Hindawi
Complexity
Volume 2021, Article ID 6643131, 17 pages
https://doi.org/10.1155/2021/6643131

Research Article

Solving a Joint Pricing and Inventory Control Problem for Perishables via Deep Reinforcement Learning

Rui Wang,[1] **Xianghua Gan**,[1] **Qing Li**,[2] **and Xiao Yan**[1]

[1]*School of Business Administration, The Southwestern University of Finance and Economics, Chengdu, Sichuan, China*
[2]*China Construction Bank, Hengshui Branch, Hengshui, Hebei, China*

Correspondence should be addressed to Xianghua Gan; ganx@swufe.edu.cn

Received 24 October 2020; Revised 6 January 2021; Accepted 12 January 2021; Published 30 January 2021

Academic Editor: Abd E.I.-Baset Hassanien

We study a joint pricing and inventory control problem for perishables with positive lead time in a finite horizon periodic-review system. Unlike most studies considering a continuous density function of demand, in our paper the customer demand depends on the price of current period and arrives according to a homogeneous Poisson process. We consider both backlogging and lost-sales cases, and our goal is to find a simultaneously ordering and pricing policy to maximize the expected discounted profit over the planning horizon. When there is no fixed ordering cost involved, we design a deep reinforcement learning algorithm to obtain a near-optimal ordering policy and show that there are some monotonicity properties in the learned policy. We also show that our deep reinforcement learning algorithm achieves a better performance than tabular-based Q-learning algorithms. When a fixed ordering cost is involved, we show that our deep reinforcement learning algorithm is effective and efficient, under which the problem of "curse of dimension" is circumvented.

1. Introduction

The inventory control of perishables has received increasing attention from the business community and academia. According to a report released by the Food Market Institute (2012) in the United States, as of 2005, the total sales of perishables accounted for more than half of sales in supermarkets and grocery stores in the US, and this proportion is still increasing. Meanwhile, losses due to the deterioration of perishables also account for a large proportion of the total retail cost. Besides, pricing is also an important and effective lever for the retail industry to manage the profitability of perishables. As shown in Karaesmen et al. [1] and Chen et al. [2], a firm's profit increases significantly by dynamic adjustment of prices of perishables according to the availability of the inventory and the remaining lives of perishables.

In this research, we study a joint pricing and inventory control problem for perishables in a finite planning horizon. Demand in each period depends on the current price and satisfies a Possion distribution. The problem of inventory control for perishables is usually more difficult than the one for nonperishables, in which the inventory state can be represented by a single variable. The state of perishables has to be recorded by a vector to account for items with different lifetimes, which makes the analytical studies much more difficult. As a fixed ordering cost can make the problem even more difficult in a dynamic setting, few studies consider it due to the tractability in analysis.

One main contribution in this research is that we consider a fixed ordering cost in our model. We study both the backlogging case and the lost-sales case and allow for positive lead time. Our goal is to find a near-optimal ordering and pricing policy to maximize the expected profit in the planning horizon. This problem is hard to analyze by traditional dynamic programming approach in the inventory control literature. Therefore, we use a reinforcement learning approach to solve the problem.

In the literature, there have been a few papers that study inventory control problems with reinforcement learning, such as Charharsooghi et al. [3], Dogan et al. [4], and Kara et al. [5]. Unlike these papers which use Q-learning, we take a deep reinforcement learning approach and show that it outperforms Q-learning models that do not use neural networks. The outperformance of deep reinforcement learning has also been shown by Ke et al. [6] and Shihab et al. [7] for complex problems.

In this paper, we set up deep reinforcement learning models to study the joint pricing and inventory control problem of perishables. We adopt a FIFO (first-in-first-out) policy in this study. When there is no fixed ordering cost involved, we show that the fixed pricing strategy is dominated by the dynamic pricing strategy, under which the price can be adjusted according to the availability of inventory and the lives of remaining items. We set up a benchmark based on realized demand for this no fixed ordering cost case and show that our designed deep reinforcement learning methods achieve a better performance than tabular-based Q-learning. We also find some monotonicity properties in our learned policies; our learned order quantity is nonincreasing in inventory position or on-hand inventory and price decision is most sensitive to the oldest on-hand inventory. Moreover, in order to show the expansibility of the proposed algorithm, we extend the distribution of the demand and take an *additive* form in Chen et al. [2] where the customer demand depends on the price of current period plus an additive random term; finally, we obtain a near-optimal performance by our proposed deep reinforcement learning models. When the fixed ordering cost is taken into account in the joint pricing and inventory control system, we set up a performance upper bound based on the realized demand in each period in order to assess the performance. Through our proposed methods, we find convergent policies and critical values under which orders should be placed.

2. Literature Review

We review two streams of literature which are closely related to our research: traditional inventory control management for perishables and inventory control management with reinforcement learning.

2.1. Traditional Inventory Control Management for Perishables. There is a considerable literature devoted to dynamic inventory control for nonperishable products; see, for example, Presman and Sethi [8], Caliskan-Demirag et al. [9], Alp et al. [10], Almaktoom et al. [11], Azghandi et al. [12], Li et al. [13], and Gan et al. [14]. The dynamic inventory control for perishable products has not been widely studied in the literature. This is not to say that the literature does not realize the importance of the study of perishable studies.

Indeed, there are a number of papers devoted to the study of inventory decisions for perishable products. Nahmias and Pierskalla [15] studied a dynamic inventory control with a fixed lifetime, zero lead time, and uncertainty demand for perishable products. Nahmias [16], Fries [17], and Nahmias [18] studied the same problem, with multiple periods of lifetime and zero lead time, and their research studies are all to satisfy the same assumption that only products that exceed the life cycle will be abandoned, which is known as the first-in-first-out policy (FIFO), and this policy is widely used in the research of perishable retailing. And they proved that the optimal order quantity under different inventory ages is decreasing. Prastacos [19] reviewed some important theories and practices in blood inventory management and proposed that this kind of application can be extended to other perishable product inventory control problem. Ferguson and Koenigsberg [20] considered a two-period joint pricing and inventory control problem with a random lifetime, emphasizing and discussing the impact of competition between new inventory and surplus inventory over the previous period on inventory and pricing decisions for the first time interval. Chen et al. [21] used Pontryagin's maximum principle method to investigate the optimal policies for the pricing and replenishment of fashion apparel with short product lifecycles. Heuristic algorithms are also increasingly being used to address the problem of dynamic pricing and inventory control for perishables. Li et al. [22] proposed a base-stock/list-price heuristic policy to solve the problem of dynamic pricing and inventory control for a perishable product, assuming that the demand is a function of price and zero lead time. Li and Lu [23] studied a joint optimization of the price and order quantity of a perishable product and proposed a Minimax Regret algorithm. Li et al. [24] discussed a new dynamic pricing and inventory control scenario for perishables. New and old products cannot be sold at the same time. The seller can decide whether to discard the remaining inventory in the previous period, even though the lifetime may not be over. And they proposed a fractional programming heuristic algorithm to obtain a stable structural policy.

Chen et al. [2] is closely related to our research. They considered positive lead time and used the concept of L-convexity/concavity to analyze the problem and proposed a heuristic algorithm to solve the problem. However, the traditional approach used in their research is not able to solve the problem with a fixed ordering cost. By using neural networks with hidden layers to approximate state-action values, our deep reinforcement learning approach exploits the advantages of deep learning [25] and reinforcement learning and is shown to be effective and efficient to find the solution.

