Using reinforcement learning method to price a perishable product, case study: orange

Abbas ShekariFirouzjaie[1], Navid Sahebjam Nia[2], Hadi Abdollahzade[3]

[1] Industrial Engineering Department, Science and Technology of Behshahr, Mazandran, Iran.
a_shekarii@yahoo.com

[2] Industrial Engineering Department, Science and Technology of Behshahr, Mazandran, Iran.
ns.navid@yahoo.com

[3] Industrial Engineering Department, Science and Technology of Behshahr, Mazandran, Iran.
hadi.sangrodi@gmail.com

Abstract:
Determining the optimal selling price for different commodities has always been one of the main topics of scientific and industrial research. Perishable products have a short life and due to their deterioration over time, they cause great damage if not managed. Many industries, retailers, and service providers have the opportunity to increase their revenue through optimal pricing of perishable products that must be sold within a certain period. In the pricing issue, a seller must determine the price of several units of a perishable or seasonal product to be sold for a limited time. This article examines pricing policies that increase revenue for the sale of a given inventory with an expiration date. Booster learning algorithms are used to analyze how companies can simultaneously learn and optimize pricing strategy in response to buyers. It is also shown that using reinforcement learning we can model a demand-dependent problem. This paper presents an optimization method in a model-independent environment in which demand is learned and pricing decisions are updated at the moment. We compare the performance of learning algorithms using Monte Carlo simulations.

[2] Corresponding author
Received: 2020-08-29 Approved: 2020-10-21

Keywords: Dynamic Pricing, Inventory Management, Reinforcement Learning, Simulation, Perishable Products
JEL Classification: L70, O25.

1 Introduction

It can be noted that the decisions made in the field of inventory are very directly related to the type of business. A business model that is very common in everyday life is a group of companies that work in the field of selling seasonal products (for example, we can refer to the existing business market in the field of fruit supply). Accordingly, in many inventory models, a limited time horizon is assumed, and after this time horizon, it is not possible to buy during the period.
Another important feature in this area is the uncertainty in demand. Demand uncertainty rarely occurs in reality, and demand as an external variable usually has a potential behavior that may follow a known or empirical distribution. In order to model and price real-time inventory systems, it is necessary to consider the possible demand behavior in such models. For pricing such products, where demand is uncertain and seasonal and have a limited lifespan, implementing a dynamic pricing policy can increase revenue and reduce waste. In the subject literature, different methods have been used to deal with this issue. As far as it is shown in the subject literature, reinforcement learning methods are one of the new and very effective methods in the field of dynamic decision making. Achieving the expected total revenue is assessed using the reinforcement learning method.

1.1 Objectives and importance of doing the problem

This article examines the issue of single-product pricing, which was first studied by Galgo and Van Raisin [21]. Proper demand forecasting is essential when developing a pricing policy. In practice, the decision maker rarely has complete information about demand performance. For this reason, imposing a structural form on the performance of demand can lead to the definition of a model that naturally causes income losses. The main purpose of this paper is to present a model-independent approach

that does not specify the probability of its transfer between modes (ie demand behavior) with a particular distribution. Reinforcement learning techniques such as Q-learning and Q-learning with proper Q (λ) routing are proposed to solve the problem of optimizing the dynamic pricing of perishable products when there is random demand with unknown characteristics. The purpose of this paper is to propose an evaluated computational method (e.g., reinforcement learning) to solve the problem of income management when information is incomplete and demand is unstable. In this article, we use two popular methods of learning Q [17, 23] and Q-learning with the right path, that these concepts were proposed by Peng and Williams [17, 23].
In this study, the aim is to obtain an optimal dynamic pricing strategy for perishable goods, so that the seller's income is maximized. Also, due to the fact that not all perishable products can be used after a limited period of time and are completely discarded, in this study, a specific expiration date has been considered for the products. Implementing a dynamic pricing strategy for commodities with these conditions can lead to better decisions and greater profitability.

1.2 Research questions

1) How to price a perishable product?

2) What is the mechanism for maximizing profits?

3) How does product life and inventory affect the price?

