Real-time dynamic pricing in a non-stationary environment using model-free reinforcement learning.

RANA, R. and OLIVEIRA, F.S.

2014

Real-Time Dynamic Pricing in a Non-Stationary Environment using Model-Free Reinforcement Learning

Abstract: This article examines the problem of establishing a pricing policy that maximizes the revenue for selling a given inventory by a fixed deadline. This problem is faced by a variety of industries, including airlines, hotels and fashion. Reinforcement learning algorithms are used to analyze how firms can both learn and optimize their pricing strategies while interacting with their customers. We show that by using reinforcement learning we can model the problem with inter-dependent demands. This type of model can be useful in producing a more accurate pricing scheme of services or products when important events affect consumer preferences. This paper proposes a methodology to optimize revenue in a model-free environment in which demand is learned and pricing decisions are updated in real-time. We compare the performance of the learning algorithms using Monte-Carlo simulation.

Keywords: Revenue Management, Dynamic pricing, Reinforcement learning, Simulation.

1 Introduction

Dynamic pricing is a business strategy that adjusts the product price in a timely fashion in order to allocate the right service, to the right customer, at the right time [1]. It is usually applied when there are uncertainties and seasonality of demand and supply, in an attempt to increase revenue. In particular, with the use of dynamic pricing over the internet the seller is informed about the level of demand in real-time and can price items using the optimal policies computed using historical data. In many industries managers face the problem of establishing a pricing policy that maximizes the

revenue from selling a given inventory of items by a fixed deadline. The common characteristics of these industries are that the full inventory of products is available for sale at the beginning of the selling period, no re-ordering is allowed, and the unsold products that remain by the deadline have a zero constant salvage value [2]. This paper is concerned with using reinforcement learning to solve the tactical problem of dynamically pricing these products to maximize the total expected revenue.

Examples of industries that must use dynamic pricing strategies are manufactured goods and services [3,4]. The first category includes goods with limited shelf-life, such as food items, electronic goods or fashion garments, which are usually sold in a finite time-frame, meaning that there is a deadline by which they must be removed from the store. The second category includes the service industries where the service will not generate revenue once the time window for availability has passed, such as airlines, hotels, conference or party facilities, cruise ship holidays, and tickets for trains, theatres, concerts, cinemas and stadiums.

The single product pricing problem addressed in this article, and originally studied by Gallego and van Ryzin [5], is important as it represents the issues faced in several industries and it is a good test framework for methodological contributions, see [6-15]. The most essential consideration when developing such a pricing policy is demand forecast accuracy. There are two major sources of randomness in demand: the customer arrival rate and customer reservation price. Most academic studies in revenue management assume that the functional relationship between arrival rate and price is known to the decision-maker. This assumption makes the problem more manageable and offers qualitative insights but it is unlikely to provide an optimal policy. In practice it is very rare that the decision-maker has full knowledge of the demand function. For this reason, to impose a structural form on the demand function can lead to model misspecification, resulting in revenue loss.

The main objective of this paper is to propose a model-free approach whereby the transition probabilities between states (i.e., the demand behaviour) are not characterized by a particular distribution. Reinforcement learning techniques, such as Q-learning and Q-learning with eligibility trace Q(λ), are applied to solve the problem of optimal dynamic pricing of perishable products when demand is stochastic with unknown characteristics. The contribution of this paper is to propose

a tested computational method (i.e., reinforcement learning) to solve the revenue management problem when information is incomplete and demand is non-stationary. In this article we use two popular methods, Q-learning [16] and the Q-learning with eligibility traces, originally proposed by Peng and Williams [17] .

When selling products or services with well known statistical distributions the information about one product can be used to update the demand for similar products. For example, one may consider all flights for a particular origin-destination pair and a specific departure time each week. Since booking can start half a year in advance, or even earlier, this provides simultaneous learning opportunities for 26 or more concurrent episodes, shifted by 1 week relatively to each other. This can similarly be applied to hotel rooms, cabins on cruise liners, and cars at rental agencies. In all these services there are opportunities to book in advance, they are perishable, and there is a window for learning from the same service, from one time period to the next. For example, when pricing hotel rooms, the behaviour of demand for a given room type, on a Monday, can be used to price for the same room type the following Monday.

However, when a new product is launched, the demand patterns may be very different from past ones. In this case reinforcement learning will be even more helpful. If iPad 2 is launched and the demand profile is different from iPad, for the same time period, this difference is used by the algorithm to update the expectations about future demand for the product, implicitly, without having to explicitly compute a demand forecast.

The paper is organized as follows. Section 2 presents a literature review and places the contribution of this paper in comparison to the literature. Section 3 discusses how the model is formulated and describes how reinforcement learning is used to solve the dynamic pricing problem. Section 4 presents the analytical results, showing that Q-learning with eligibility converges to the optimal policy, and that the rate of learning is faster than with the simple Q-learning. Section 5 provides numerical results, and qualitative insights into the advantages of model-free reinforcement learning and compares the Q and Q(λ) learning algorithms.

2 Related Literature

The two main research areas relevant to this study are dynamic pricing and reinforcement learning, each of which is addressed in turn, together with a discussion of the contribution in this article, where appropriate.

The majority of papers that address the problem of demand learning or estimation of pricing in the context of a single product, do so by assuming that one or more of the demand parameters are unknown. Recent examples include [1, 3, 18, 19], who have incorporated real-time demand information in their models. Anjos et al. [18] develop a general methodology for implementing a pricing policy, and describe how the policy can be updated in real-time to react to changes in the predicted purchase patterns of consumers. Lin [1] forecast customer arrival rates in real-time using Bayesian statistics. Aviv and Pazgal [19] and Lin [1] assumed a known reservation price distribution. Berk et al. [20] investigate pricing of perishable products in menu costs recognizing that, although demand autocorrelation within the selling is important, it is often ignored in revenue management literature. They highlight that the complexity of modelling demand in an environment in which autocorrelation is a concern. In this case the Q-learning with eligibility traces algorithm has the merit of learning the pricing policy in a model free-environment and hence has the ability to implicitly incorporate autocorrelation of demand information within its policy. Zhao et al. [21] study a dynamic pricing problem for perishable goods to consumers who may exhibit inertia. They formulate this problem using the finite-horizon dynamic programming approach and derive an optimal dynamic pricing policy. Daso and Tong [22] have modelled the pricing of perishable products considering strategic buyers. Banerjee and Turner [23] have developed a model for pricing perishable goods based on differential equations, which is able to deal with group arrivals and continuous prices. Li et al. [24] have modelled dynamic pricing of perishable products with stochastic demand which they represented using randomness and fuzziness.

In most studies, there a few underlying assumptions in the model-based approaches, as in the case of the Bayesian learning. In this case, when the conditions of the model are violated, the policy is non-optimal. Lim and Shanthikumar [25] presented an approach for the single product dynamic pricing problem that accounts for errors in the underlying model at the optimization stage. They emphasized the importance of the underlying assumptions about the demand-rate

when computing the optimal pricing policies. Besbes and Zeevi [26] developed nonparametric approaches that learn demand in a model free environment. However, there is a major issue with nonparametric approaches; the loss of tractability.

The reinforcement algorithms used in this article produce look-up tables of value functions of all state-action pairs where the state is characterized as the capacity level and time until expiration and the action is the price. The value functions of a state-action pair are calculated by the immediate revenue gained and the expected future revenues. Using these algorithms allows the initialization of the value functions as the best estimated demand function and then observe the real-time demand and update the value functions. The decision-maker is then better informed than an agent that starts pricing assuming no prior knowledge. This is illustrated in the numerical experiments in Section 5.

There is limited literature in the area of revenue management using reinforcement learning to find an optimal pricing policy. Gosavi et al. [27] used reinforcement learning to develop a strategy for seating allocation and overbooking in order to maximize the average revenue gained by an airline. In particular, Raju et al. [28] used a reinforcement learning (Q-learning) algorithm to price products dynamically with customer segmentation. They considered an infinite horizon learning problem where there is no deadline for the sale of stock, and price changes according to queue length and time. Carvalho and Puterman [29] have also investigated dynamic pricing and reinforcement learning by studying maximization and learning problems in finite horizons for unlimited product quantities. They have focused on specific parametric forms of customer arrival distribution and on the probability of sales; the parameters are assumed to be fixed and unknown.

Cheng [30] applied the Q-learning approach to dynamic pricing in e-retailing. Cheng acknowledges that Lin's [1] approach to adjust price in response to changes in demand may not be plausible due to the computational complexity of using dynamic programming. Cheng has characterized demand in the same way as Lin except that the parameters of the model are learned using reinforcement learning. Price updates are made in real-time as the Q-learning algorithm produces a look-up table and, therefore, value function updates can be made with ease. Cheng focused on the computational advantages of using reinforcement learning and was not concerned with the accurate representation of demand.

This article differentiates itself from existing literature in dynamic pricing as it uses reinforcement learning in the context of pricing perishable products with non-stationary selling demand.

3 Model formulation of the Dynamic Pricing problem

3.1 Markov Decision Process

In this article, the dynamic pricing problem of a perishable service is modeled as a discrete finite horizon Markov decision process (MDP). The dynamic pricing problem is formulated as a MDP because pricing is a real-time decision-making problem in a stochastic environment. This article aims to approximate a pricing policy that maximizes the revenue for selling a given inventory of products by a fixed deadline.

In practice, as it is difficult to apply a continuous price change, we use periodic price reviews, where prices are only allowed to change at specific time-points. Here, a general model for a fixed number of identical products or services is presented. The price, at any given time period, is determined by the remaining capacity and the time (e.g., number of days) left before the deadline expires. Let n be the total capacity of seats to sell and m be the total number of times intervals, i.e., the number of times the price is reviewed and can be changed. The key components of the MDP are:

• The state space: $x \in X = \{0, 1, .., n\}$. This represents the remaining capacity.

