policy, observes the new history, updates the parameter distribution, and so on. After the present planning horizon is completed or all inventory is sold out, the final posterior distribution of parameters becomes the prior distribution of a subsequent planning horizon. Use of subintervals of length 1 corresponds to "online" learning, while subintervals of length T (only one "subinterval" per planning horizon) correspond to "offline" learning. We refer to the length of subintervals as *learning periodicity*. While learning offline is likely to be inadequate, a fully online approach may be computationally excessive. For most distributions, it is impossible to perform the updates in the exact form. Therefore, we employ finite sample approximations to prior and posterior distributions. The form of distribution updates allows for extra flexibility. In addition to purely Bayesian ideas explored in this section, we can consider complementary approaches developed in the learning literature such as the Aggregating Algorithm discussed in the next section.

3. AGGREGATING ALGORITHM: UPDATES TO POSTERIOR DISTRIBUTION

Our distribution update procedure can be understood from several perspectives. The original motivation for this procedure comes from the Aggregating Algorithm of Vovk (1999) applied to a certain type of stochastic prediction strategies (Markov switching strategies, see, for example, Levina (2004)). It can also be viewed as a Bayesian computational procedure.

The Aggregating Algorithm is based on a very general model of learning as a sequential perfect information game between the following players: Decision Maker (DM), Pool, and Reality. DM is an entity which has to choose a decision from a particular set of decisions (for example, to make a forecast). Pool can be viewed as a set of experts or prediction strategies whose recommendations

about decisions can be considered by DM. Finally, Reality is an entity which determines the actual payoffs for the decisions made by DM as well as all experts (strategies) in the Pool. The game, which *does not generally need to be adversarial*, can be described by the following **Protocol:**

```
for each game stage:
    Experts in the Pool make their recommendations
    DM views recommendations and makes a decision
    Reality determines payoffs of experts and DM.
```

The players have perfect information in the usual sense that payoffs, decision sets and decisions in the previous stages are known to all participants. We will now associate these "players" with the elements of our problem. In our case, DM is the company. While it is tempting to think that the company decision for learning purposes is the price, it seems impossible to construct a meaningful perfect information game with any nontrivial set of experts since the payoff (revenue earned) of any pricing policy except for the one actually selected by the company is unknown in practice. Therefore, the decision of the company is associated with the demand parameter vector which is used for the heuristic policy. This parameter vector is a prediction in a sense that it predicts future demand. Pool and its experts represent alternative demand predictions (parameter vectors). In terms of AA's formalism, Pool can be any measurable set which may even include possible realizations of a stochastic process. In our case, we use the set of realizations of a Markov process (random walk on the set of parameter vectors) as stochastic prediction strategies. Finally, Reality determines the demand realization (sales) resulting from the application of the pricing policy which corresponds to the parameter vector selected by DM. One natural "payoff" function which describes how well the forecast (parameter vector) fits the observed data is the log-likelihood.

To track relative strategy performance, AA keeps a record of the current "weight" for each strategy in the pool starting from some initial (prior distribution) values. In our case, if Pool corresponded to constant parameter vectors, then the weight of a vector would be equal to the value of the posterior distribution at this vector. In the case of strategies which switch values according to some random walk, the weight of each particular realization (sequence of vectors) is proportional to the product of the likelihoods of consecutive history segments observed after each respective recommendation (vector) in the sequence.

The Aggregating Algorithm describes how to "merge" the recommendations of the strategies, taking into account both the actual values of recommendations and the current weights of the strategies making them. One common variation of AA merges by taking a weighted average of recommendations using the current weights. We emphasize that this is just one of many possible ways to merge. This operation could be written explicitly for the case of constant strategies. However, due to accumulation of terms in the likelihood, the computation of the weighted average will progressively become more difficult as DM acquires more and more data. An alternative, which we discuss next, is based on the Markov switching strategies and a finite sample approximation, and appears to be much more attractive computationally.

As we have already stated, the distribution of $\mathbf{x}$ is approximated by a discrete sample $\mathcal{W} = (\mathbf{x}_i)_{i=1}^N$. After observing the sales and price history, we generally want to perform an update to the distribution of parameters and recompute the heuristic pricing policy using the resulting new posterior distribution mean.