1.3 Pricing according to inventory

From the past to present, the issue of inventory control and management has been of great importance and inventory management and control have a significant role in the profitability and loss of an economic center and the development of correct policies and methods has always been considered by economic enterprises. Inventory costs usually account for 20 to 60% of the company's total costs [18, 19, 21]. Therefore, proper pricing of inventory for timely sale and as much as products at the right time is very important and should be considered. This article distinguishes itself from the existing dynamic pricing literature by using reinforcing literature on pricing perishable products with unstable demand. Therefore, this article

is the first study on this issue. Table 1 summarizes the research that has been done in recent years on the dynamic pricing of perishable products by category, in which the various cases studied in this study are presented and compared with the present study. According to the table of items such as product type, the number of products studied, the expiration date of products, the solution method, and technique used, none of them have studied the topics covered in this article.

2 Research Methods

2.1 problem definition

In this paper, the dynamic pricing problem of a perishable product is modeled on finite and discrete horizons. The model under study in this issue has several features, including:

1) limited horizon to sell the product;

2) interdependent demand;

3) inability to buy during the product sales period;

4) limited life and perishability;

5) Dynamic pricing noted.

Dynamic pricing problem is formulated as a Markov decision-making process because pricing is an instant decision-making problem in a random environment. Demand is learned over time and prices are updated instantly. The purpose of this paper is to estimate a pricing policy that maximizes revenue for the sale of inventory of products with a specified expiration date. In practice, because it is difficult to present a price change at a continuous time, the price review is used periodically in which the price changes at discrete times. Here is a basic model for a certain number of identical products or services. The price in each given time period is determined by the remaining capacity and the time (ie number of days) remaining before expiration. In this article, the approach and models of article [21] are used to model this problem.
The key parts of Markov's decision-making process are:

Table 1: Compare our work with the works done

Reference	products perishable	Expiration Definite	Indefinite	Single products	Multi products	Demand stochastic	Method
[**u2**]	*	*		*		*	
[**u3**]			*		*	*	*
[**u4**]	*		*	*			*
[**u5**]	*	*			*		
[**u6**]	*		*		*		
[**u7**]	*		*		*		
[**u8**]	*	*		*			
[**u9**]	*	*			*	*	
[**u10**]	*	*		*			
[**u11**]	*	*		*		*	
[**u12**]	*	*		*		*	
[**u13**]	*	*			*		
[**u14**]	*	*			*	*	
[**u15**]	*	*		*		*	*
[**u16**]			*		*	*	*
[**u17**]			*		*		*
[**u18**]	*		*	*		*	*
[**u19**]	*		*	*		*	*
[**u20**]	*	*			*		*
[**u21**]	*	*			*	*	*
[**u22**]	*		*		*		*
This paper	*	*		*		*	*

(i) State space: $x \in X = \{0, 1, \ldots, n\}$ Indicates the remaining capacity.

(ii) The time horizon $t \in T = \{0, 1, \ldots, m\}$ is a discrete set of finite times performed in each pricing operation.

(iii) X_t Shows the remaining capacity (state of the system) at time t.

(iv) $A(x_t)$ The set of prices that the seller can choose when capacity x remains at time t. $a_t \in A(x_t)$ is a price for capacity at time t.

(v) R The revenue function is defined for each decision step.

The Q-learning and Q(lambda) roughly solve large-scale Markov decision process problems. In the context of the Markov decision-making process: V represents the sum of the rewards expected when starting with $x_t$and following a policy (ie $a_t = \pi(x_t)$ where (s) is the selected action in s when the policy Shows adopted). $Q(x_t, a_t)$ shows the total discount discount expected when starting from x_t performing the action $a_t$and following the policy. Q is the function of the state-action value for the policy at time t. μ discount factor, $0 < \mu < 1$. Under Belman's optimal function, Equation (1) is defined for an arbitrary $x_t \in X$ where $Q^*(x_t, a_t)$is the optimal value function for each state-action pair.

$$Q^*(x_t, a_t) = \sum_{x_{t+1} \epsilon X} p_t(x_{t+1} | x_t, a_t)[r(x_t, a_t, x_{t+1}) + \mu \max_{a_{t+1} \in A(x_{t+1})} Q^*_{t+1}(x_{t+1}, a_{t+1})] \quad (1)$$

In T0-learning and Q(lambda) example, the decision-maker reacts to the environment during a set of actions. The modified environment then sees a new mode agent and a reward signal at any point in time. In this process, learning takes place during trial and error in a dynamic environment.