• Time horizon $t \in T = \{0, 1, .., m\}$ is the set of finite discrete times at which pricing actions are executed, where m is the last selling period before the service expires.

• x_t represents the remaining capacity (state of the system) at time t.

• $A(x_t)$ denotes the set of prices the seller can choose to set when there is x remaining capacity at time t, $a_t \in A(x_t)$ is a price for the capacity at time t.

• Transition Probabilities: $p_t(x_{t+1}|x_t, a_t)$ is the probability of having x remaining capacity at time $t + 1$, given that there is x remaining capacity at time t and price a is set.

• R is the revenue function defining, for each decision step, at every state and action, a real number $r(x_t, a_t)$ for $x_t \in X$, $t \in T$ and $a_t \in A(x_t)$ which specifies the expected immediate revenue gained for executing price a when there is x capacity remaining at time t.

The objective is to maximize the total expected revenue presented in Eq. (1), in which $x \in X$ and E_π is the expected value given the policy π.

$$V^\pi(x_1) = E_\pi[r(x_1, a_1) + r(x_2, a_2) + ... + r(x_m, a_m)|x_1] \tag{1}$$

A policy is a function $\pi : x \times t \rightarrow a_t$ specifying the price that should be set given the remaining capacity and time.

3.2 Adapting Reinforcement Learning to Dynamic Pricing

Reinforcement learning originated in the cybernetics, psychology, neuroscience, and computer science disciplines and has ever since attracted increasing interest in artificial intelligence and machine learning [31].

Reinforcement learning is an approach to sequential decision making in an unknown environment based learning from past experience. Reinforcement learning algorithms apply directly to the agent's experience, changing the policy in real-time. The first advantage of using reinforcement learning is that it does not require a pre-specified model of the environment on which to base the action selections. Instead, relationship between states, actions and rewards are learned through dynamic interaction with the environment. The second advantage is that is adaptive in the sense that it is capable of responding to a dynamically changing environment through ongoing learning and adaption.

The Q-learning and Q(λ) algorithms have been proposed to approximately solve large scale MDP problems. In the MDP framework: $v_t^\pi(x_t)$ denotes the expected total reward when starting at state x_t and following a policy π (i.e., $a_t = \pi(x_t)$ where $\pi(s)$ denotes the action chosen in state s when policy π is pursued). $Q_t^\pi(x_t, a_t)$ denotes the discounted expected total reward when starting at state x_t, taking action a_t and following policy π. That is Q_t^π is the state-action value function for policy π at time t. Eq. (2) represents the relationship between $Q_t^\pi(x_t, a_t)$ and V_t^π,

$$Q_t^\pi(x_t, a_t) = \sum_{x_{t+1} \in X} P_t(x_{t+1}|x_t, a_t)[r(x_t, a_t, x_{t+1}) + \eta V_{t+1}^\pi(x_{t+1})] \tag{2}$$

where η is the discount factor, $0 < \eta < 1$. In terms of the Bellman optimality function, Eq. (3) holds for arbitrary $x_t \in X$, where $Q_t^*(x_t, a_t)$ is the optimal value function for each state-action pair. For $t = 1, 2, ..., m$.

$$Q_t^*(x_t, a_t) = \sum_{x_{t+1} \in X} P_t(x_{t+1}|x_t, a_t)[r(x_t, a_t, x_{t+1}) + \eta \max_{a_{t+1} \in A(x_{t+1})} Q_{t+1}^*(x_{t+1}, a_{t+1})] \tag{3}$$

In the Q-learning and Q(λ) paradigms the decision-maker interacts with the environment through executing a set of actions. The environment is then modified and the agent perceives a new state, and a reward signal, at each time-point. In this process learning takes place through trial and error in a dynamic environment.

Over the course of the learning process, the Q-values of every state-action pair, $Q_t(x_t, a_t)$, are stored and updated. A Q-value represents the usefulness of executing a pricing action a_t when the environment is in a state x_t. This paper considers the dynamic pricing model of identical items over a finite selling horizon allowing the value of the state-action pairs, from one selling horizon to the next, to be learned. Let k denote the selling horizon and each selling horizon be split into m time intervals. The episode (selling horizon) refers to multiple instances of the dynamic pricing problem in consecutive time horizons and the transition probabilities are the same for different episodes (this makes them stationary across different episodes) but non-stationary within each episode. The algorithms consist of updating Q_t^k, the Q-values at every selling horizon k, for time t, which is a representation the estimation of Q_t^*, the optimal Q-values, from the current observed transitions and reward $< x_t^k,\ a_t^k,\ x_{t+1}^k,\ r_t^k >$, where $x_t^k,\ a_t^k, r_t^k$ are the remaining capacity, price action, current observed reward at time t, in episode k, respectively and x_{t+1}^k is the new remaining capacity at time $t + 1$. Note that r_t^k depends on x_t^k, a_t^k and x_{t+1}^k : $r_t^k \equiv r(x_t^k, a_t^k, x_{t+1}^k)$.

The Q-learning approach updating rule is represented by Eq.(4), where $\alpha(x_t^k, a_t^k)$ is the learning rate and $0 < \eta < 1$. For $t = 1, ..., m$

$$Q_t^{k+1}(x_t^k, a_t^k) = (1 - \alpha(x_t^k, a_t^k))Q_t^k(x_t^k, a_t^k) + \alpha(x_t^k, a_t^k)(r_t^k + \eta \max_{a_{t+1} \in A(x_{t+1})} Q_{t+1}^k(x_{t+1}^k, a_{t+1})) \tag{4}$$

The $Q(\lambda)$ algorithm is an extension to the one-step Q-learning algorithm. The Q-value estimates of the value of all state-action pairs are updated in proportion to their eligibility. The idea

behind the eligibilities is very simple: each time a state-action pair is selected within an episode a short-term memory is assigned (known as a trace) which decays as the sales horizon moves forward. The magnitude of the trace determines the eligibility of a state-action pair for learning; a state-action pair visited more recently has a larger eligibility. The use of eligibility traces increases the ability of a learning system to solve the temporal-credit assignment problem, i.e., to calculate how to punish or reward a state-action choice, when it may have far reaching effects. Additionally, reinforcement learning with eligibility traces has the important ability of learning in non-stationary selling horizons, performing particularly well in situations in which the demand between successive times (e.g., days) exhibits autocorrelation as it can incorporate hidden states into its decision making process.

The transition information available to the decision-maker, at episode k, at time t, is $< x_t^k,\, a_t^k,\, x_{t+1}^k,\, r_t^k >$. The eligibility trace function, at episode k, at time t, is denoted by e_t^k. On experiencing transition $< x_t^k,\, a_t^k,\, x_{t+1}^k,\, r_t^k >$ the following updates are performed to the eligibility traces:
$e_t^k(x_t^k, a_t^k) = 1$,
$\forall\ i < t$, $e_t^k(x_i, a_i) = \lambda e_{t-1}^k(x_i, a_i)$ if $Q_t^k(x_t^k, a_t^k) = \max_{a_t \in A(x_t)} Q_t^k(x_t^k, a_t)$,
otherwise $e_t^k(x_i, a_i) = 0$.

For all state-action pairs, the eligibility trace decays at a rate λ , except for the last state-action visited where the eligibility trace is incremented by one unit. The traces can be updated in two ways. If a greedy selection is made, then all the traces of the state-action pairs visited in the episode decay at a parameter λ . The decay parameter determines how different state-action pairs are assigned a certain prediction error. If an exploration action was taken, then the eligibility traces are set to zero.

At episode k, for each time t, the estimation error is equal to
$\delta_t^k = r_t^k + \eta \max_{a_{t+1}} Q_t^k(x_{t+1}^k, a_{t+1}) - Q_t^k(x_t^k, a_t^k)$, and it is assigned to each prior state-action pair visited, in episode k, according to their eligibility trace. The error is computed at each time t, and its value, and the value of the state-action pairs previously visited in that episode, are updated. Earlier visited state-action pairs are given less credit for the current error. The pricing action many time-steps away take less credit, or blame, for the error at time t. Learning accelerates because of the use of eligibility traces. The eligibility trace methods can strengthen the whole

sequence of pricing actions. The objective of the learning strategy to teach the decision-maker the optimal pricing policy. The Q(λ) algorithm computes the optimal value function iteratively: for each state x at every time t, the optimal value $Q_t^*(x_t, a_t)$ of each action a_t is estimated on the basis of simulated transitions. When all these values have been estimated correctly the optimal policy can be derived through through Eq.(5).

$$\forall t \leq m,\ \forall x \in \mathbf{X},\ \pi_t^*(x_t) = \max_{a_t \in A(x_t)} Q_t^*(x_t, a_t). \tag{5}$$

Before discussing the $Q(\lambda)$ algorithm in procedural form (presented in Table 1), two important elements must be introduced: the exploration rate and the learning rate. The absence of perfect prior information concerning the demand model introduces a new component into the dynamic optimization problem, the trade-off between exploration (attempting non-optimal decisions in order to improve the current policy) and exploitation (choosing the best policy so far in order to maximize the expected profit). The longer one spends learning the demand, the less time is spent exploiting prices to increase revenue.

The objective of the algorithm is to obtain an accurate estimate of the optimal policy based on observations during the exploration phase while, at the same time keeping the exploration rate small, in order to limit revenue loss over this learning phase. The exploration rate (ϵ) chosen is an ϵ-greedy policy at a rate $1/k$ so that learning progresses as the exploration rate decreases. The result of this assumption is that, as the decision-maker gains more knowledge, sub-optimal prices are explored less often. The learning rate is set in a similar manner and also decreases with time. The learning rate for each state-action pair is denoted by $\alpha(x_t^k, a_t^k)$. The learning rate $\alpha(x_t^k, a_t^k)$ is equal to $1/n^k(x_t, a_t)$, where $n^k(x_t, a_t)$ is the equal to 1 plus the number of times the state-action pair (x_t, a_t) was visited by the process (x_t^k, a_t^k) before time k.