2.2. Inventory Control with Reinforcement Learning. In the literature, there have been few papers that study inventory control problems with reinforcement learning.

Giannoccaro and Pontrandolfo [26] studied the coordination of inventory policies adopted by different supply chain factors which are a major issue in supply chain inventory management, and they used a reinforcement learning approach to manage inventory decisions at all stages of the supply chain in an integrated manner and aimed at optimizing the performance of the whole supply chain. Chaharsooghi et al. [3] proposed an inventory control system based on reinforcement learning methods, which included uncertain delivery times and uncertain customer requirements to determine the ordering policy for each order point in the supply chain. Chaharsooghi et al. [3] used Q-learning to solve supply chain ordering management and applied to the beer game. Jiang and Sheng [27] proposed a case-based reinforncement learning algorithm (CRL) for dynamic inventory control in a multiagent supply-chain system. They studied a multiagent simulation of a simplified two-echelon supply chain and showed the effectiveness of the method they proposed. Sui et al. [28] considered a Vendor-Managed Inventory (VMI) system where the supplier makes decisions of inventory decisions of inventory management for the retailer, and the retailer is not responsible for placing orders. Through a methodology based on reinforcement learning and numerical study, they show their approach can outperform the newsvendor. Zarandi et al. [29] presented a flexible fuzzy reinforcement learning algorithm where the value function is approximated by a fuzzy rule-based system and considered the problem of a fuzzy agent (supplier), that is, how to determine the amount of orders for each retailers based on their utility for supplier when its supply capacity is limited. Finally, the effectiveness of their proposed algorithm is proved by a simulation. Dogan et al. [4] used the Q-learning method to study an ordering and pricing policy in a multiretailer environment. Rana and Oliveira [30] use reinforcement learning methods to develop dynamic pricing strategies for interdependent perishable products or service. Kara and Dogan [5] used Q-learning and Sarsa reinforcement learning algorithms to study a dynamic inventory control issues for perishable products, with positive lead time and fixed lifetime. Our research further uses deep reinforcement learning to study this dynamic inventory control of perishable products.

The aforementioned studies investigate the inventory problem for nonperishable and perishable products and use the nondeep reinforcement learning methods. Compared to their problems, our problem focuses on the inventory control of perishables, which makes the problem much more difficult. We use neural networks to avoid the curse of dimensionality and show that our deep reinforcement learning model outperforms the traditional reinforcement learning models without using neural networks.

3. Model

We consider a periodic-review single-product inventory system over a finite horizon of T periods. The whole process can be defined as a Markov Decision Process. The decision maker is called the *agent*, and the thing it interacts with is called the *environment*. At each period (step) of a sequence of discrete time periods, $t = 1, 2, \ldots, T$, the agent and the environment interact; the agent selects the action denoted by A^t , and the environment responds to A^t and presents a new situation to the agent. At the end of the period, the agent receives a numerical *reward* denoted by R^{t+1}, $R^{t+1} \in \mathbb{R}$, in part as a consequence of its action. Throughout this paper, we let superscript t denote the period. More specifically, by superscript t, we mean the beginning of the period; we denote the end of period t, which coincides with the beginning of the next period as $t + 1$.

Customer demand, denoted by D^t, at the beginning of period t, is represented by a Poisson distribution with the parameter as d^t or $d^t(A^t)$ if the agent's action A^t at the beginning of period t changes the demand distribution and $d(\cdot)$ is a function of selling price p, strictly decreasing the selling price p. Let the product's finite *lifetime* be denoted by l, *variable cost* by c^t, and *leadtime* by L^t $(0 \le L^t < l)$. Let the *age* of an item be 0 by the time it is shipped to the agent, and its *residual lifetime* be $l - i$ when its age is i. When an item's age is greater than l, it has to be disposed. The inventory state, also known as the state of the agent, at the beginning of period t can be represented by a $(l - 1)$-dimensional vector:

$$X^t = \left(x_1^t, \ldots, x_i^t, \ldots, x_{l-1}^t\right), \tag{1}$$

where x_i^t represents the level of inventory position of the items at the age of i. In particular, $x_o^t \equiv \sum_L^{l-1} x_i^t$ (when $L = 0$, $x_o^t \equiv \sum_1^{l-1} x_i^t$) is the level of on-hand inventory, and $x^t \equiv \sum_1^{l-1} x_i^t$ is the level of inventory position of all ages.

3.1. An Action.

Here, action space A^t refers to the order quantity q^t and price decision p^t. The selling pricing p^t is restricted to an interval $[\underline{p}, \overline{p}]$. Based on the selling price p, the parameter of Poisson demand $d \in [\underline{d}, \overline{d}]$, where $\underline{d} = d(\overline{p}), \overline{d} = d(\underline{p})$:

$$A^t = \left(q^t, p^t\right). \tag{2}$$

3.2. Update Rule.

Update rule, denoted by $h(\cdot)$, describes the update of the environment state. In our research, the supply state remains unchanged in each period $(M^{t+1} = M^t)$. The demand state in each period depends on selling price p. Last, we need to define the update rules for the inventory state.

The update rules for the inventory state X^t can be divided into two cases according to the unmet demand handing principle. We first consider the backlogging case. If $L = 0$ and $L = 1$, then $x_i^{t+1} = x_{i-1}^t - (D^t - (x_i^t + \cdots + x_{l-1}^t))^+$ for $i = 1$ and $x_i^{t+1} = (x_{i-1}^t - (D^t - (x_i^t + \cdots + x_{l-1}^t))^+)^+$ for $i = 2, \ldots, l - 1$; although $L = 0$ and $L = 1$ have the same state transition rule, the reward function is different; if $L > 1$, then $x_i^{t+1} = x_{i-1}^t$ for $i = 1, \ldots, L - 1$, $x_i^{t+1} = x_{i-1}^t - (D^t - (x_i^t + \cdots + x_{l-1}^t))^+$ for $i = L$, and $x_i^{t+1} = (x_{i-1}^t - (D^t - (x_i^t + \cdots + x_{l-1}^t))^+)^+$ for $i = L + 1, \ldots, l - 1$.

For the lost-sales case, if $L = 0$ and $L = 1$, then $x_i^{t+1} = (x_{i-1}^t - (D^t - (x_i^t + \cdots + x_{l-1}^t))^+)^+$ for $i = 1, \ldots, l - 1$; if $L > 1$, then $x_i^{t+1} = x_{i-1}^t$ for $i = 1, \ldots, L - 1$ and $x_i^{t+1} = (x_{i-1}^t - (D^t - (x_i^t + \cdots + x_{l-1}^t))^+)^+$ for $i = L, \ldots, l - 1$.