During the learning process, Q values are stored and updated from each $Q(x_t, a_t)$ state-action pair. A value of Q indicates the usefulness of performing a pricing operation when the environment is in the x_t state. This paper considers a dynamic pricing model of identical items on a given sales horizon that allows the value of action-pair pairs to be learned from a sales horizon onwards. Suppose k represents the sales horizon and each

sales horizon is divided into m time intervals. The sales horizon section addresses several issues of dynamic pricing in consecutive time horizons, and the transfer probabilities are the same for different segments but are unstable during each segment. These algorithms include updating Q_t^k, Q values on each sales horizon k for time t, which provides an approximation of Q_t^*. The optimal Q values are the currently observed transitions and the rewards $< x_t^k, a_t^k, x_{t+1}^k, r_t^k >$ where $x_t^k, a_t^k, , r_t^k$ are the remaining capacities, respectively. The remainder, price action, is the current reward observed at time t in k and x_{t+1}^k is the new residual capacity at time t + 1. Note that r_t^k depends on x_t^k, a_t^k and $x_{t+1}^k : r_t^k \equiv r(x_t^k, a_t^k, x_{t+1}^k)$. An updated rule of the Q learning method in Equation (2) shows that $\alpha\left(x_t^k, a_t^k\right)$ is the learning rate and $0 < \mu <1$ for t = 1,2,. . . , m.

$$Q_{t+1}^k\left(x_t^k, a_t^k\right) = \left(1 - \alpha\left(x_t^k, a_t^k\right)\right) Q_t^k\left(x_t^k, a_t^k\right) + \alpha(x_t^k, a_t^k)(\ r_t^k + \mu) \max_{a_{t+1} \in A(x_{t+1})} Q_{t+1}^k(x_{t+1}^k, a_{t+1}) \tag{2}$$

The Q (λ) algorithm is a generalization to the one-step Q-learning algorithm. The value of Q is approximated by the value of all up-to-date state-action pairs. The idea of decent routing Q(λ) is very simple: each time a mode-action pair is selected in a segment, a short-term memory (known as tracking) is allocated, which disappears as the sales horizon moves. The amount of traceability determines the suitability of a mode-action pair for learning. The recently visited mode-action pair is more worthy. Properly reinforced learning has the important potential of learning on a non-fixed sales horizon, especially in situations where demand is interdependent between consecutive times (such as days), as it can reveal hidden states in the decision-making process.

The transfer information available to the decision maker in k at time t is $< x_t^k, a_t^k, x_{t+1}^k, r_t^k >$. The proper path of the function is shown in part k at time t with e_t^k. In the transfer experience $< x_t^k, a_t^k, x_{t+1}^k, r_t^k >$ the following updates are made to be traceable: $e_t^k\left(x_t^k, a_t^k\right) = 1,\ \forall i < t, e_t^k\left(x_i, a_i\right)\ if\ Q_t^k\left(x_t^k, a_t^k\right) = \max_{a_t \in A(x_t)} Q_t^k\left(x_t^k, a_t\right)$, Otherwise $e_t^k\left(x_i, a_i\right) = 0$. For all mode-operation pairs, routing decreases by λ, except for the last mode-operation visited, where the proper path increases by one unit. Tracking can be updated in two ways. If a greedy choice is made, then all traces of the action-pair pairs are viewed in a discount parameter λ. If an exploratory action is performed, the results of the worthy path are set to zero. In k, for each time t, the error is approximately equal to

$\delta_t^k = r_t^k \max_{a_{t+1}} Q_t^k\left(x_{t+1}^k, a_{t+1}\right) - Q_t^k\left(x_t^k, a_t^k\right)$. And is assigned to each of the previous state-action pairs that have been visited in section k according to their tracking properties. The error is calculated at any time t, and its value and the value of the action-pair pairs previously visited in that section are updated. The initial state-of-action pairs visited are given less credit for the current error. The pricing operation in the earlier stages receives a lower reward or penalty for time error t. Learning is accelerated by proper navigation. The goal of the learning strategy is to train decision-makers on optimal pricing policy. The algorithm Q (λ) calculates the desired value function iteratively: for each case x at any time t, the optimal value of $Q_t^*(x_t, a_t)$ from each operation a_t is estimated based on the simulated transitions. When all these values are approximated correctly, the optimal policy is obtained from Equation (2): $\forall t \leq m,\ \forall x \in X,\ \pi_t^*(x_t) = \max_{a_t \in A(x_t)} Q_t^*(x_t, a_t)$.