The algorithm starts by initializing the Q-value and the following steps are repeated for each episode k. All the eligibility traces for all state-action pairs are set to zero. The starting state, x_t^k (the total number of inventory) and a_t^k (the price for the product) are chosen. After choosing this price the next state x_{t+1}^k(remaining capacity) and the immediate revenue gained, r_t^k are observed. The price a_{t+1}^k in the next state is chosen using the (non-greedy) policy π. The temporal difference error (δ) is calculated and the eligibility trace $e_t^k(x_t^k, a_t^k)$ is updated to 1. Then all the Q-values of

Table 1: Q(λ) An online policy TD dynamic pricing algorithm for products in finite selling horizon

Initialise $k = 1$ and $\forall\ t\ Q_t^1(x_t^1, a_t^1)$;
Repeat for each episode $k \to \infty$
Step 1: All traces are set to zero at the beginning of the sales horizon
$\forall\ t,\ x_t,\ a_t,\ e_t^k(x_t, a_t) = 0$;
Step 2: We initialize the initial capacities and price set $x_t^k,\ a_t^k$
Step 3: Repeat $t = 1$ to m (for each decision step in the selling horizon)
Step 3.1: Take price a_t^k, observe the reward r_t^k and remaining capacities x_{t+1}^k
Step 3.2: We choose a price a_{t+1}^k for state x_{t+1}^k using ϵ-policy π
Greedy price $a_{t+1}^* \leftarrow argmax_{a_{t+1}} Q_{t+1}^k(x_{t+1}^k, a_{t+1})$
Step 3.3: TD error is $\delta_t^k \leftarrow r_t^k + \eta Q_{t+1}^k(x_{t+1}^k, a_{t+1}^*) - Q_t^k(x_t^k, a_t^k)$
Step 4.: Update the (x_t^k, a_t^k) pair's trace $e^k(x_t^k, a_t^k) \leftarrow 1$
Step 4.1: $\forall\ i \leq t \in T,\ \forall\ x_i,\ a_i$ Update all Q-values according to their eligibility traces
Step 4.2: $Q_t^{k+1}(x_i, a_i) \leftarrow Q_i^k(x_i, a_i) + \alpha(x_i^k, a_i^k)\delta_t^k e_t^k(x_i, a_i)$
Step 4.3: If $a_{t+1}^k = a_{t+1}^*$, then $e_{t+1}^k(x_i, a_i) \leftarrow \lambda e_t^k(x_i, a_i)$
else $e_{t+1}^k(x_i, a_i) \leftarrow 0$
Step 5: $x_t^k \leftarrow x_{t+1}^k,\ a_t^k \leftarrow a_{t+1}^k,\ k \leftarrow k + 1$

the state-action pairs, are updated using their eligibility traces. All eligibility traces are updated: if the price a_{t+1}^k is a greedy (optimal so far) action then all traces are multiplied by a parameter λ; if a_{t+1}^k is an exploration action then all traces are set to zero. Then the next state x_{t+1}^k at time $t+1$ and action a_{t+1}^k becomes the new x_t^k and a_t^k. The process is repeated until the last time m, for each episode.

4 Analytical Results

In this section it will be proven that the $Q(\lambda)$ algorithm converges when demand is non-stationary within a selling horizon. Additionally it will be proven that the $Q(\lambda)$ algorithm has a faster learning rate than the simple Q-learning algorithm and therefore converges faster. A general proof of convergence for the Q-learning algorithm is described in Szepesvri and Littman [32]. In this article we prove the convergence of $Q(\lambda)$ in a non-stationary environment.

A non-stationary MDP has different state-spaces, actions, transition probabilities and reward values at each time-step. The actions taken at each state are time-dependent. Consider a non-stationary discounted finite MDP with a discount factor $0 < \eta < 1$. At episode k a pricing decision is made at each time $t \in T = (1, 2, ..., m)$. Each time-step is associated in a finite state space

X and a finite set of pricing actions $A(x_t)$. At episode k, for each time $t \in T$, we are given a four-tuple $< x_t^k, a_t^k, x_{t+1}^k, r_t^k >$.

Assumption 4.1 (Sampling Assumptions) *Consider a non-stationary finite MDP, where $p_t(x_{t+1}|x_t, a_t)$ are the transition probabilities and $r(x_t, a_t)$ are the immediate expected rewards. Let $< x_t^k, a_t^k, x_t^k, r_t^k >$ be a fixed stochastic process, for each time t, and let F^k be an increasing sequence of σ-fields (the history space) for which $< x_t^k, a_t^k, x_{t-1}^k, r_{t-1}^k, ..., x_0^0 >$ are measurable. Then assume that the following holds for all $t \in T$:*

1. $p_t(x_{t+1}^k = x_{t+1}|x_t = x_t^k, a_t = a_t^k, F^k) = p_t(x_{t+1}|x_t, a_t)$.

2. $E[r_t^k|x_t = x_t^k, a_t = a_t^k, F^k] = r(x_t, a_t)$ and $var[r_t^k|x_t^k, a_t^k, F^k]$ is bounded independently of k.

The Q(λ) updating is represented by Eq. (6), where $0 < \eta < 1$. At time t for every state and action visited in episode k. Let j_i denote the time where the last greedy action was chosen for time i, i.e., the last time where a greedy action was taken and $Q_{m+1}(x_{m+1}, a_{m+1}) = 0$. For $i = 1, .., t.$,

$$Q_t^{k+1}(x_i^k, a_i^k) = Q_t^k(x_i^k, a_i^k) + \sum_{t'=i}^{j_i} \lambda^{t'-i} \alpha(x_{t'}^k, a_{t'}^k)[r_{t'}^k + \eta \max_{a_{t'+1}} Q_{t'+1}^k(x_{t'+1}^k, a_{t'+1}) - Q_{t'}^k(x_{t'}^k, a_{t'}^k)]. \tag{6}$$

Eq. (6) represents an off-line updating algorithm meaning that all state-action pairs visited before and including time t, are updated at the same time. The Q-values are updated according to the temporal difference error (Step 3.3 in Table 1) and their eligibility traces (Step 4.3) represented as $\lambda^{t'-i}$ in Eq.(6). Theorem 4.1 proves that when using Q(λ) in a non-stationary selling horizon, the Q-values converge. This is an extension of a theorem by Szepesvari and Littman (1999), who proved that when using the Q-learning algorithm the Q-values converge.

Theorem 4.1 *Consider Q(λ) in a non-stationary finite MDP where the sequence $< x_t^k, a_t^k, x_t^k, r_t^k >$ satisfies assumption 4.1. Assume that the learning rate sequence $\alpha(x_t^k, a_t^k)$ satisfies the following:*

1. $0 \leq \alpha(x_t^k, a_t^k)$, $\sum_{k=0}^{\infty} \alpha(x_t^k, a_t^k) = \infty$, $\sum_{k=0}^{\infty} (\alpha(x_t^k, a_t^k))^2 < \infty$ and both hold uniformly and w.p.1 and

2. $\alpha(x_t, a_t) = 0$ if $\alpha(x_t, a_t) \neq \alpha(x_t^k, a_t^k)$ w.p.1. Then the values defined by Eq. 6 converge.

This theorem shows that the $Q(\lambda)$ in a Markovian environment converges if there are a sufficient number of episodes. It will now be proven that the Q(λ) in a non-stationary finite horizon converges more rapidly than the standard Q-learning algorithm, Theorem 4.2.
In order to compare the two methods we start by describing the way the respective Q-values are updated in Eq. (7) and (8), where $Q_{m+1}(x_{m+1}, a_{m+1}) = 0$.

$$Q_t'^{k+1}(x_t^k, a_t^k) = (1 - \alpha(x_t^k, a_t^k))Q_t^k(x_t^k, a_t^k) + \alpha(x_t^k, a_t^k)(r_t^k + \eta \max_{a_{t+1}} Q_{t+1}^k(x_{t+1}^k, a_{t+1})) \tag{7}$$

On the other hand the Q(λ) updates the Q-values at each iteration. Where j_i denotes the period where the last greedy action was chosen for time i and where $Q_{m+1}(x_{m+1}, a_{m+1}) = 0$. For $i = 1, .., t.$,

$$\begin{aligned} Q_t''^{k+1}(x_i^k, a_i^k) = Q_t^k(x_i^k, a_i^k) + \sum_{t'=i}^{j_i} \lambda^{t'-i}\alpha(x_{t'}^k, a_{t'}^k)[r_{t'}^k + \eta \max_{a_{t'+1}} Q_{t'+1}^k(x_{t'+1}^k, a_{t'+1}) \\ - Q_t^k(x_{t'}^k, a_{t'}^k)]. \end{aligned} \tag{8}$$

The eligibility traces are used to speed up the learning process by tracking visited state-action pairs and adding a portion of the reward received from each decision to each state-action pair visited in an episode. Instead of updating a state-action pair at each iteration, as the Q-learning algorithm does, the Q-learning with eligibility traces updates the Q-values of all pairs with eligibilities different from zero, allowing the rewards to be carried over several state-action pairs. Let Q^+ be a fixed point.

Theorem 4.2 *For every natural number k, $\forall x_t \in X$, $a \in A(x_t)$ and $\forall\ t = 1, 2, .., m$,*
$|Q_t'^k(x_t, a_t) - Q_t^+(x_t, a_t)| \geq |Q_t''^k(x_t, a_t) - Q_t^+(x_t, a_t)|$.