3.3. Reward Function.

In our study, our goal is to maximize the accumulative expected profit in the planning horizon, so our reward function R^{t+1} can be represented by the following form. We first consider the backlogging case. If $L = 0$, then

$$R^{t+1}=\begin{cases} p^t * D^t - c*q^t - h*\left(x_o^t + x_0^t - D^t\right) - \nu*\left(x_{l-1}^t - D^t\right)^+, & \text{if } x_o^t + x_0^t \geq D^t,\\ p^t * D^t - c*q^t - u*\left(D^t - x_o^t - x_0^t\right), & \text{if } x_o^t + x_0^t < D^t.\end{cases} \tag{3}$$

If $L > 0$, then

$$R^{t+1}=\begin{cases} p^t * D^t - c*q^t - h*\left(x_o^t - D^t\right) - \nu*\left(x_{l-1}^t - D^t\right)^+, & \text{if } x_o^t \geq D^t,\\ p^t * D^t - c*q^t - u*\left(D^t - x_o^t\right), & \text{if } x_o^t < D^t.\end{cases} \tag{4}$$

For the lost-sales case, if $L = 0$, then

$$R^{t+1}=\begin{cases} p^t * D^t - c*q^t - h*\left(x_o^t + x_0^t - D^t\right) - \nu*\left(x_{l-1}^t - D^t\right)^+, & \text{if } x_o^t + x_0^t \geq D^t,\\ p^t * \left(x_o^t + x_0^t\right) - c*q^t - u*\left(D^t - x_o^t - x_0^t\right), & \text{if } x_o^t + x_0^t < D^t.\end{cases} \tag{5}$$

If $L > 0$, then

$$R^{t+1}=\begin{cases} p^t * D^t - c*q^t - h*\left(x_o^t - D^t\right) - \nu*\left(x_{l-1}^t - D^t\right)^+, & \text{if } x_o^t \geq D^t,\\ p^t * x_o^t - c*q^t - u*\left(D^t - x_o^t\right), & \text{if } x_o^t < D^t.\end{cases} \tag{6}$$

where inventory carried forward to the next period incurs a unit holding cost h, unmet demand incurs a unit penalty cost u, and ν is unit disposal cost. When fixed ordering cost K is considered, reward function above will subtract K if order quantity is not 0.

The sequence of events in period t is as follows:

(1) Based on the environment state E^t, $E^t \equiv (D^t, M^t, X^t)$, the agent selects an *action* A^t. Note that A^t is a vector, including ordering and pricing decisions. The order will be delivered at the beginning of period $t + L$; when $L = 0$ the order is delivered immediately.

(2) During period t, demand D^t arrives, which is discrete and stochastic depending on the selling price p^t, and is satisfied by the on-hand inventory as much as possible by the agent. Unsatisfied demand is either backlogged or lost; the remaining inventory with positive lifetime can be carried over to the next period.

(3) At the end of period t, the agent receives a reward R^{t+1}, which depends on the environment state and action A^t.

(4) At the beginning of period $t + 1$, the agent receives an order (if any), and the environment state is updated to E^{t+1} according to the *update rule*.

$$h: \left(E^t, A^t\right) \longrightarrow E^{t+1}. \tag{7}$$

For this joint pricing inventory problem, we introduce the notations in Table 1.

In this paper, we assume as in Chen et al. [2] that $c \leq u/1 - \gamma$, which eliminates the incentive to intentionally carry the back orders. We also assume that items with different lifetimes are charged the same price and that the back orders are met at cost c at the end of each planning period.

4. Deep Reinforcement Learning Methods

The objective of reinforcement learning is to learn a policy π that achieves near-optimal accumulated reward for the agent. Q-learning [31] is one widely used value iterative reinforcement learning method where the expected total discount rewards of state-action pairs can be approximated by a Q-function table based on the bellman equation, as shown in Function 7. Q-learning also has obvious limitation, that is, when there is a large state space, it is impractical and inefficient to record all the states and actions. Mnih et al. [32] extends Q-learning to Deep Q-network (DQN) which uses a neural network to approximate the Q-function table. DQN updates the parameters of the neural network by minimizing the difference between the predicted Q-values and the target Q- values, where the target Q-values are estimated by current

Table 1: Notations grouped by the elements of the RL problem.

p^t	Price charged by the agent for each item
q^t	Order quantity
D^t	Customer demand ($D^t(p^t)$ if depending on p^t)
l	Lifetime of the product, $0 < l < \infty$
L	Order lead time, $L < l$
K	Fixed order cost
c	Variable cost per item
u	Penalty cost per unmet item
h	Holding cost for per item left
v	Cost for each item disposed
x_i^t	Inventory position of age i, $0 \le i \le l-1$
x^t	Inventory position of all ages
x_o^t	On-hand inventory
R^{t+1}	The reward function of the agent

reward and predicted Q-values from the next state. Meanwhile, to avoid training instability caused by correlation between training data, a replay memory pool is used:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[R + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]. \tag{8}$$

As mentioned before, in our joint pricing and inventory control problem, the state of the agent is expressed by the inventory state X^t which integrates different ages and corresponding quantities. Here, the initial inventory state is X^0. In our proposed algorithm PAQ-DQN, there are two same neural networks with the same structure but different parameters θ and $\hat{\theta}$, respectively. We adopt the fixed Q-targets' policy in standard DQN. The neural network that predicts Q-values has the latest parameters, while the neural network that predicts target Q-values uses the old parameters. Each neural network has two hidden layers, and there are 128 neurons in each layer, we use the ReLU as the activation function. In each time period, based on Q-values from the neural network, the ε-greedy policy will be executed to select an action from the action space which contains a combination of ordering and pricing. After receiving the reward from the environment, the target Q-values are estimated by current rewards and discounted predicted Q-values from the next state, as shown in equation (9). The parameters of the network θ are updated by minimizing the difference between the predicted Q-values and the target Q-values, as shown in equation (10). After a fixed number of steps, assign the value of parameter θ to $\hat{\theta}$. The details of the algorithm named perishables integrate age and quantity deep Q-network (PAQ-DQN) are shown in Algorithm 1:

$$y^t = \begin{cases} R^{t+1}, & \text{if epoch terminates at step } t+1, \\ R^{t+1} + \gamma^* \max_{A'} \hat{Q}\left(X^{t+1}, A'; \hat{\theta}\right), & \text{otherwise,} \end{cases} \tag{9}$$

$$L(\theta) = E_{X_i^t, A_i^t, R_i^{t+1}, X_i^{t+1}}\left[\left(y^t - Q\left(X^t, A^t; \theta\right)\right)^2\right]. \tag{10}$$

The second reinforcement learning algorithm named perishables integrate age and quantity advantage actor-critic (PAQ-A2C). A2C is a method combining policy gradient and function approximation. Actor-critic (A2C) has two networks, one policy network, known as actor and used to output policy, and one value network, known as critic and used to evaluate the policy from actor. In our algorithm, both the policy network and value network have two hidden layers, and there are 128 neurons in each layer with the ReLU activation function. Especially, the activation function of the policy network output layer is the Softmax, which outputs the probability of each action being executed in the current state. In each time period, based on the current inventory state, an action will be executed by the policy network. After receiving the reward from the environment, the value network will evaluate this policy and output a *td_error*. The parameters of value network θ_v can be updated by Equation (11), where y^t is the target value calculated by equation (12). The policy network is updated by $\theta_p \leftarrow \theta_p + \alpha * \nabla_{\theta_p} J(\theta_p)$, where α is learning rate, and gradient $\nabla_{\theta_p} J(\theta_p)$ is shown in equation (13) where advantage function is estimated by equation (14). The details of the algorithm of proposed perishables integrate age and quantity advantage actor-critic (PAQ-A2C) are shown in Algorithm 2:

$$L(\theta_v) = \left[y^t - V^\pi\left(X^t; \theta_v\right)\right], \tag{11}$$

$$y^t = R^{t+1} + \gamma * V^\pi\left(X^{t+1}; \theta_v\right), \tag{12}$$

$$\nabla_{\theta_p} J\left(\theta_p\right) \approx \nabla_{\theta_p} \log \pi_{\theta_p}\left(A^t | X^t\right) \hat{A}\left(X^t, A^t\right), \tag{13}$$

$$\hat{A}\left(X^t, A^t\right) = R^{t+1} + \gamma * V^\pi\left(X^{t+1}; \theta_v\right) - V^\pi\left(X^t; \theta_v\right). \tag{14}$$

5. Experiments

In this section, we conduct simulation studies to evaluate the performance of our proposed reinforcement learning algorithms and investigate the positive effects of the proposed algorithms on the profit of dynamic pricing and the impacts of the key parameters. Ordering and pricing policy are also discussed in situation involving fixed ordering cost. In this experiment, we only show the discussions on the backlogging case, the discussions of the lost-sales case is carried out in Appendix.