2.2 Algorithm expression

The purpose of this algorithm is to obtain an accurate estimate of the optimal policy based on observations in the exploration phase while keeping the exploration rate small to limit the loss of income in this phase of learning. The selected exploration rate (ε) is a greedy policy at the 1/k rate, so that learning progresses as the exploration rate decreases. The result of this assumption is that the more knowledge the decision-maker acquires, the less sub-optimal prices are explored. The learning rate is adjusted similarly and also decreases over time. The learning rate for each state-action pair is denoted by $\alpha(x_t^k, a_t^k)$. The amount of learning is equal to $\alpha(x_t^k, a_t^k)$., which is equal to $1/(n^k(x_t, a_t))$, which $(n^k(x_t, a_t))$is equal to 1 plus the number of times that the (x_t, a_t)mode-action pair is visited by process (x_t^k, a_t^k)before time k.

The algorithm starts with the initial value of Q and the following steps are repeated for each part of Q. All eligible tracks are set to zero at the beginning of the selling horizon for all action-pair pairs. Initial mode, (x_t^k) (total inventory number) and (a_t^k) (product price) are selected. After selecting this price, the next state (x_{t+1}^k) (remaining capacity) and the instantaneous income obtained (r_t^k) will be observed. The (a_{t+1}^k) value is then selected using the π (non-greedy) policy. The time difference error $(\delta\ )$ is calculated and the $e_t^k(x_t^k, a_t^k)$ routing is updated to 1. All Q-values of the action-pair pairs is then updated using the tracking property.

All traces are updated: if the value of (a_{t+1}^k) is a greedy action (so far desirable), then all traces are multiplied by a parameter λ, if (a_{t+1}^k) is an exploratory action then all traces are set to zero. Then the next state (x_{t+1}^k) at time t+1 and operation (a_{t+1}^k) becomes the new (x_t^k) and (a_t^k). This process is repeated until the last time m for each part.

2.3 Solving Method

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. Reinforcement learning differs from supervised learning in not needing labelled input/output pairs be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead the focus is on finding a balance between exploration and exploitation.

Temporal difference (TD) learning refers to a class of model-free reinforcement learning methods which learn by bootstrapping from the current estimate of the value function. These methods sample from the environment, like Monte Carlo methods, and perform updates based on current estimates, like dynamic programming methods. While Monte Carlo methods only adjust their estimates once the final outcome is known, TD methods adjust predictions to match later, more accurate, predictions about the future before the final outcome is known.

3 Being independent of the model versus the parametric structure

Here, the model-independent method is compared with the parametric learning algorithm. It starts with a one-step process in which Q-learning and Q (λ) are equally effective. The performance of three algorithms is examined: (a) learning algorithm Q (b) learning algorithm Q in which the values of Q are estimated to be the best function of demand and (c) a learning algorithm in which the parametric structure is known but the parameter values (adjustment parameters)) Is unknown.
The exponential demand model(1), $\theta = l_1 \exp(1)$, $\lambda(a) = \theta \exp(-l_2 a)$ and

the linear demand model $\lambda(a) = h_1 - h_2 a$, the values of the parameter used are different. Here represents the pricing action. The Q learning algorithm does not make any assumptions about the structure of the demand function. Learning Q with the Best Estimated Demand Function Algorithm (Best Q-Learning Estimation) gives the initial values of Q of a demand function that the decision-maker believes is reasonable. For the exponential demand model, the initial values of Q are performed using the demand function $\lambda(a) = 15\exp(-0.5a)$ And for the linear model with λ (a) = 30-5a. Table 2 shows the performance of the three algorithms for the two basic demand models for different parameter values. In the first set of experiments, the parametric algorithm first assumes an exponential parametric structure for the demand function, $, \theta = l_1 \exp(1)$, $\lambda(a) = \theta\exp(-l_2 a)$ but the basic 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 main demand model is assumed to be exponential by the parametric algorithm.