Theorem 4.2 proves that the Q(λ) strictly dominates Q-learning and therefore the policies produced by Q(λ), in the learning process, lead to higher revenues.

5 Numerical Results and Qualitative Insights

5.1 Model-Free *versus* Parameterized Structure

The model-free approach is now compared with a parametric learning algorithm. To begin with a one step process is considered so that the Q-learning and Q(λ) perform with the same efficiency. The performance of three algorithms is explored: (i) The Q-learning algorithm, (ii) the Q-learning algorithm where Q-values are set to the best estimated demand function and (iii) a learning algorithm where the parametric structure is known but the parameter values (the tuning parameters) are unknown [26].

The performance of these algorithms is measured by the percentage deviation from the optimal policy. The optimal policy is computed with full information using dynamic programming. Our results are based on running 10^3 independent simulation replications from which we averaged the revenues. The discount factor used in all the numerical results was $\eta = 0.999$, because as η tends to 1 the reinforcement learning algorithms in (6) and (7) converge to the optimal policy [33].

Table 2 summarizes results for the two underlying models, the exponential demand model $\lambda(a) = \theta exp(-l_2 a)$; $\theta = l_1 exp(1)$ and the linear demand model $\lambda(a) = h_1 - h_2 a$, using different parameter values. Here a denotes the pricing action.

The Q-learning algorithm does not make any assumptions about the structure of the demand function (beyond the assumptions made in Section 4). The Q-learning with the best estimated (BE) demand function algorithm (Q-learning BE) initializes the Q-values to a demand function that the decision-maker believes to be sensible. For the exponential demand model it initialized the Q-values using the demand function $\lambda(a) = 15exp(-0.5a)$ and for the linear model using $\lambda(a) = 30 - 5a$. The demand function (linear or exponential) is an input parameter to generate simulation data, the Q-learning algorithm and Q-learning algorithm with best estimated demand function do not know the demand function but, use the simulated data to guess the best price at each point of time in the horizon. The parametric algorithm knows the structure of the demand function, e.g., linear but does not know the parameter values, i.e., l_1 and l_2 but uses the simulation data to estimate the parameter values.

The Q-learning BE algorithm learns in the same manner as the Q-learning algorithm (as seen

in Table 1), with the only difference being that its Q-values are initialised to an estimated demand function as opposed to zero's with the Q-learning algorithm. The Q-learning BE algorithm assumes some prior knowledge of demand and will therefore initially selects actions less randomly in comparison to the Q-learning algorithm. The Q-learning and Q-learning BE algorithm converge to the same policy.

In all cases, the initial capacity level was $x = 20$, the selling horizon was $T = 1$ and the feasible price set was $\{0.1, 10\}$.

Table 2: Performance of algorithms measured as a percentage of the optimal policy for different demand functions

Demand	Q-learning	Q-learning with BE	Parametric
Exponential			
$l_1 = 10$, $l_2 = 0.5$	93.2	98.1	97.1
$l_1 = 15$, $l_2 = 1$	94.3	97.2	97.5
$l_1 = 20$, $l_2 = 0.75$	93	96.3	98.2
$l_1 = 25$, $l_2 = 3$	94.2	92.9	98.0
$l_1 = 30$, $l_2 = 0.5$	94.5	98.1	97.8
Linear			
$h_1 = 50$, $h_2 = 4$	94.2	95.1	97.6
$h_1 = 35$, $h_2 = 2$	93.2	94.3	97.3
$h_1 = 30$, $h_2 = 3$	93.4	96.1	97.2
$h_1 = 20$, $h_2 = 2.5$	92.5	96	97.2
$h_1 = 15$, $h_2 = 1.5$	93.7	97.1	98

The less structure is assumed *a priori*, the higher the profit loss relative to the full information benchmark; this is the cost caused by increasing uncertainty with regard to the demand model. An additional fundamental difference between the various policies is the degree to which the price domain is explored.

The Q-learning algorithm essentially needs to explore the entire domain; the Q-learning with the best estimated function has some prior knowledge so initially smarter pricing decisions are made; the parametric policy with unknown parameters explores prices to ensure precision. When a single parameter is unknown, one can infer information about the parameter from observations of demand at a single price. After the first stage a^* can be estimated but we continue learning, to ensure the estimate of a^* becomes more precise.

It was illustrated in the previous experiment that more refined information regarding the de-

mand model yields higher revenues. However if a parametric structure of demand is postulated, which is incorrect relative to the true underlying demand model, there is a danger of model misspecification. This misspecification risk can be eliminated *via* reinforcement learning algorithms, but at the price of settling for more modest performance revenue-wise. An illustration of this trade-off is now given. The time horizon is fixed to be T = 1 and the set of feasible prices is fixed to be $\{0.1, 10\}$. Table 3 depicts the performance of the three algorithms for two underlying demand models for various parameter values. In the first set of experiments the parametric algorithm first assumes an exponential parametric structure for the demand function, $\lambda(a) = \theta exp(-l_2 a)$; $\theta = l_1 exp(1)$, but the underlying demand model is linear $\lambda(a) = (h_1 - h_2 a)$. In the second set of experiments the parametric algorithm is assumed to be linear but the underlying demand model is assumed by the parametric algorithm as exponential.

The Q-learning with the best estimated (BE) function algorithm (Q-learning BE) also initializes the Q-values assuming the same parametric structure (misspecified model) as the parametric algorithm. For the first set of experiments the BE demand function: $\lambda(a) = 15exp(-2a)$ and for the second set of experiments is $\lambda(a) = 30 - 5a$. The inventory $x = 20$ is used for all the experiments. The results depicted in the Table 3 are based on running 10^3 independent simulations from which the performance indicators were derived by averaging their revenues. The percentage deviation from the optimal policy for the three algorithms was calculated.

The purpose of this comparison is to illustrate how the model-free approaches are robust to modeling errors and assumptions. The parametric algorithm outperforms its model-free algorithm counterpart when the assumed parametric model is consistent with the true demand function; otherwise the parametric algorithm leads to a non-optimal policy. The parametric algorithm assumes a demand structure and learns the parameter values using the simulated data. If the assumed demand structure (for example, linear) is different from the actual demand structure (for example exponential) then the parametric algorithm does not learn the optimal policy.

If the model is misspecified, the performance of the parametric algorithm fails to achieve the full revenues asymptotically. The model-free learning approach eliminates the risk stemming from model misspecification but at a price of extracting lower revenues than its parametric counterpart. The compromise for this is to initialize the starting Q-values to the best estimated function and

Table 3: Performance of algorithms measured as a percentage of the optimal policy for different demand functions

True Demand	Q-learning	Q-learning with BE	Parametric (misspecified)
Linear			
$h_1 = 40$, $h_2 = 2.5$	92.9	93.2	82
$h_1 = 20$, $h_2 = 1.5$	94.2	94.9	82.9
$h_1 = 60$, $h_2 = 5$	94.3	94.2	80.0
$h_1 = 30$, $h_2 = 0.5$	92.7	93.1	81.1
$h_1 = 20$, $h_2 = 0.2$	94.5	94.6	82.1
Exponential			
$l_1 = 10$, $l_2 = 1$	93.2	93.1	64.3
$l_1 = 15$, $l_2 = 3$	92.0	94.2	63.1
$l_1 = 30$, $l_2 = 4$	90.2	93.2	63.2
$l_1 = 20$, $l_2 = 3$	93.2	94.2	65.2
$l_1 = 10$, $l_2 = 0.5$	92.1	93.1	62.1

then continue learning under the model-free approach. The difference then is that when a greedy action is taken, it is likely to yield more revenue than an action chosen by a model with no or very little information about demand.

5.2 Q-learning *versus* Q(λ)

Next, the performances of the Q-learning, Q-learning with BE, Q(λ), Q(λ) with BE, parametric (well defined) and parametric (misspecified) algorithms are compared. Consider a single-leg flight problem where the airline company does not know customer demand, i.e., it has no explicit model of customer buying behavior, and only observes realized demand after taking pricing action in different states. The demand data is generated using the following assumptions:

1. Time dependent Poisson Distribution. Customer arrival rate is a Poisson distribution with discrete times for the mean arrival $\mu(t)$. It is assumed that the mean arrival rate decreases at a linear rate $\mu(t)$ - $5t$. This assumption is stochastic and has an autocorrelation element of demand. The initial customer arrival rate is drawn randomly from a uniform distribution [b, c]. For these experiments, b=50 and c=100.

2. The customer's reservation price (how much the customer is willing to pay) for the service increases exponentially as the decision time-steps get closer to expiring. Customers who do not buy the product do not wait for a cheaper price. The time horizon is fixed to be a discrete set

$T = \{1, 10\}$ and the set of feasible prices is fixed to be $\{70, 120\}$. The stock is a discrete set $\{10, 100\}$.

The performance of these algorithms is measured by the percentage deviation from the optimal policy. The optimal policy is the policy under full information about demand. The results given here are based on running 10^3 independent simulation replications from which the average revenues were computed. The best estimated demand for the Q-learning and Q(λ) is defined by 75 - $5t * exp(-2/t * a/100)$, the parametric well-defined is defined as $\mu(t)$ - $\beta_1 t * exp(-\beta_2/t * a/100)$ and the parametric misspecified is $\mu(t)exp(-\beta_1) * exp(-\beta_2/t * a/100)$. Table 4 summarizes the performance of the algorithms.

Table 4: Performance of algorithms measured as a percentage of the optimal policy

Algorithm	
Q-learning	72.3
Q-learning with BE	74.4
Q(λ)	91.4
Q(λ) with BE	94.1
Parametric (well-defined)	97.4
Parametric (misspecified)	60.1

Next, the Q-learning and Q(λ) algorithms are examined in more detail. To compare the performance of the Q(λ) algorithm to the simple Q-learning, the total expected revenue generated by both algorithms together with their confidence intervals were computed for the entire selling horizon and initialized at a given state-action pair. The Q-values are compared for a capacity of 100 seats at a decision time where there are 10 days to departure and the unit price is 70. Figure 1 shows that the Q(λ) yields more revenue than the simple Q-learning algorithm.