The values of various parameters are set in Table 2. For simplicity, the value range of the price p and order quantity q are restricted to $[32, 37]$ and $[0, 31]$, respectively.

In the reinforcement learning method, the effect of hyperparameters on final performance is very important, so we need to set the variation rules for relevant parameters, exploration rate ε and learning rate α. We adopt ε-greedy policy here; ε is decreasing linearly, that is, search-then-convergence form in Darken et al. [33]:

$$\varepsilon_{\text{epoch}} = \frac{\varepsilon_0}{1+y}, \tag{15}$$

where $y = \text{epoch}^2/\varepsilon_{\text{decay}}$, ε_0 is the initial value of the ε, and $\varepsilon_{\text{decay}}$ is the decay parameter.

```
(1) Initialize replay memory pool $D$ to capacity $N$
(2) Use random weights $\theta$ to initialize the action-value function $Q$
(3) Initialize target action-value function $\widehat{Q}$ with weights $\widehat{\theta} = \theta$
(4) For $epoch = 1$ to number of $epochs$ do
(5) Reset the environment and initialize state $X^0$
(6) for $t = 1, T$ do
(7) With probability $\varepsilon$, select a random action $A^t$, otherwise select $A^t = \mathrm{argmax}_{A^t} Q(X^t, A^t; \theta)$ ($\epsilon$-greedy policy)
(8) Execute action $A^t$ and observe reward $R^{t+1}$ and $X^{t+1}$
(9) Store transition $(X^t, A^t, R^{t+1}, X^{t+1})$ in the replay memory pool $D$
(10) Set $X^{t+1} = X^t$
(11) Sample a minibatch of transitions $(X_i^t, A_i^t, R_i^{t+1}, X_i^{t+1})$, $\forall i = 1, \ldots, N$ from replay memory pool $D$
(12) Calculate the target Q-value by equation (9)
(13) Update the parameters of network $\theta$ by equation (10)
(14) Every C steps reset $\widehat{Q} = Q$
(15) end for
(16) end for
```

Algorithm 1: Perishables integrate age and quantity deep Q-network.

```
(1) Use random weights $\theta_p$ and $\theta_v$ to initialize the policy network and value network
(2) for $epoch = 1$ to number of $epochs$ do
(3) Reset the environment and initialize state $X^0$
(4) For $t = 1, T$ do
(5) Take action $A^t$ based on action probability $\pi_{\theta_p}(\cdot|X^t)$
(6) Execute action $A^t$ and observe reward $R^{t+1}$ and $X^{t+1}$
(7) Update the parameters $\theta_v$ of the value network by minimizing the loss function equation (11)
(8) Estimate advantage function by equation (14)
(9) Update the policy network parameters $\theta_p \leftarrow \theta_p + \alpha_p \nabla_{\theta_p} J(\theta_p)$, where $\nabla_{\theta_p} J(\theta_p)$ is calculated by equation (13)
(10) Set $X^{t+1} = X^t$
(11) end for
(12) end for
```

Algorithm 2: Perishables integrate age and quantity advantage actor-critic.

Table 2: The parameter values.

Parameter	Values	Description
T	30	Periods of the planning horizon
l	{2,3,4}	Lifetime of perishables
x^l	0	Initial inventory position of all ages
γ	0.9	Discount factor
L	{0, 1, 2}	Lead time
K	{25, 50}	Fixed ordering cost
c	22.5	Unit variable cost
h	0.22	Unit holding cost
u	10.78	Unit penalty cost
v	10	unit disposal cost
d^t	$84 - 2p$	Function of selling price
ε_0	1	Initial exploration rate
α	{0.01, 0.001, 0.0001}	Learning rate
$\varepsilon_{\mathrm{decay}}$	$\{1 \times 10^3, 1 \times 10^4, 1 \times 10^5\}$	Exploration rate decay parameter

5.1. Experiments on Dynamic Pricing with Positive Lead Time. Firstly, we conduct experiments for perishables' joint ordering and pricing with positive lead time (where $L = 1$). In order to examine the positive impact of dynamic pricing, we consider a fixed-price policy where the agent always takes the fixed best price which achieves the highest revenue. Let MEP and $\mathrm{MEP}_{\mathrm{FP}}$ be the expected mean epochs profits for the dynamic ordering and pricing policy and the fixed-price ordering policy, respectively, and MDC and $\mathrm{MDC}_{\mathrm{FP}}$ be the mean epochs disposal cost. After ten thousand simulations, we get the results in Table 3. Table 3 shows that PAQ-DQN achieves better performance than PAQ-A2C when lifetime is

Table 3: Results for dynamic pricing with positive lead time.

Method	Lifetime	MEP	MEP_{FP}	MDC	MDC_{FP}
PAQ-DQN	2	3242.104	2911.139	254.905	291.997
	3	4734.766	4455.814	40.439	110.146
	4	4840.443	4714.712	34.066	28.642
PAQ-A2C	2	3081.546	2305.394	232.000	573.749
	3	4513.786	4474.915	207.470	130.278
	4	4919.332	2278.324	49.802	748.916

2 and 3, but when lifetime is 4, they achieve almost the same results. Mean epochs profits and mean epochs disposal cost for two algorithms are almost all increasing and decreasing with lifetimes, which are in line with expectations, because the longer the lifetime is, the more similar it is to ordinary goods and perishables have more lifetime to sell out under the FIFO policy. From Table 3, it is easy to find out that it is better to adjust the price in a dynamic way so that the price can be adjusted according to the availability of inventory and the remaining life of the product and maximize the profits.

Table 4 shows the comparison between the tabular Q-learning and reinforcement learning methods on mean epoch profits and mean epoch disposal cost. From the table, we can see our proposed PAQ-DQN and PAQ-A2C obviously performs better than the Q-learning method. As we have mentioned before, Q-learning is a tabular method; it stores every state-action value in a table, but in our perishables inventory system, we considered the different ages, so the state space increases exponentially with lifetime, which is inefficient and impractical. Moreover, the amount of computing power and time required increase greatly with lifetime for the Q-learning method.