3.1 Comparison of Q-learning and Q (λ)

In the next step, the performance of Q-learning algorithms, Q learning with the best estimate, Q (λ), Q (λ) with the best estimate, parametric (well-defined), and parametric (incorrect) algorithms are compared. Consider the issue of an orange warehouse that the wholesaler does not know the customer demand, that is, has no explicit model of customer buying behavior and only sees the realized demand after the pricing operation in different cases. Demand data is generated using the following hypotheses:

(i) Time-dependent Poisson distribution. The customer entry rate is a Poisson distribution with discrete times with average mean μ (t). It is assumed that the average entry rate decreases with the linear rate μ (t) -5t. This assumption is random and has an element of demand correlation. The initial customer entry rate randomly follows a uniform distribution [b, c]. For these experiments b = 50 and c = 100.

(ii) The customer purchase price for services increases exponentially as the decision time approaches the expiration time. Customers who do not buy this product do not expect a cheaper price. The time

Table 2: The performance of the algorithms is measured as a percentage of the optimal policy for the various demand functions

The right demand	Parametric (incorrect)	Q-Learning best estimates	Q - learning
Linear			
h1=42,h2=2.5	83	93.4	92.6
h1=20,h2=1.5	83.1	95	94.1
h1=55,h2=5	81	94.5	94.5
h1=25,h2=0.5	81.4	93.4	92.3
h1=20,h2=0.4	82.2	94.8	94.4
Exponential			
l1=15,l2=1	64.5	93.2	93.4
l1=20,l2=3	62.9	94.5	92.1
l1=25,l2=4	63.1	93.6	90.4
l1=30,l2=3	65.4	94.4	93.4
l1=15,l2=0.5	62.3	93.1	92.2

horizon is considered as the discrete set T = {1,10} and the set of feasible prices {70,100}.. Inventory is a discrete set of {10,100}.

The performance of these algorithms is measured by the percentage of deviation from the optimal policy. The optimal policy is a policy that is under full information about demand. The results presented here are based on the results of the implementation of 10^3independent simulations from which the average income was calculated. The best-estimated demand for learning Q and Q (λ) is defined as $75 - 5t.\exp(2/t.a/100)$ the good parametric definition is defined as $\mu(t) - \beta_1 t.\exp(-\beta_2/t.a/100)$, and an incorrect parameter is defined as $\mu(t)\exp(\beta_1).\exp(-\beta_2/t.a/100)$. Table 3 shows a summary of the performance of the algorithms. Next, the learning algorithms Q and Q (λ) are examined in more detail. To compare the performance of the Q (λ) algorithm with simple Q-learning, the total expected revenue generated by both algorithms along with their

Table 3: The performance of the algorithms is measured as a percentage of the optimal policy

Algorithm	Percentage of optimality
Q- Learning	72.1
Learn Q with the best estimates	74
Q (λ)	91.3
Q (λ) with best estimate	94
Parametric (correct definition)	97.1
Parametric (incorrect)	59.8

Table 4: Expected revenue in different iterations for the Q-learning and Q (λ) algorithm

Episodes	Average Q (λ)	Average Q-learning	LC Q (λ)	UC Q (λ)	LC Q-learning	UC Q-learning
50	4966	4532	3800	5672	4211	4102
500	8934	6042	8100	9620	5623	6515
1000	9290	6610	8431	10150	6040	7121
2000	9660	7100	8822	10521	6531	7635

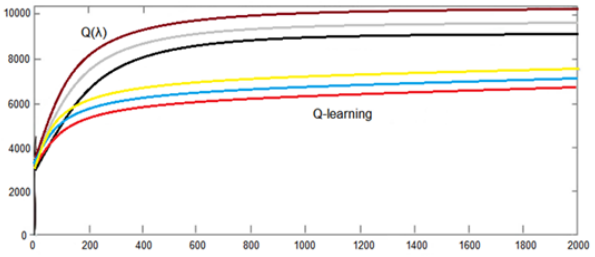

Figure 1: Compare the expected income from Q-learning and Q (λ) as a function of the number of parts