Suppose we want to update a finite-sample approximation $\mathcal{W}$ to the prior distribution (corresponding to $\mathbf{x}(0, \emptyset, \emptyset)$) to obtain a finite-sample approximation $\mathcal{W}'$ to the posterior distribution (corresponding to $\mathbf{x}(t, \mathcal{N}_t, \mathcal{P}_t)$). This is done using a Monte-Carlo accept-reject algorithm with bootstrap-like resampling of the elements of $\mathcal{W}$, (see Levina (2004)). The elements of the new sample are denoted as $\mathbf{x}'_k$ where k is the counter of points in the new sample.

Algorithm: accept-reject bootstrap resampling
Inputs: finite-sample (size N) approximation $\mathcal{W}$ of $\mathbf{x}(0, \emptyset, \emptyset)$,
sales and price process history $(t, \mathcal{N}_t, \mathcal{P}_t)$
Output: finite-sample (size N) approximation $\mathcal{W}'$
of $\mathbf{x}(t, \mathcal{N}_t, \mathcal{P}_t)$
Set $k := 0$
Set $L_{\max} := \max_{i=1,\ldots,N} L(t, \mathcal{N}_t, \mathcal{P}_t \mid \mathbf{x}_i)$
while $k < N$ **do**
  Choose u from $U[0, 1]$
  Choose j randomly from $\{1, \ldots, N\}$
  if $u \leq \frac{L(t, \mathcal{N}_t, \mathcal{P}_t \mid \mathbf{x}_j)}{L_{\max}}$ **then**
    Set $k := k + 1$
    Set $\mathbf{x}'_k := \mathbf{x}_j$
  endif
enddo

When AA is applied to Markov switching strategies, $\mathcal{W}'$ represents a finite sample approximation to the distribution of current values of the random walk realizations. The value of this distribution at a particular parameter vector represents the combined current weight of all strategies which recommended this vector at the previous stage of the game. For the purposes of merging the recommendations in the next step, AA is concerned only with this combined weight because of the Markovian property of the strategies. Future recommendations for this discrete set of strategies are easily obtained by sampling a step of the random walk from their current values. This results in a new sample $\mathcal{W}''$. The random walk can make

local moves (for example, Gaussian random walk) as well as large, nonlocal moves (for example, a reset back to the prior distribution).

We can also explain the resulting update procedure from a genetic algorithm perspective. The sample $\mathcal{W}$ represents a population of parameter vectors. The fitness of each parameter vector is evaluated according to the likelihood function of the observed sales and price history given this parameter vector. The number of offspring of the vector in the new sample is proportional to its fitness function. Each offspring also undergoes a random mutation. This mutation is usually small (e.g., Gaussian random walk), but sometimes a strong deviation from the parent point appears (reset to a prior distribution). This achieves diversity in the sample and an appropriate trade off between exploitation of previous good values (keeping duplicates of points with high likelihood values) and exploration (random deviation from the previous values).

4. NUMERICAL ILLUSTRATIONS

In this section we report two numerical illustrations for the algorithm. In the first experiment, we use a parametric demand model to learn demand which is actually described by a model in the same class. In the second experiment, we use a general demand model to learn the demand of a form which is unknown to the company.

The demand model used in the first experiment is non-stationary of the form $\Lambda(t,p) = \lambda(1 - F_t(p))$, where the reservation price is normally distributed with mean $\mu(t) = \mu_0 + \mu_1 t/T$ and standard deviation σ. Therefore,