5.2. Experiments on the Performance of Proposed Algorithms. In this case, we compute the mean epoch profits for the optimal policy and proposed PAQ-DQN and PAQ-A2C with zero lead time. In particular, we set up an upper bound benchmark for this computation and define it as the optimal policy. The optimal policy takes the same price action as the PAQ-DQN and PAQ-A2C in each period, and its order quantity is always equal to the real demand D^t in each period, which means there is always no holding cost, penalty cost, and disposal cost for the planning horizon. Although there may be still some unreasonable place, this can be a useful metric to gauge the performance of the agent. Table 5 shows the computed results after twenty thousand simulations and $\text{MEP}_{\text{average}}$, $\text{MEP}_{\text{average}} = (\text{MEP}_{\text{optimal}} - \text{MEP})/T$, where T denotes the mean difference between the mean epochs profits from deep reinforcement learning methods and the mean epoch profits from the optimal policy. From the table, we can see our proposed algorithms achieve a good performance for three different lifetimes, where the benchmark is a loose upper bound from the real demand D^t and the difference from the average optimal profit is almost always less than the highest possible profit per unit, that is, $\text{MEP}_{\text{average}} \leq \overline{p} - c$. And the algorithm PAQ-A2C is slightly better than algorithm PAQ-DQN. Figure 1 shows the real epoch profits for the proposed PAQ-DQN and PAQ-A2C (in order to show the variation, we let the initial negative values as zero); from the figure, we can see that two methods quickly reached a relatively flat of profitability and PAQ-A2C showed more stable properties at the beginning of the learning process.

Figures 2–4 show the scatter plots of the profits difference between the optimal policy and the proposed PAQ-DQN algorithm for three different lifetimes. To better show the convergence rate, the figure is drawn on a log-log scale. From three figures, we can see three MEP differences begin to decrease rapidly after about fifty simulations; this demonstrates our deep reinforcement learning method works, the agent gradually learns how to order, and price is near optimal. Besides, the fitting lines in the figures are used to depict the convergence rate, and the following fitting line functions are for lifetime 2, 3, and 4. Here, we also carry out sensitivity analysis to investigate the effects of learning rate α and exploration parameter $\varepsilon_{\text{decay}}$ for the training of the proposed deep reinforcement learning methods, respectively. Figure 5 demonstrates the MEP for three different learning rates on PAQ-DQN and the learning rate α at 0.001 is the best for three different lifetimes, and α at 0.01 is very close to the best performance. Figure 6 shows the effects of exploration parameters $\varepsilon_{\text{decay}}$ on PAQ-DQN, and when the exploration parameter $\varepsilon_{\text{decay}}$ is 1×10^3, the agent gets a higher reward than the other two parameters. And the difference between the three parameters is very obvious. From the above two sensitivity analysis cases, the importance of hyperparameter is verified, and this is a common problem in deep learning. To show the expansibility of the algorithm, we also extend the distribution of the demand into an additive form in Chen et al. [2], where random term has a zero mean. By setting the random term which satisfies a uniform distribution in $[A, B]$, where A and B are symmetric and the absolute value is 2, we get a near-optimal performance with optimal rate 96.344%:

$$\log(\text{MEP}_{\text{diff}}) \approx -0.609\log(\text{epochs}) + 11.794\left(r^2 = 0.963\right), \tag{16}$$

$$\log(\text{MEP}_{\text{diff}}) \approx -0.737\log(\text{epochs}) + 12.445\left(r^2 = 0.986\right), \tag{17}$$

$$\log(\text{MEP}_{\text{diff}}) \approx -0.575\log(\text{epochs}) + 11.278\left(r^2 = 0.956\right). \tag{18}$$

5.3. Experiments on Dynamic Ordering and Pricing with No Fixed Ordering Cost. In this case, when the real epoch profits gradually become *stable* (*stable* means the real epoch profits

Table 4: Results.

Method	lifetime	Lead time	MEP	MDC
PAQ-DQN	2	0	5377.021	18.747
		1	3242.104	254.905
PAQ-A2C	2	0	5385.653	21.222
		1	3081.546	232.000
Q-learning	2	0	4800.622	31.376
		1	592.721	151.462

Table 5: MEP for proposed algorithm and optimal policy.

Method	Lifetime	MEP	$\text{MEP}_{\text{optimal}}$	$\text{MEP}_{\text{average}}$	$\text{MEP}/\text{MEP}_{\text{optimal}} * 100\%$
PAQ-DQN	2	5377.021	5691.355	10.477	94.477
	3	5514.710	5691.253	5.884	96.898
	4	5409.587	5664.803	8.507	95.495
PAQ-A2C	2	5385.653	5662.379	9.224	95.113
	3	5590.559	5674.625	2.802	98.519
	4	5428.072	5677.201	8.304	95.612

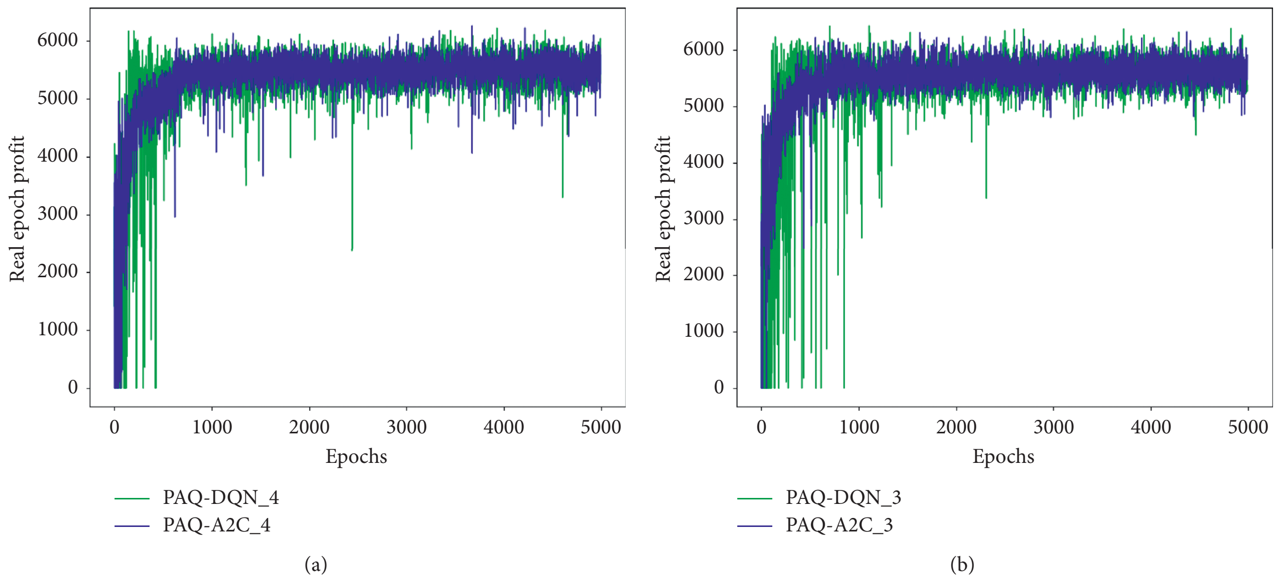


Figure 1: Real epoch profits for PAQ-DQN and PAQ-A2C, where the lifetime 3 and lifetime 4 are shown.