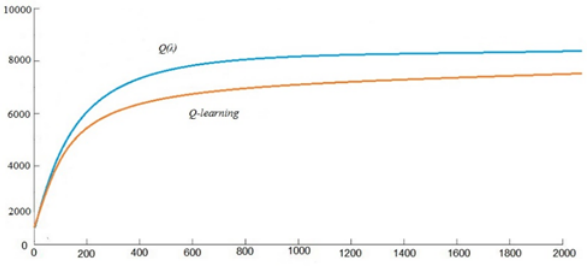

Figure 2: Compare the expected income from Q- learning and Q as a function of capacity Convergence occurs for a case, type of decision stage, practical measures with an exploration rate of 0.01

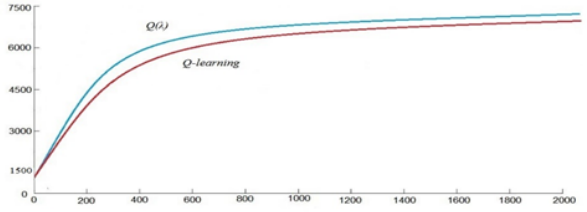

Figure 3: Compare the expected income from learning Q and Q (λ) as a function of capacity Convergence occurs for a case, type of decision stage, practical measures with an exploration rate of 0.05

is reported as a percentage. The maximum result is set to 100%. The results show that a lower exploration rate produces a higher expected total revenue. Q (λ) gives better results when the exploration rate is lower. This is because when the amount of exploration is high, it does the updates in a similar way to Q-learning. The sensitivity analysis discussed above shows that the Q (λ) algorithm is better than the standard Q learning algorithm for all detection rates tested. Another reason for the difference in total expected revenue between the learning algorithm Q and the algorithm Q (λ) is that demand shows a correlation between consecutive days. Customer entry rates are randomly assumed to be uniformly distributed in the initial case. As time goes on, the number of customers entering decreases exponentially.

Table 5: Percentage of highest receipts at different exploration rates

Algorithm	1/k	0.01	0.05	0.1	0.5
Q-learning	92.1	92.1	92.1	90.5	81.5
Q (λ)	100	99.7	99.8	94.1	84.7

By comparing the average learning performance of Q and Q (λ) in different fields, as reported in Tables 3-5 and in Figures 1-3. We now consider two very specific examples of the behavior of algorithms in pricing different units. To this end, Figure 3 reports the revenue for each of the 2000 units generated using the Monte Carlo simulation. The specific revenues reported for each of the 2000 units are random because the customer entry rate is generated randomly.

4 Conclusion

In this paper, reinforcement learning is used to solve the problem of dynamic pricing with specific inventory and unstable demand. The Q (λ) algorithm was used to solve Markov's unstable decision-making process. We have shown analytically, using simulations, that Q (λ) converges, creating a better policy than the standard Q learning algorithm.
In this paper, it is shown that the parametric algorithm is better than the non-parametric algorithm as long as the assumed parametric model is consistent with the actual demand function, otherwise the parametric model leads to a loss due to a model error error. In addition, decisions

made using demand at the moment are used because inventory is tracked at the moment and used to determine current demand levels. This article analyzes how companies can both learn and optimize their pricing strategies in the face of the customer. The Q (λ) algorithm in particular performs well, especially when demand is correlated between consecutive days.

In this article, dynamic pricing of single products was considered, which can be proposed simultaneously for several products and the selection of the best pricing strategy.Pricing for non-perishable products such as cereals and wheat using the method used in this issue.In this paper, time is considered discretely that continuous time can be used in future studies.

Bibliography

[1] Miguel F Anjos, Russell CH Cheng, and Christine SM Currie, *Maximizing revenue in the airline industry under one-way pricing*, Journal of the Operational Research Society **55** (2004), no. 5, 535–541.

[2] ———, *Optimal pricing policies for perishable products*, European Journal of Operational Research **166** (2005), no. 1, 246–254.

[3] Tal Avinadav and Teijo Arponen, *An eoq model for items with a fixed shelf-life and a declining demand rate based on time-to-expiry technical note*, Asia-Pacific Journal of Operational Research **26** (2009), no. 06, 759–767.

[4] Tal Avinadav, Avi Herbon, and Uriel Spiegel, *Optimal ordering and pricing policy for demand functions that are separable into price and inventory age*, International Journal of Production Economics **155** (2014), 406–417.

[5] Yossi Aviv and Amit Pazgal, *A partially observed markov decision process for dynamic pricing*, Management science **51** (2005), no. 9, 1400–1416.