The model based on the example described earlier for 2^*10^3 flights was run 10^3 times. Samples were generated for both the algorithms to test the significant difference between the total expected revenue for different numbers of iterations. The 95% confidence intervals were calculated for the total expected revenues for 50, 500, 1000 and 2000 episodes, respectively.

The convergence of the algorithm in Figure 1 starts with the decision maker assuming no prior knowledge of demand. In Figure 1, the decision-maker learns how to make over 60% of his profit within the first 50 episodes. This information regarding demand can be used to generate a demand function and if this demand function is correct an optimal policy can be established after a few

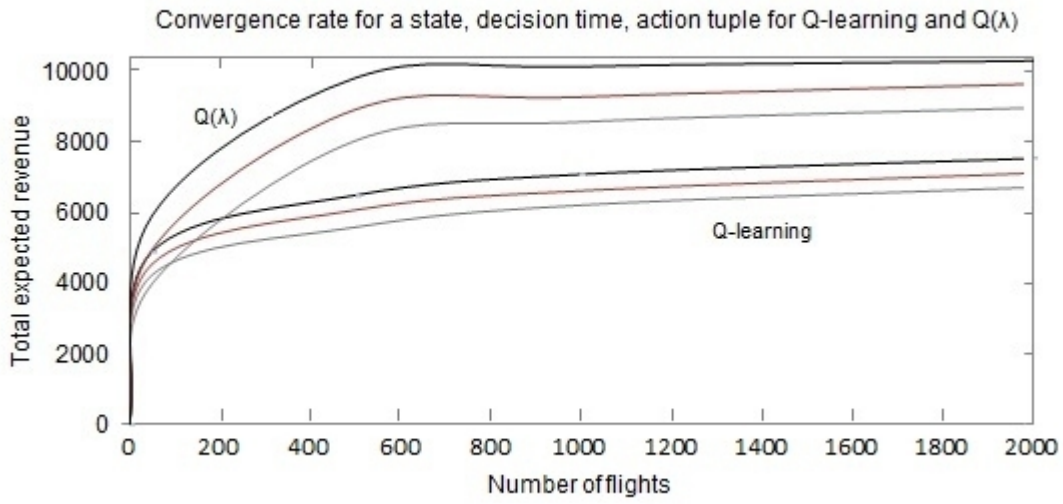

Figure 1: Comparison of the expected revenue from Q-learning and Q(λ) as a function of the number of episodes

more episodes. Two algorithms can be run simultaneously. The first would assume a demand function based on the information gained in the first 50 episodes, from which it can then learn the parameter values of the demand function thereafter. The second can learn the Q-values using the simulated data assuming no model. Therefore, by running both algorithms, the risk of miss-specifying the demand model is removed. In Table 5, LC and UC represent the lower and upper value of the confidence interval, respectively.

Table 5: The expected revenue at different iterations for the Q-learning and Q(λ) Algorithm

No Episodes	mean Q(λ)	Mean Q-learning	LC Q(λ)	UC Q(λ)	LC Q-learning	HC Q-learning
50	4966	4555	3997	5931	4362	4097
500	8934	6083	8050	9798	5750	6504
1000	9290	6631	8584	10050	6224	7097
2000	9660	7130	8985	10307	6731	7549

It can be deduced from Table 5 that for 500, 1000, and 2000 episodes the confidence intervals for the algorithms do not overlap, hence this indicates that there is a significant difference between the total expected revenue for the two algorithms. At iteration number 50 the algorithms produce similar results because the rate of exploration is higher in the beginning and it reduces as the number of iterations increases. The Q(λ) make updates in a similar manner to the Q-learning

traces when the exploration rate is high. The total expected revenue for Q(λ) is consistently higher. For this example, it is evident that the Q(λ) algorithm outperforms the simple Q-learning algorithm and, therefore, delivers higher expected revenue.

The sensitivity of the algorithms with respect to exploration levels was also analyzed. The sensitivity analysis was performed in order to clarify the empirical performance of Q(λ) and to compare it with the standard Q-learning algorithm. Figure 2 presents the performance of two exploration rates, 0.01 and 0.5. The plots show that both exploration rates produce generally similar results in terms of the total expected revenue gained at the Q-value with a capacity of 100, ten days to departure and a unit price of 70.

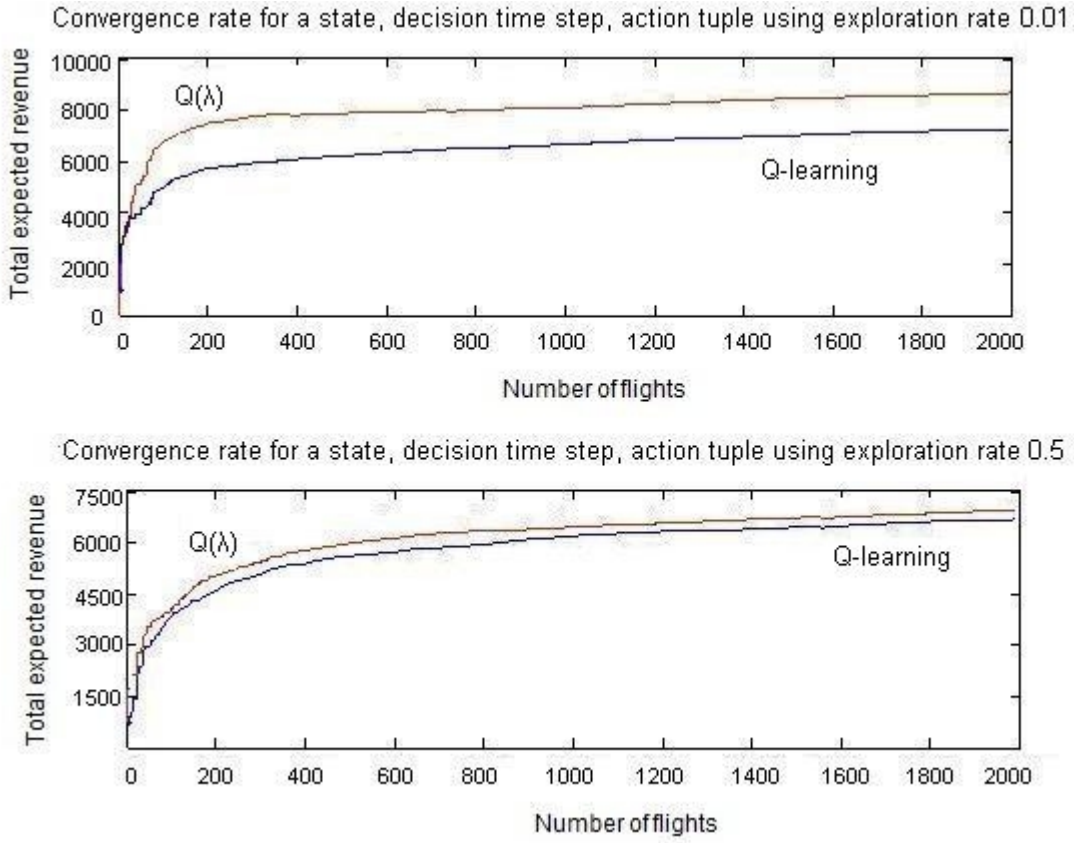

Figure 2: Comparison of the expected revenue from Q-learning and Q(λ) as a function of the number of flights

In Table 6 the total revenue for $2 * 10^3$ flights at different exploration rates are reported as a percentage. The maximum result is set to 100%. The results show that a lower exploration rate produces higher total expected revenue. Exploration is important because it helps reduce the likelihood of the decision-maker exercising a suboptimal policy. More exploration is not better in

Table 6: The percentage of the maximum received at different exploration rates.

Algorithm	1/k	0.01	0.05	0.1	0.5
Q-learning	91.28	91.28	91.28	90.09	82.38
Q(λ)	100	99.8	99.9	93.68	84.55

this case because the decision-maker is making less money when he exercises non-optimal pricing actions too often as in rate 0.5. The exploration rate $(1/k)$ produces the highest total revenue. It is not rational to repeatedly try non-optimal pricing actions and it may be in the decision-maker's best interest to reduce this rate of exploration as he becomes more knowledgeable with time.

Q(λ) produces better results when the rate of exploration is lower. This is because when the exploration rate is high it will perform updates in a similar manner to the Q-learning. The sensitivity analysis discussed above shows that the Q(λ) algorithm outperforms the standard Q-learning algorithm for all the exploration rates tested. The Q-learning algorithm, because of its temporal adjacent based credit assignment, may fail to learn the true expected revenue value of an action in a finite number of episodes. However, the Q(λ) algorithm is able to update estimates of the Q-values based more on rewards and less on estimates from successor states, and its learning rate is much faster. The trace mechanism provides an immediate propagation of the reinforcement signal information, assigning credit in various degrees to possibly many more than one state-action pair that is immediately affected in one-step Q-learning. The slow backwards propagation of credit that is associated with Q-learning is potentially accelerated. The Q-values are therefore more likely to converge to their true value within a finite number of episodes.

Another reason for the difference in total expected revenue between the Q-learning algorithm and the Q(λ) algorithm is that demand between successive days exhibits autocorrelation. The customer arrival rate is drawn randomly at the initial state from a uniform distribution. As time increases the number of customers arriving decreases exponentially. As the service approaches expiry, fewer customers arrive to purchase the service. Since the customer arrival rate at the start of the selling horizon is stochastic, the customer arrival rate at the subsequent time intervals is stochastic and autocorrelated. Hence, when a low customer arrival rate is drawn at the initial state, the optimal dynamic pricing policy will be different compared to when a higher initial customer rate is drawn. The Q-learning with eligibility traces algorithm is able to distinguish a pricing path

according to the demand at the previous stage (time), hence the autocorrelation between demand at each decision is captured and aids in finding the optimal policy.