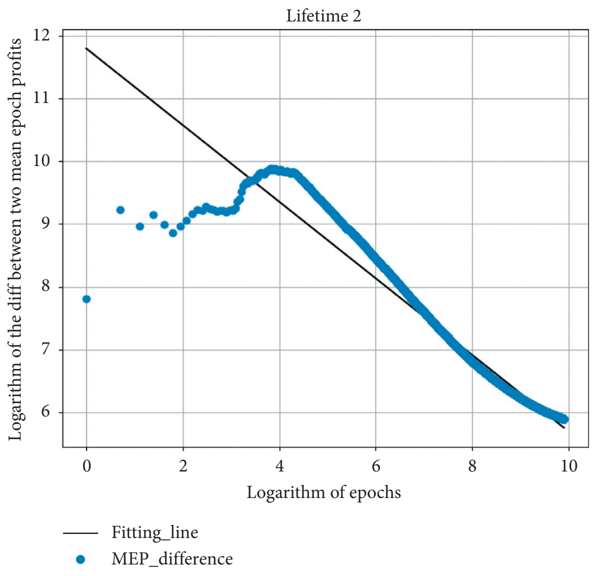


Figure 2: Log-log scale MEP difference.

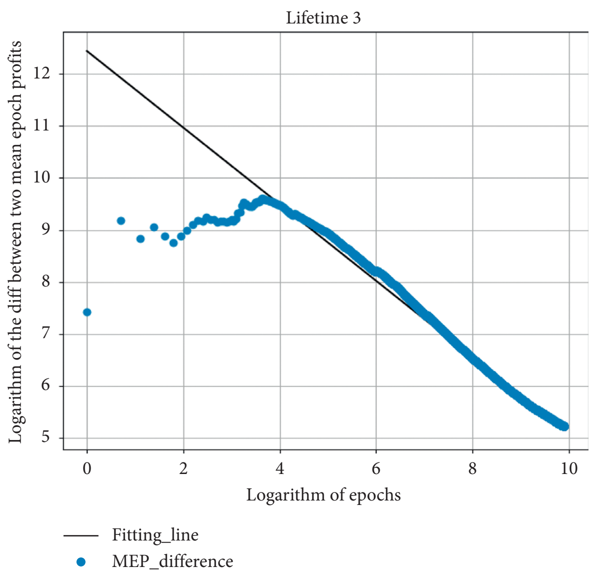


Figure 3: Log-log scale MEP difference.

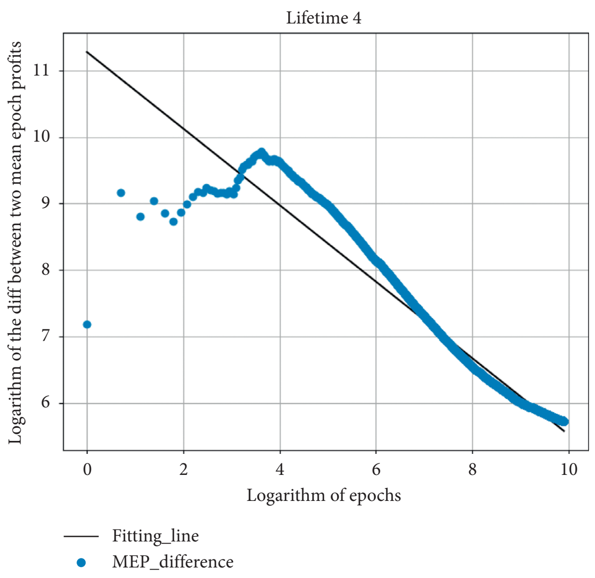


Figure 4: Log-log scale MEP difference.

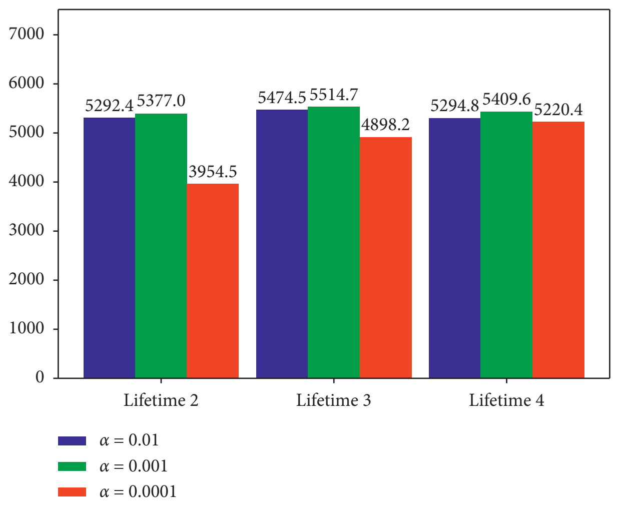


Figure 5: MEP for learning rates α.

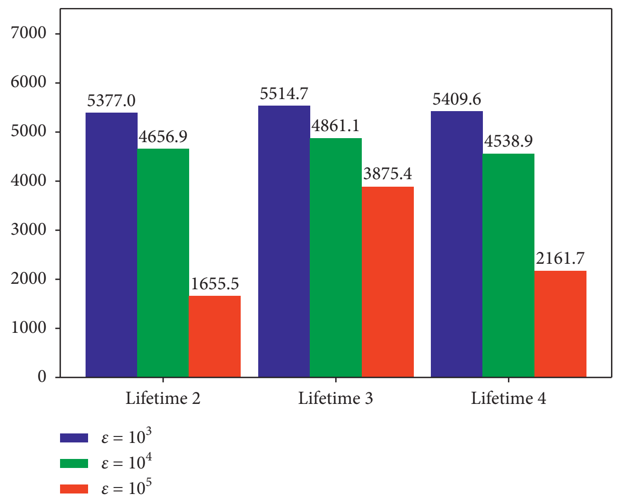


Figure 6: MEP for $\varepsilon_{\text{decay}}$.

always go up and down in a small fixed range over time) with the number of epochs, it indicates that the agent has learned a relatively stable state-to-action mapping relationship. In this section, we first extract the latest n epochs with stable mapping, denoted by dataset $\mathbf{p} = \{[x^{i1}, A^{i1}, \ldots, x^{it}, A^{it}, \ldots, x^{iT}, A^{iT}]_{i=1}^{n}\}$, where x^{it} is the inventory state vector for epoch i and period t. To facilitate discussion, we will use the fragment $\widetilde{\mathbf{p}}^t = \{[x^{it}, A^{it}]_{i=1}^{n}\}$ $(t = 1, \ldots, T)$ extracted from $\mathbf{p}$.

Chen et al. [2] has discussed the properties of optimal policies in the joint pricing and inventory system without fixed ordering cost. From the above settings and through our proposed reinforcement learning methods, when there is no fixed ordering cost, we get that the learned order quantity is nonincreasing in both outstanding and on-hand inventory levels. When $L = 0$, the learned price is always equal to the price that achieves highest expected revenue, and when $L > 0$, the learned price is most sensitive to the oldest on-hand inventory.

Figure 7 shows that the order quantity decreases with the inventory position and on-hand inventory. In order to show the sensitivity, we extract the fragment $\widetilde{\mathbf{P}}^5$ from $\mathbf{P}$ as an example, where $l = 4, L = 2$, and $n = 1000$. In the inventory state $X^5 = (x_1, x_2, x_3)$, where x_1 is the outstanding order and x_3 is the oldest on-hand inventory, we find that x_1 and x_2 are equal to a fixed value happens more than 500 times out of 1000, and when $x_1 = x_2$, the price decreases with the oldest on-hand inventory. The same results can be obtained from other fragments. In this setting, Figure 8 shows that the price decreases with the oldest on-hand inventory, which means when the oldest inventory increases, the agent tends to set a lower price.