$$\Lambda^{\mathbf{x}}(t,p) = \lambda(1 - \Phi((p - \mu(t))/\sigma))$$

with the parameter vector of the form $\mathbf{x} = (\lambda, \mu_0, \mu_1, \sigma)$. The true parameter values were $\lambda = 0.1$, $\mu_0 = 3$, $\mu_1 = 1$ and $\sigma = 1$, while the prior distribution was uniform over the hyperrectangle $[0.01, 0.5] \times [0, 10] \times [-3, 3] \times [0.1, 5.0]$. The pricing model used a time horizon of $T = 2000$ time intervals, the initial inventory $Y = 50$, and a discrete price set Π with 20 values in the range $\{0.5, 1, 1.5, \ldots, 9.5, 10\}$. We simulated 20 independent replications of the learning process by means of AA over 10 consecutive planning horizons. The parameter distribution approximation used 40000 discrete sample points. Random walk perturbations to parameter values were distributed as Student's t distribution with 3 degrees of freedom scaled by 0.1 for μ_0, 0.05 for μ_1, 0.05 for σ and 0.01 for λ and truncated to ensure the walk remains within the bounds specified by the prior distribution. The reason for the use of Student's t distribution is its heavier tails than, for example, those of the normal distribution, resulting in somewhat more active exploration of the parameter space. Probability of a reset to the prior distribution was 0.001 for each strategy. We compared three runs with different periodicity of parameter distribution updates (times between AA stages): 2000 (offline), 200 and 20. (For the last two runs, the step scale parameters were reduced by the factors of $\sqrt{0.1}$ and $\sqrt{0.01}$, while the reset probability was reduced by factors of 0.1 and 0.01, respectively). Figure 1 shows the expected revenues for the true parameter values (solid line) together with the average revenues over 20 simulations for each of 10 consecutive planning horizons for each value of periodicity (dashed lines). We see that there is a significant initial benefit from decreasing periodicity from 2000 (offline) to 200. However, the benefits from further reduction (more online) are not clear. The difference is noticeable only during the first planning horizon. After that all three lines remain very close to true optimum. The standard deviation of total revenues over 20 replications for each learning periodicity are shown in Figure 2. The standard deviation obtained in learning situations is comparable to the standard deviation of total revenues under the optimal policy for true parameter values (solid line), has a tendency to decrease as the learning process unfolds, and appears to be smaller for learning periodicities 200 and 20 than for offline learning. While keeping all experiment parameters the same and fixing learning periodicity at $T/10$, we have also examined the effects of a shorter time horizon T which is equivalent to a smaller expected number of arrivals λT. The algorithm remained an effective learning tool for values of T above 500 (thus, $\lambda T \geq 50$).

The second experiment concerns a stationary (independent of time) semi-parametric (in the sense of Section 2) model. Here we observe that if the set of admissible prices Π is discrete, then it is sufficient to learn the products x_i of the arrival probability and the probability of the reservation price falling between two consecutive values p_i and p_{i+1} of Π (to simplify notation, assume that $p_0 = -\infty$ and $p_{n+1} = \infty$, where n is the number of finite price values in Π). Specifically, in this experiment we assume that demand is stationary and model the purchase probability as

$$\Lambda^{\mathbf{x}}(t,p) = \sum_{p_i \in \Pi:\, p_i \geq p} x_i.$$

This method is applied to learn an exponential demand model of the form $\Lambda(p) = 0.1 \exp(-p/3)$. We make the learning task deliberately difficult by using a true model which is quite different from any curve in the initial population. The latter corresponds to a random sample of stationary ($\mu_1 = 0$) normal models similar to the ones considered in the previous example. The parameters (λ, μ_0, σ) of

these curves are sampled from the uniform prior distribution over $[0.01,\ 0.5] \times [0,\ 5] \times [0.1,\ 5.0]$. We used the periodicity of 200, probability 0.01 of a reset to the prior distribution and the scaled geometric Gaussian (correlated) random walk on the space of parameters. That is, a new parameter vector $\mathbf{x}'$ is sampled according to

$$x_i' = \frac{x_i e^{\xi_i}}{\sum_{j=0}^{n} x_j e^{\xi_j}},$$

where $\mathbf{x}$ denotes an old parameter vector, $\xi_0 = \epsilon_0$, and $\xi_i = 0.7\xi_{i-1} + \epsilon_i$, $i = 1, \ldots, n$, and ϵ_i's are sampled independently from the normal distribution with mean 0 and standard deviation of $0.1/\sqrt{10}$. Other problem characteristics are the same as in the previous example. The results of 10 simulations over 20 consecutive planning horizons are given in Figure 3. This figure shows the optimal expected revenue (for the true demand model), the average (over 10 simulations) of the anticipated expected revenue resulting from the first policy computation in each planning horizon, and the average (over 10 simulations) of the actual total revenue for each planning horizon. AA finds a reasonable approximation to the true demand model right in the first planning horizon. The average and the standard deviation of the total revenues over all planning horizons of all simulations were 200.3 and 19.6, respectively. The standard deviation of the overall average is 1.4. We see that the average performance of the algorithm, 200.3 ± 1.4, is very close to the optimum of 202.4.