Having compared the average performance of the Q-learning and Q(λ) in different settings, as reported in Tables 4, 5 and 6, and in Figures 1 and 2, we now analyze two very specific examples of the behavior of the algorithms in pricing different flights. With this purpose in Figure 3 we report the revenue for each one of the 2000 flights generated using monte-carlo simulation. Figure 3 allows us to observe an example of the raw data used to generate the Q-values reported in Table 5. The specific revenues reported for each one of the 2000 flights are stochastic, as the customer arrival rate is generated at random.

From Figure 3 we observe that the algorithms have converged to different pricing policies. In this case the autocorrelation of demand plays a role in searching for the optimal policy. Whereas the Q-learning algorithm managed to increase the average revenue by reducing the number of flights with low revenues (at the cost of peak revenue which never goes above 7500), but still have many flights with revenues below 5500; the Q(λ) algorithm was able to increase the average revenue by reduding the number of flights with low revenue (there is no flight with revenue below 5500) and at the same time keeping the policy that allows the company to receive large revenues at the peak times. This superior performance of the Q(λ) is justified by its ability to take into account the autocorrelation of demand in its pricing policies. This property is explained more clearly in the next example.

Q-learning is designed to work when the expected value of the pay-off from choosing a particular action from a particular state is independent of antecedent states. On the other hand the Q(λ) works well when the payoffs of sequential actions are autocorrelated. This idea is illustrated in the next example where there are two days left before the service expires.

Figure 4 illustrates different demand pathways. If the decision-maker is at state $x2$, (there are 60 seats remaining) two days before the flight departs and he sets the price at 100, the next day, he reviews his stock level and changes the price. He now has 40 seats remaining (state $x3$), is at the final decision step, one day before departure, he takes the pricing action $a3$ (150), and sells all the stock. However, the same pricing action $a3$ will yield zero revenue if the initial state was $x1$, and the pricing action $a1$. This is because the process follows a different demand path.

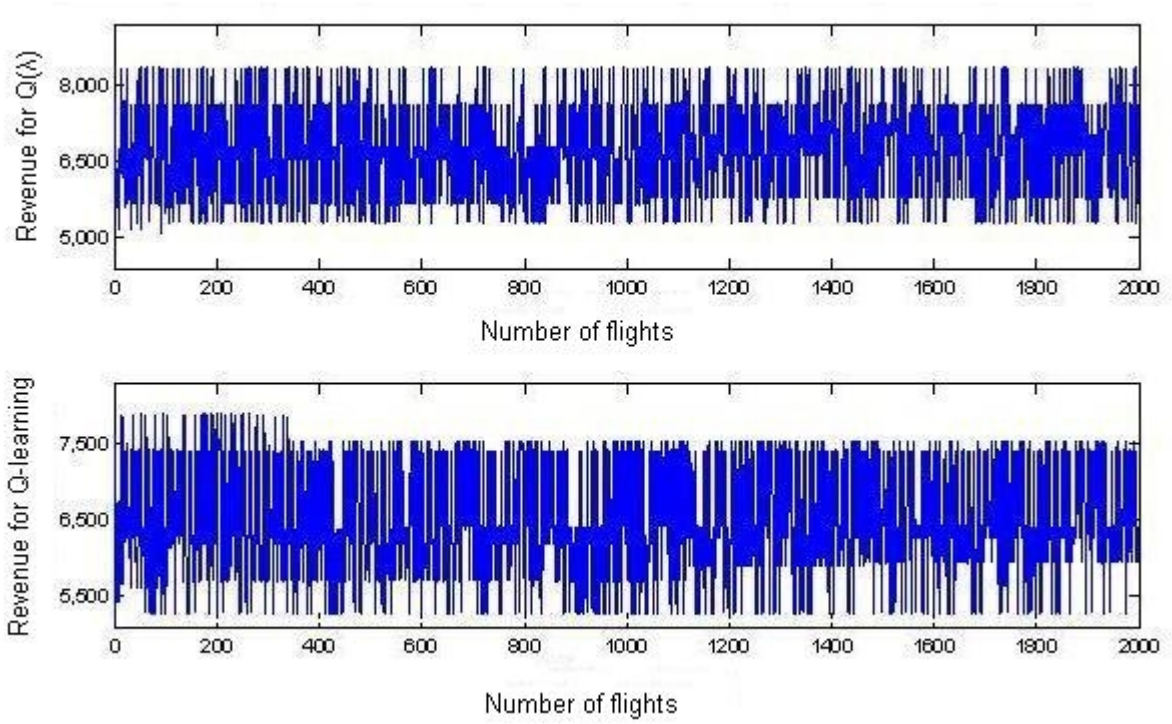

Figure 3: Revenue generated for 2000 flights for Q-learning and Q(λ) Algorithms

What we mean is that the demand for the service is different for different customer arrival rates and different antecedent pricing actions. The previously visited state, time, action (stock levels, days left before departure and pricing action) tuple encapsulate information, implicitly, about the future demand. If it is more likely that state $x3$ is reached from state $x2$ and pricing action $a2$ then the agent soon learns that action a3 is profitable in state $x3$. Now if the actions are credited and selected according to the updating rule based on the Q-values of the temporary states, as in Q-learning, then when the decision-maker encounters state $x1$ he will believe that action $a3$ is performing well and thus assigns undue credit to action, $a1$. If 100 simulation runs were conducted and from them 85 of the runs went from state $x2$ to state $x3$, then the Q-value using Q-learning for state $x1$, two days before departure at price 120 would be approximately 5100 and the Q-learning with eligibility traces will give the value to be approximately zero, its true value.

Alternatively, if state $x1$ and $x2$ are equally likely to occur then the Q-learning algorithm would fail to learn the optimal price sequence for state $x2$ with two days left before departure, as $a2$ and $a3$ rather than $a4$ and $a5$. The Q-learning algorithm will assign $Q(60_2, 100_2) = 5000$ and $Q(60_2, 90_2) = 6900$, when in fact the true value of $Q(60_2, 100_2)$ is 8000. The Q-learning algorithm,

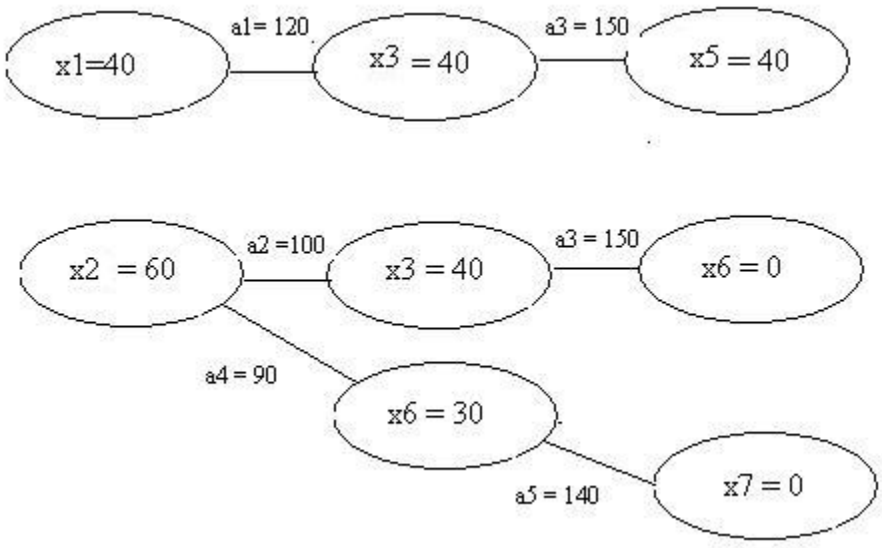

Figure 4: Decision tree for pricing outcomes

because of its temporal adjacency based credit assignment, will fail to learn the true expected value for an action due to this biasing effect. With the Q(λ) algorithm the Q-values will converge and, in the process, implicitly learn the demand autocorrelation.

Using the Q(λ) when there is autocorrelation in demand the Q-values are estimated by Eq. (9):

$$Q_t^*(x_t, a_t) = \max_{a_t}[r(x_t, a_t) + \sum_{x_{t+1} \in X \times T} p_t(x_{t+1}|x_t, a_t)\eta \max_{a_{t+1}} Q_{t+1}^*(x_{t+1}, a_{t+1}|x_t, a_t)]. \tag{9}$$

where $Q_{t+1}^*(x_{t+1}, a_{t+1}|x_t, a_t)$,

$$\begin{aligned} Q_{t+1}^*(x_{t+1}, a_{t+1}|x_t, a_t) = \max_{a_{t+1}}[r(x_{t+1}, a_{t+1}|x_t, a_t) + \\ \sum_{x_{t+2} \in X \times T} p_{t+1}(x_{t+2}|x_{t+1}, a_{t+1}, x_t, a_t)\eta \max_{a_{t+2}}(Q_{t+2}^*(x_{t+2}, a_{t+2}|x_{t+1}, a_{t+1})]. \end{aligned} \tag{10}$$

whilst using the Q-learning algorithm updates of the Q-values are estimated by Eq. (11):

$$Q_t^*(x_t, a_t) = \max_{a_t}[r(x_t, a_t) + \sum_{x_{t+1} \in X \times T} p_t(x_{t+1}|x_t, a_t)\eta \max_{a_{t+1}} Q_{t+1}^*(x_{t+1}, a_{t+1})] \tag{11}$$

It is clear from this example that the eligibility trace learner has an important advantage when demand in the selling horizon is autocorrelated, since, if necessary, it is possible to take

into account the biasing effects of preceding states in assigning credit. The Q-learning algorithm does not take into account the autocorrelation of demand between decision time-steps in the selling horizon and the revenue generated by its policy is significantly lower. This illustrates the effect of simplified model assumptions on the resulting policy and revenue gained, emphasizing the advantages of using a reinforcement learning algorithm, in particular, the Q(λ) algorithm. Models used in literature have made simplified assumptions about the real-world demand process, especially when the demand between time intervals in the selling horizon is autocorrelated.