5.4. Experiments on Dynamic Ordering and Pricing with Fixed Ordering Cost.

In this part, we will consider the case when there is a fixed ordering cost in this joint pricing and inventory system. We use our propose deep reinforcement learning algorithms to solve this case, and in order to measure the final performance, we set up a loose upper bound as our benchmark. In this benchmark, there are trade-offs between different costs. The price decision is supplied by algorithms. For ease of discussion and simplicity, we assume zero penalty costs and zero disposal costs to be achieved, which mean each demand will be met and each order will be sold within l period. We also assume that the initial inventory is zero; thus, the first order will always be placed at the beginning of the planning horizon. In particular, when $L > 0$, there may be a penalty cost at the beginning. D^t is the real demand in period t, $t = 1, \ldots, l, \ldots, T$. It is obviously unwise to order every period in this setting.

When $L = 0$, taking into account the width of finite lifetime l and the minimization of total cost, it is easy to see that the agent needs to place at least one order every l term and every order is just consumed by the next one. In the first l term, if $[(D^2 + \cdots + D^l) + (D^3 + \cdots + D^l) + \cdots + (D^{l-2} + D^{l-1}) + D^{l-1}] * h \leq K$, it only needs one order at the beginning of the first period and ordering quantity $q = D^1 + \cdots + D^l$. Morover, at some point t_o, whether to order depends on the time of last order $t_n, t_o - t_n \leq l$, if $[(D^{t_n+1} + \cdots + D^{t_o-1}) + (D^{t_n+2} + \cdots + D^{t_o-1}) + \cdots + D^{t_o-1}] * h \leq K$ and $[(D^{t_n+1} + \cdots + D^{t_o}) + (D^{t_n+2} + \cdots + D^{t_o}) + \cdots + D^{t_o}] * h > K$, it should order at point t_o and the order quantity for t_n is $q = D^{t_n} + \cdots + D^{t_o-1}$. When $L > 0$, taking into account the width of finite lifetime l and the minimization of total cost, it is easy to see that the agent needs to place at least one order every l term. In the first l term, there is a penalty cost, $(D^1 + \cdots + D^L) * u$, due to the lag of the order. At some point t_o $(t_o \neq 1)$, whether to order depends on the time of last order t_n and in order to make the subsequent penalty cost zero, $t_o - t_n \leq l - L$. If $[(D^{t_n+L+1} + \cdots + D^{t_o+L-1}) + (D^{t_n+L+2} + \cdots + D^{t_o+L-1}) + \cdots + D^{t_o+L-1}] * h \leq K$ and $[(D^{t_n+L+1} + \cdots + D^{t_o+L}) + (D^{t_n+L+2} + \cdots + D^{t_o+L}) + \cdots + D^{t_o+L}] * h > K$, it should order at point t_o, and when $t_n = 1$, the order quantity for t_n is $q = D^{t_n} + \cdots + D^{t_o+L-1}$; when $t_n \neq 1$, the order quantity for t_n is $q = D^{t_n+L} + \cdots + D^{t_o+L-1}$.

We consider two different fixed ordering costs K, $K \in \{25, 50\}$, two different penalty costs u, $u \in \{10.78, 4.18\}$ (corresponding, $u/(h+u) \in \{98\%, 95\%\}$), and two different

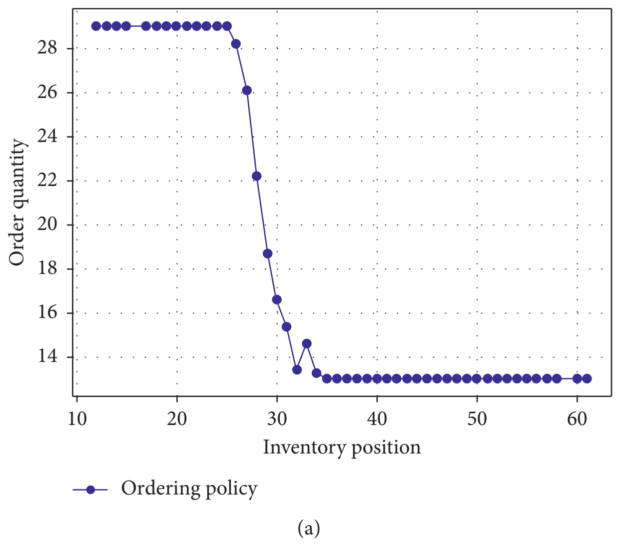
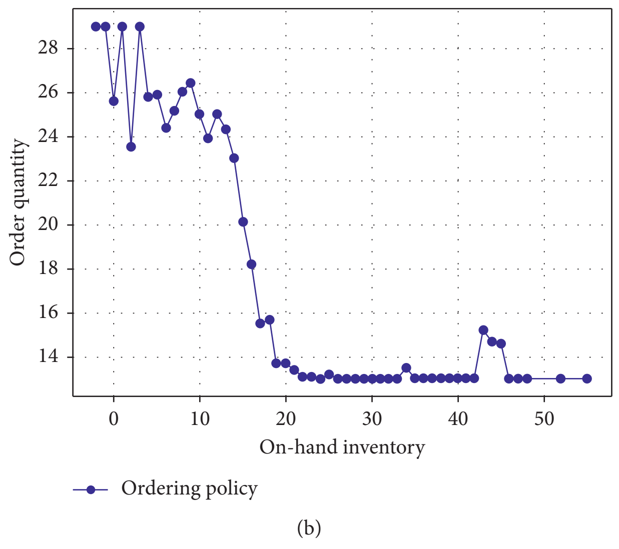


Figure 7: This statistic results are from $\tilde{\mathbf{p}}^5$, where $l = 4$ and $L = 2$. When $L = 2$, x^{it} is a vector and order quantity is the mean from the same inventory value.



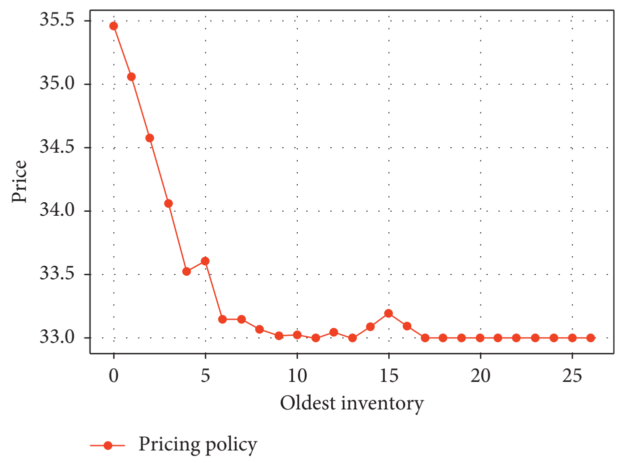


Figure 8: This statistic results are from $\tilde{\mathbf{p}}^5$, where $l = 4$ and $L = 2$. When $L = 2$, x^{it} is a vector and price is the mean from the same inventory value.

price-demand functions d^t, and the first one is shown in Table 2 and another one is $d^t = 380 - 10p$. Lifetime $l = 4$, lead time $L = 0, 1$, and a larger order action space is considered for $d^t = 380 - 10p$.