[6] Seyed Mohammad Esmaeil Pour Mohammad Azizi and Abdolsadeh Neisy, *A new approach in geometric brownian motion model*, Fuzzy Information and Engineering and Decision (Cham) (Bing-Yuan Cao, ed.), Springer International Publishing, 2018, pp. 336–342.

[7] Seyed Mohammad Esmaeil Pourmohammad Azizi and Abdolsadeh Neisy, *Mathematic modelling and optimization of bank asset and liability by using fractional goal programing approach*, International Journal of Modeling and Optimization **7** (2017), no. 2, 85.

[8] Alexandre X Carvalho and Martin L Puterman, *Dynamic pricing and reinforcement learning*, Proceedings of the International Joint Conference on Neural Networks, 2003., vol. 4, IEEE, 2003, pp. 2916–2921.

[9] Yan Cheng, *Real time demand learning-based q-learning approach for dynamic pricing in e-retailing setting*, 2009 International Symposium on Information Engineering and Electronic Commerce, IEEE, 2009, pp. 594–598.

[10] Richard P Covert and George C Philip, *An eoq model for items with weibull distribution deterioration*, AIIE transactions **5** (1973), no. 4, 323–326.

[11] Guillermo Gallego and Garrett Van Ryzin, *Optimal dynamic pricing of inventories with stochastic demand over finite horizons*, Management science **40** (1994), no. 8, 999–1020.

[12] PM Ghare, *A model for an exponentially decaying inventory*, J. ind. Engng **14** (1963), 238–243.

[13] Abhuit Gosavii, Naveen Bandla, and Tapas K Das, *A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking*, IIE transactions **34** (2002), no. 9, 729–742.

[14] Chengzhi Jiang and Zhaohan Sheng, *Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system*, Expert Systems with Applications **36** (2009), no. 3, 6520–6526.

[15] Ahmet Kara and Ibrahim Dogan, *Reinforcement learning approaches for specifying ordering policies of perishable inventory systems*, Expert Systems with Applications **91** (2018), 150–158.

[16] Kyle Y Lin, *Dynamic pricing with real-time demand learning*, European Journal of Operational Research **174** (2006), no. 1, 522–538.

[17] Amir Hossein Nafei, Seyed Mohammad Esmaeil Pourmohammad Azizi, and Rajab Ali Ghasempour, *An approach in solving data envelopment analysis with stochastic data*, International workshop on Mathematics and Decision Science, Springer, 2016, pp. 154–162.

[18] Amir Hossein Nafei, Wenjun Yuan, and Hadi Nasseri, *Group multi-attribute decision making based on interval neutrosophic sets*, Infinite Study, 2019.

[19] Jing Peng and Ronald J Williams, *Incremental multi-step q-learning*, Machine Learning Proceedings 1994, Elsevier, 1994, pp. 226–232.

[20] CVL Raju, Y Narahari, and K Ravikumar, *Learning dynamic prices in electronic retail markets with customer segmentation*, Annals of Operations Research **143** (2006), no. 1, 59–75.

[21] Rupal Rana and Fernando S. Oliveira, *Real-time dynamic pricing in a non-stationary environment using model-free reinforcement learning*, Omega **47** (2014), no. C, 116–126.

[22] Rupal Rana and Fernando S Oliveira, *Dynamic pricing policies for interdependent perishable products or services using reinforcement learning*, Expert systems with applications **42** (2015), no. 1, 426–436.

[23] Gerald Tesauro and Jeffrey O Kephart, *Pricing in agent economies using multi-agent q-learning*, Autonomous agents and multi-agent systems **5** (2002), no. 3, 289–304.

[24] Chih-Te Yang, Liang-Yuh Ouyang, and Hsing-Han Wu, *Retailer's optimal pricing and ordering policies for non-instantaneous deteriorating items with price-dependent demand and partial backlogging*, Mathematical Problems in Engineering **2009** (2009).

[25] Yajun Zhang and Zheng Wang, *Integrated ordering and pricing policy for perishable products with inventory inaccuracy*, 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE) (2018), 1230–1236.

[26] Wen Zhao and Yu-Sheng Zheng, *Optimal dynamic pricing for perishable assets with nonhomogeneous demand*, Management science **46** (2000), no. 3, 375–388.