6 Conclusion

This paper uses reinforcement learning to solve the dynamic pricing problem with finite inventory and non-stationary demand. The Q(λ) algorithm was used to solve the non-stationary Markov decision process. We have shown analytically, and using simulation, that the Q(λ) converges and produces a better policy than the standard Q-learning algorithm.

The learning models offer many advantages: they allow the decision-maker to make pricing decisions without neither knowing the values of exogenous variables (i.e., competitors prices) nor the structural form of the demand function. We show that the parametric algorithm outperforms the non-parametric algorithm when the assumed parametric model is consistent with the true demand function, otherwise the parametric model leads to loss owing to a model misspecification error. The presented learning algorithm can take advantage of both approaches, the initial Q-values can be set to the best believed demand function and, if this is the correct model, convergence will be much faster than when no prior knowledge of demand is assumed. If the true model is different from the assumed structure, then after observing realized demand data, the algorithm derives a policy for the correct underlying demand model.

Additionally, decisions are made using real-time demand since inventory is tracked in real-time and used to determine the current demand level. This paper analyzes how companies can both learn and optimize their pricing strategies while interacting with their customers. In particular the Q(λ) algorithm performs particularly well in situations where the demand between successive days exhibits autocorrelation. Thus it is possible to learn the autocorrelation of the real-time customer arrival rates and customer reservations along with any other influential factors implicitly within

the selling horizon.

7 Appendix

Proof of Theorem 4.1

By relying on the observation made by Szepesvri and Littman [32] then one can deduce that the Q-learning with eligibility traces in a non-stationary finite horizon is a relaxation process.

In order to prove Theorem 4.1 , we start by analyzing if the Q(λ) updating process meets the conditions of Szepesvri and Littman [32, p.2025] corollary 1 by identifying the set of possible state actions $X \times T \times A(X_T)$. If we let $f^k(x_t, a_t)$ be defined by equation (12):

$$f^k(x_t, a_t) = \begin{cases} \alpha(x_t^k, a_t^k) & \text{if } (x_t, a_t) = (x_t^k, a_t^k); \\ 0 & \text{otherwise .} \end{cases} \tag{12}$$

and $\forall\ t \in T,\ \ (P^k Q_t)(x_t, a_t) = r_t^k + \eta \min_{a_{t+1}} Q_{t+1}^k(x_{t+1}^k, a_{t+1})$ then we see that condition 1 and 2 of corollary 1 are satisfied because $\| \alpha(.,.) \| \to 0$ and $k \to \infty$ w.p.1, so for large enough k, $f^k \leq 1$.

It remains to be proven that for a fixed function $Q_t \in B(X \times T \times A)$, the Q($\lambda$) updating process (x_t, a_t) is defined by Eq. 13, if $(x_t, a_t) = (x_t^k, a_t^k)$ and where $Q_{m+1}(x_{m+1}, a_{m+1}) = 0$ and j_t denote the time where the last greedy action was chosen for time t.

$$\begin{aligned} \hat{Q}_t^{k+1}(x_t^k, a_t^k) = \hat{Q}_t^k(x_t^k, a_t^k) + \sum_{t'=t}^{j_t} \lambda^{t'-t} \alpha(x_{t'}^k, a_{t'}^k)[r_{t'}^k + \eta \max_{a_{t'+1}} Q_{t'+1}^k(x_{t'+1}^k, a_{t'+1}) \\ - \hat{Q}_{t'}^k(x_{t'}^k, a_{t'}^k)]. \end{aligned} \tag{13}$$

converges to HQ_t, where H is defined by Eq. (14).

$$(HQ_t)(x_t, a_t) = r(x_t, a_t) + \sum_{x_{t+1} \in X} p_t(x_{t+1} \mid x_t, a_t) \eta \max_{a_{t+1}} \tilde{Q}_{t+1}(x_{t+1}, a_{t+1}) \tag{14}$$

When using Lemma 1 from Szepesvri and Littman [32, p.2024] this is straightforward. We observe that the different components of $\hat{Q}_t^k$ are decoupled, that is $\hat{Q}_t^k(x_t, a_t)$ does not depend on

$\hat{Q}_t^k(x'_t, a'_t)$ and *vice versa* whenever $(x_t, a_t) \neq (x'_t, a'_t)$. Thus it is sufficient to prove the convergence of the one-dimensional process $\hat{Q}_t^k(x_t, a_t)$ to $(HQ_t)(x_t, a_t)$ for any fixed pair (x_t, a_t) and identify Q_t^k of Lemma 1 with $\hat{Q}_t^k(x_t, a_t)$ as defined by Eq. (13).

Let F^k be the field σ-field this is adapted to $< x_t^k, a_t^k, x_{t-1}^k, r_{t-1}^k, ..., x_0^0, a_0^0 >$ If $k \geq 1$ and then F^0 adapted to (x_0^0, a_0^0). We now rewrite the Q(λ) updating process in the form of Lemma 1 as shown in Eq. (15)-(17).

We start by splitting Eq. (13) in order to define it in the form of the equation in lemma 1

$$\begin{aligned}\hat{Q}_t^{k+1}(x_t^k, a_t^k) = (1 - \alpha(x_t^k, a_t^k))\hat{Q}_t^k(x_t^k, a_t^k) + \alpha(x_t^k, a_t^k)[r_t^k + \eta \max_{a_{t+1}} Q_{t+1}^k(x_{t+1}^k, a_{t+1})] \\ + \sum_{t'=t+1}^{j_t} \lambda^{t'-t}\alpha(x_{t'}^k, a_{t'}^k)[r_{t'}^k + \eta \max_{a_{t'+1}} Q_{t'+1}^k(x_{t'+1}^k, a_{t'+1}) - \hat{Q}_{t'}^k(x_{t'}^k, a_{t'}^k)].\end{aligned} \tag{15}$$

Updating the Q-values using eligibility traces allows for a better estimate of the Q-values of future successive state-action pair. This allows one to obtain a better estimate of the $\max_{a_{t+1}} Q_{t+1}^k(x_{t+1}^k, a_{t+1})$. Let $\max_{a_{t+1}} \tilde{Q}_{t+1}^k(x_{t+1}^k, a_{t+1})$ be defined as Eq. (16)

$$\begin{aligned}\max_{a_{t+1}} \tilde{Q}_{t+1}^k(x_{t+1}^k, a_{t+1}) = \max_{a_{t+1}} Q_{t+1}^k(x_{t+1}^k, a_{t+1}) + 1/\alpha(x_t^k, a_t^k)[\sum_{t'=t+1}^{j_t} (\lambda)^{t'-t}\alpha(x_{t'}^k, a_{t'}^k)[r_{t'}^k \\ + \eta \max_{a_{t'+1}} Q_{t'+1}^k(x_{t'+1}^k, a_{t'+1}) - \hat{Q}_t^k(x_{t'}^k, a_{t'}^k)]]\end{aligned} \tag{16}$$

Eq. (17) represents Eq. (13) in the same form as the equation in lemma 1.

$$\hat{Q}_t^{k+1}(x_t^k, a_t^k) = (1 - \alpha(x_t^k, a_t^k))\hat{Q}_t^k(x_t^k, a_t^k) + \alpha(x_t^k, a_t^k)(r_t^k + \eta \max_{a_{t+1}} \tilde{Q}_{t+1}^k(x_{t+1}^k, a_{t+1})) \tag{17}$$

Then at episode k we have for all $t \in T$, $\alpha^k = \alpha(x_t^k, a_t^k)$, $w_t^k = r_t^k + \eta\ max_{a_{t+1}} \tilde{Q}_{t+1}^k(x_{t+1}^k, a_{t+1})$.

The proof from here follows the same as Szepesvri and Littman [32, p.2028], the conditions of Lemma 1 are satisfied,

1)F^k be an increasing sequence of σ-fields by definition;

2)$0 \leq \alpha^k \leq 1$ by the property of $\alpha(x_t^k, a_t^k)$;

3)α^k and w_t^{k-1} are F^k are measurable because of the definition of F^k;

4)$E[w_t^k|F^k, \alpha^k \neq 0] = E[r_t^k + \eta \max_{a_{t+1}} \tilde{Q}_{t+1}^k(x_{t+1}^k, a_{t+1})|F^k] =$

$\sum_{x \in X} P_t(x_{t+1} \mid x_t, a_t)(r(x_t, a_t, x_{t+1}) + \eta \ max_{a_{t+1}} \tilde{Q}_{t+1}(x_{t+1}, a_{t+1})) = (HQ_t)(x_t, a_t)$ because of the first part of condition 1 from Assumption 1;

5)$E[(w_t^k)^2|F^k]$ is uniformly bounded because x_{t+1} can take finite values since, by assumption X is finite, the bounded variance of r_t^k(from the second part of condition 2 from assumption 1);

6)$\sum_{k=0}^{\infty} \alpha^k = \infty$, $\sum_{k=0}^{\infty} (\alpha^k)^2 < \infty$ (Condition 1 of theorem 1).

Therefore we prove that $\hat{Q}_t^{k+1}(x_t^k, a_t^k)$ converges to $E[w_t^k \mid F^k, \alpha^k \neq 0] = HQ_t(x_t, a_t)$, which proves Theorem 4.1. QED.