In conclusion, we remark that the approach of this paper can be extended to a multiproduct case by introducing separate probabilities of sale for each product and learning the combined parameter vector.

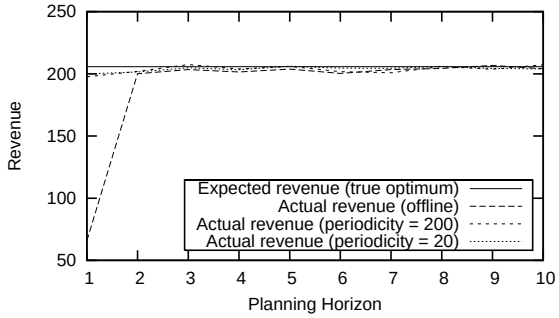

Fig. 1. Average of total revenues

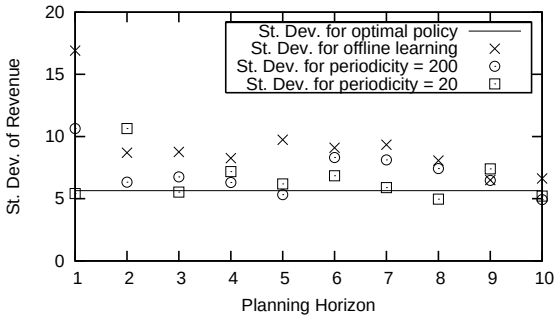

Fig. 2. Standard deviation of total revenues

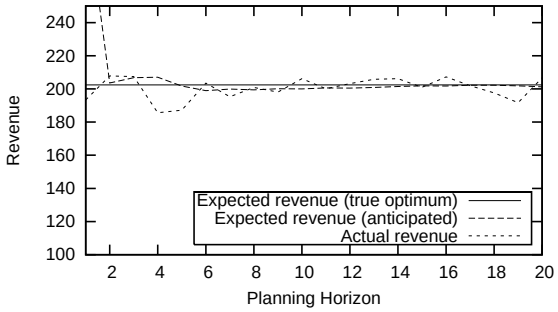

Fig. 3. The average performance for the general model over 10 simulations

REFERENCES

Aviv, Y. and A. Pazgal (2002). Pricing of short-cycle products through active learning. *Working paper, Washington University.*

Balvers, R. J. and T. F. Cosimano (1990). Actively learning about demand and the dynamics of price adjustment. *Economic Journal* **100**, 882–898.

Bertsimas, D. and G. Perakis (2003). Dynamic pricing: learning approach. *Working paper, MIT.*

Cover, T. M. (1991). Universal portfolios. *Math. Finance* **1**, 1–29.

Gallego, G. and G. J. van Ryzin (1994). Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Man. Sci.* **40**(8), 999–1020.

Levina, T. (2004). Using the aggregating algorithm for portfolio selection. PhD thesis. Rutgers, The State University of New Jersey.

Lin, K. (2005). Dynamic pricing with real-time demand learning. *EJOR.* Forthcoming.

Petruzzi, N. C. and M. Dada (2002). Dynamic pricing and inventory control with learning. *Naval Res. Logist.* **49**, 303–325.

Phillips, R. (2005). *Pricing and revenue optimization.* Stanford University Press.

Talluri, K. T. and G. J. van Ryzin (2004). *The Theory and Practice of Revenue Management.* Kluwer.

Vovk, V. (1990). Aggregating strategies. *Proceedings of the third Annual Workshop on Computational Learning Theory* pp. 371–383. San Mateo, CA. Morgan Kaufmann.

Vovk, V. (1999). Derandomizing stochastic prediction strategies. *Mach. Learn.* **35**, 247–282.

Zhao, W. and Y. Zheng (2000). Optimal dynamic pricing for perishable assets with nonhomogeneous demand. *Man. Sci.* **46**(3), 375–388.