Proof of Theorem 4.2

We use mathematical induction. It is sufficient to prove for any state-action pair at any time t as they are updated in the same manner.

Initialize $Q_t^1 = 0$

Base case: when $k = 1$, $\forall\ t = 0, 1, 2, .., m$ the Q-learning update is made as follows,

$$\begin{aligned} Q_t'^2(x_t^1, a_t^1) &= (1 - \alpha(x_t^1, a_t^1)Q_t^1(x_t^1, a_t^1) + \alpha(x_t^1, a_t^1)(r_t^1 + \eta max_{a_{t+1}} Q_t^1(x_{t+1}^1, a_{t+1})) \\ &= 0 \cdot (1 - 1) + 1/1 \cdot (r_t^1 + 0) \\ &= r_t^1, \end{aligned}$$

where $\alpha(x_t^1, a_t^1) = 1$. The Q(λ) update is computed as follows,

$$\begin{aligned} Q_t''^2(x_t^1, a_t^1) &= Q_t^1(x_t, a_t) + \sum_{t'=t}^{j_t} \lambda^{t'-t} \alpha(x_{t'}^1, a_{t'}^1)(r_{t'}^1 + \eta \max_{a_{t'+1}} Q_t^1(x_{t'+1}^1, a_{t'+1}) - Q_t^1(x_{t'}^1, a_{t'}^1)) \\ &= 1/1(r_t^1) + 1/1(\lambda)(r_{t+1}^1) + 1/1(\lambda)^2(r_{t+2}^1) + ... + 1/1(\lambda)^k(r_{j_t}^1). \end{aligned}$$

Since all the initial Q-values are set to zero, $j_t = m$. Then $|Q_t'^2(x_t^1, a_t^1) - Q_t^+(x_t, a_t)| > |Q_t''^2(x_t^1, a_t^1) - Q_t^+(x_t, a_t)|$.

We are learning more using the Q(λ) updating rule then, following Theorem 4.1 the difference between the estimated Q-value and the fixed point Q-value is smaller using the Q(λ) updating rule.

Induction step: let k be an arbitrary natural number and suppose that $|Q_t'^k(x_t, a_t) - Q_t^+(x_t, a_t)| \geq |Q_t''^k(x_t, a_t) - Q_t^+(x_t, a_t)|$. Then if $(x_t, a_t) = (x_t^k, a_t^k)$

$$|Q_t'^{k+1}(x_t^k, a_t^k) - Q_t^+(x_t, a_t)| = |(1 - \alpha(x_t^k, a_t^k))Q_t^k(x_t^k, a_t^k) + \alpha(x_t^k, a_t^k)(r_t^k + \eta \max_{a_{t+1}} Q_{t+1}^k(x_{t+1}^k, a_{t+1}) - Q_t^+(x_t, a_t))|.$$

Then if $j_t = t$ we have the Q(λ) update as

$$\geq |Q_t^k(x_t^k, a_t^k)(1 - \alpha(x_t^k, a_t^k)) + \alpha(x_t^k, a_t^k)(r_t^k + \eta \max_{a_{t+1}} Q_t^k(x_{t+1}^k, a_{t+1}) - Q_t^+(x_t, a_t))|.$$

This is equal to Q-learning update if the Q-value has converged to Q^+. Otherwise the Q-value, $Q_t''^{k+1}(x_t^k, a_t^k)$ estimated using the Q(λ), is closer to the Q^+ since $|Q_t'^2(x_t^1, a_t^1) - Q_t^+(x_t, a_t)| > |Q_t''^2(x_t^1, a_t^1) - Q_t^+(x_t, a_t)|$ holds. We can prove this by just observing the Q-value estimated for both algorithms at the base case. Then if $j_t > t$,

$$\geq Q_t^k(x_t^k, a_t^k) + \sum_{t'=t}^{j_t} (\lambda)^{t'-t} \alpha(x_{t'}^k, a_{t'}^k)(r_{t'}^k + \eta max_{a_{t'+1}} Q_{t'+1}^k(x_{t'+1}^k, a_{t'+1}) - Q_{t'}^k(x_{t'}^k, a_{t'})) - Q_t^+(x_t, a_t)|$$

$$= |Q_t''^{k+1}(x_t^k, a_t^k) - Q_t^+(x_t, a_t)|. \text{ (inductive step)}$$

Therefore $|Q_t'^{k+1}(x_t^k, a_t^k) - Q_t^+(x_t, a_t)| \geq |Q_t''^{k+1}(x_t^k, a_t^k) - Q_t^+(x_t, a_t)|$ as required. QED

8 References

[1] K. Lin, Dynamic pricing with real-time demand learning, European Journal of Operational Research. 174 (2006) 522-538.

[2] W.H. Tsaia, S.j. Hung, Dynamic pricing and revenue management process in Internet retailing under uncertainty: An integrated real options approach, Omega. 37 (2009) 471-481.

[3] R. Anjos, R.C.H. Cheng, C. Currie, Optimal pricing policies for perishable products, European Journal of Operational Research. 166 (2005) 246-254.

[4] K. Pan, K.K. Lai, L. Liang, S.C.H. Leung, Two-period pricing and ordering policy for the dominant retailer in a two-echelon supply chain with demand uncertainty, Omega. 37 (2009) 919-929.

[5] G. Gallego, G.j. van Ryzin, Optimal dynamic pricing of inventories with stochastic demand over finite horizons, Management Science. 40 (1994) 999-1020.

[6] R.V. Ramesesh, Lot-sizing decisions under limited-time price incentives: a review, Omega. 39 (2010) 118-135.

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

[8] G.R. Bitran, R. Caldentey, An overview of pricing models for revenue management, Manufacturing & Service Operations Management, 53(3) (2003) 203-229.

[9] W. Elmaghraby, P. Keskinocak, Dynamic pricing in the presence of inventory considerations: research overview, current practices, and future directions, Management Science. 49(10) (2003) 1287-1309.

[10] J.I. McGill, G.J. van Ryzin, Revenue management: research overview and prospects, Transportation Science. 33(2) (1999) 233-256.

[11] L. Weatherford, S. Bodily, A taxonomy and research overview of perishable-asset revenue management: yield management, overbooking and pricing, Operations Research, 40 (1992) 831-844.

[12] P. Belobaba, Airline yield management an overview of seat inventory control, Transportation Science. 21 (1987) 63-73.

[13] R. Chenavaz, Dynamic pricing, product and process innovation, European Journal of

Operational Research. 222 (2012) 553-557.

[14] S. Marbn, R. van der Zwaan, A. Grigoriev, B. Hiller, T. Vredeveld, Dynamic pricing problems with elastic demand, Operations Research Letters. 40 (2012) 175-179.

[15] J. Zhang, W.K. Chiang, L. Liang, Strategic pricing with reference effects in a competitive supply chain, Omega. (2013) http://dx.doi.org/10.1016/j.omega.2013.07.002.

[16] G. Tesauro, J.O. Kephart, Pricing in agent economies using multi-agent Q-learning, Autonomous Agents and Multi-Agent Systems. 5 (2002) 289-304.

[17] J. Peng, R.J. Williams, Incremental multi-step Q-learning, Machine Learning. 22 (1996) 283-290.

[18] R. Anjos, R.C.H. Cheng, C. Currie, Maximizing revenue in the airline industry under one-way pricing, Journal of the Operational Research Society, 55 (2004) 535-541.

[19] Y. Aviv, A. Pazgal, A partially observed Markov decision process for dynamic pricing, Management Science. 51(9) (2005) 1400-1416.

[20] E. Berk, l. Gurler, G. Yildirim, On pricing of perishable assets with menu costs. Int. J. Production Economics. 121 (2009) 678-699.

[21] L. Zhao, P. Tian, L. Xiangyong, Dynamic pricing in the presence of consumer inertia, Omega. 40 (2012) 137-148.

[22] S. Dasu, C. Tong, Dynamic pricing when consumers are strategic: Analysis of posted and contingent pricing schemes, European Journal of Operational Research. 204 (2010) 662-671.

[23] P.K. Banerjee, T.R. Turner, A flexible model for the pricing of perishable assets, Omega. 40 (2012) 533-540.

[24] G. Li, Z. Xiong, Y. Zhou, Y. Xiong, Dynamic pricing for perishable products with hybrid uncertainty in demand, Applied Mathematics and Computation. 219 (2013) 10366-10377.

[25] A.E.B. Lim, J.G. Shanthikumar, Relative entropy, exponential utility, and robust dynamic pricing. Operations Research. 55 (2007) 198-214.

[26] O. Besbes, A. Zeevi, Dynamic pricing without knowing the demand function: risk bounds and near-optimal algorithms. Operations Research. 57(6)(2009) 1407-1420.

[27] A. Gosavi, N. Bandla, T.K. Das, A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking, IIE Transactions. 34

(2002) 729-752.

[28] C. Raju, Y. Narahari, K. Ravikumar, Learning dynamic prices in electronic retail markets with customer segmentation. Annals of Operations Research (2006) 59-75.

[29] A.X. Carvalho, M.L. Puterman, Dynamic pricing and reinforcement learning, Proceedings of the International Conference on Neural Networks. 4 (2003) 2916-2921.

[30] Y. Cheng, Real-time demand learning-based Q-learning approach for dynamic pricing in e-retailing setting, International Symposium on Information Engineering and Electronic Commerce. (2009) 594-598.

[31] R.S. Sutton, A.G. Barto, Reinforcement learning: an introduction, Cambridge, Bradford, 1998.

[32] C. Szepesvri, M.L. Littman, A unified analysis of value-function-based reinforcement learning algorithms. Neural Computation. 11 (1999) 2017-2060.

[33] A. Gosavi, Target-sensitive control of Markov and semi-Markov processes, International Journal of Control, Automation, ans Systems. 9(5) (2011) 1-11.