Under all of the above setting, we find that when $L = 0$, the convergent price is always the price that maximizes the expected revenue. This is in line with expectations. Same as the no fixed ordering cost case, the order quantity is nonincreasing in both outstanding and on-hand inventory levels. Table 6 shows the MEP results from PAQ-DQN and PAQ-A2C when the fixed ordering cost is 25 and 50 and price-demand function $d^t = 380 - 10p$ after thirty thousand simulations. From the table, we can see that, under the same conditions, the mean epoch profits MEP decreases with the lead time and the fixed ordering cost. When $L = 0$, algorithm PAQ-A2C performs better than PAQ-DQN, and when $L > 0$, our proposed PAQ-DQN performs better than PAQ-A2C. More interestingly, in our learned convergent policies, we find there exist one or two critical values in the inventory position in each period in each case when $L = 0$ and $L = 1$. We denote cv^t as the critical value in each period for the one critical value cases and cv_1^t and cv_2^t in each period for the two critical value cases, $cv_1^t < cv_2^t$. In the one critical value case, when $x^t < cv^t$, there will be a fixed order quantity q_1; when $x^t \geq cv^t$, the fixed order quantity is q_2. In the case of two critical values, when $x^t < cv_1^t$, the fixed order quantity is q_1; when $cv_1^t \leq x^t < cv_2^t$, the fixed order quantity is q_2; when $x^t \geq cv_2^t$, the fixed order quantity is q_3. For more details about obtaining the critical values, see Appendix.

Table 7 shows the MEP of learned policies from algorithm PAQ-DQN; from the table, we can see our learned policies achieve a higher optimal rate and are closer to the upper bound, compared to Table 6. Table 8 shows the MEP comparison between the learned policies and the algorithm PAQ-DQN; from the table, we can see our learned policies achieve a higher MEP and lower MDC, which means our learned policies are working well.

Table 6: MEP with fixed ordering cost.

Method	Lifetime	Lead time	K	$u/h+u$ (%)	MEP	$\text{MEP}_{\text{benchmark}}$	$\text{MEP}/\text{MEP}_{\text{benchmark}} * 100\%$
PAQ-DQN	4	0	50	98	15041.55	16172.39	93.01
		1	50	98	14484.18	15576.99	92.98
		0	25	98	15753.94	16507.79	95.43
		1	25	98	14544.00	15436.53	94.22
PAQ-A2C	4	0	50	98	15116.77	16178.22	93.44
		1	50	98	14223.13	15423.03	92.22
		0	25	98	15813.49	16502.16	95.83
		1	25	98	14896.70	15858.32	93.94

Table 7: MEP for the learned policies.

Lifetime	Lead time	K	$\text{MEP}_{\text{policies}}$	$\text{MEP}_{\text{benchmark}}$	$\text{MEP}_{\text{policies}}/\text{MEP}_{\text{benchmark}} * 100\%$
4	0	25	16013.56	16525.75	96.90
	1	25	14911.06	15540.62	95.95
4	0	50	15421.45	16197.45	94.10
	1	50	14771.21	15583.89	94.79

Table 8: Comparison for PAQ-DQN.

Lifetime	Lead time	K	MEP	$\text{MEP}_{\text{policies}}$	MDC	$\text{MDC}_{\text{policies}}$
4	0	25	15753.94	16013.56	2.45	0
	1	25	14544.00	14911.06	36.87	0
4	0	50	15041.55	15241.45	1.92	0
	1	50	14484.18	14771.21	97.67	56.97

Table 9: MEP for new state forms.

Lifetime	Lead time	K	$u/h+u$ (%)	$\text{MEP}_{\text{PAQ-DQN}}$	$\text{MEP}_{\text{PAQ-A2C}}$	$\text{MEP}_{\text{new-state}}$
3	0	50	98	15052.747	15126.553	15465.963
	1	50	98	13685.987	13544.115	13842.541
	2	50	98	8310.238	7018.234	9633.665
3	0	50	95	14937.285	14996.444	15604.507
	1	50	95	13894.832	13847.474	14317.453
	2	50	95	10470.134	9989.935	10299.061

Table 10: MEP for new state forms.

Lifetime	Lead time	K	$u/h+u$ (%)	$\text{MEP}_{\text{PAQ-DQN}}$	$\text{MEP}_{\text{PAQ-A2C}}$	MEP_{S^1}	MEP_{S^2}
3	0	50	95	14937.285	14996.444	15604.507	15309.898
			98	15052.747	15126.553	15465.963	15117.522

Table 11: Results for dynamic pricing with positive lead time.

Method	Lifetime	MEP	MEP_{FP}	MDC	MDC_{FP}
PAQ-DQN	2	2761.297	2648.036	371.397	371.135
	3	4832.157	4624.937	51.167	73.137
	4	4917.960	4787.649	28.737	42.665
PAQ-A2C	2	2739.912	2492.352	330.639	329.539
	3	4647.317	4490.300	94.006	102.618
	4	4807.079	4638.502	84.877	84.269

The above discussion is mainly based on the current inventory state. Next, we will try to add the historical inventory states and action information to the state to discuss its impact on the final performance. Here, we define the new state to be $S^t = (X^{t-L}, A^{t-L}, X^{t-L+1}, \ldots, X^t)$ when $L > 0$, and when $L = 0$, we also discuss the gradual influence of the addition of information in the state on the final performance. At the same time the dimension of the state accompanying the increase in information will also increase.

Table 9 shows the results after ten thousand simulations and there we add the inventory state and action from the previous period for the new state when $L = 0$. From the table,

Table 12: Results.

Method	lifetime	Lead time	MEP	MDC
PAQ-DQN	2	0	5394.735	21.519
		1	2761.297	371.397
PAQ-A2C	2	0	5367.027	16.139
		1	2739.912	330.639
Q-learning	2	0	4437.220	87.911
		1	<0	0

Table 13: MEP for proposed algorithm and optimal policy.

Method	Lifetime	MEP	$MEP_{optimal}$	$MEP_{average}$	$MEP/MEP_{optimal} * 100\%$
PAQ-DQN	2	5349.735	5670.955	10.707	94.336
	3	5574.282	5693.097	3.960	97.913
	4	5435.068	5685.511	8.348	95.595
PAQ-A2C	2	5367.027	5695.111	10.936	94.239
	3	5627.395	5686.337	1.964	98.963
	4	5504.487	5694.402	6.330	96.665

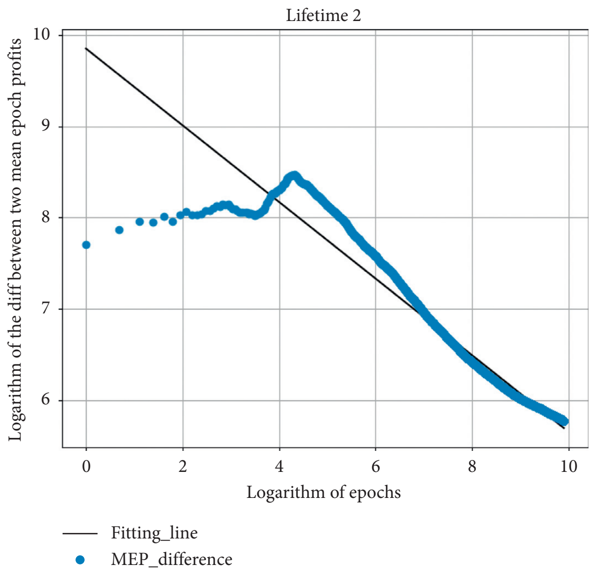


Figure 9: Log-log scale MEP difference for lifetime 2.

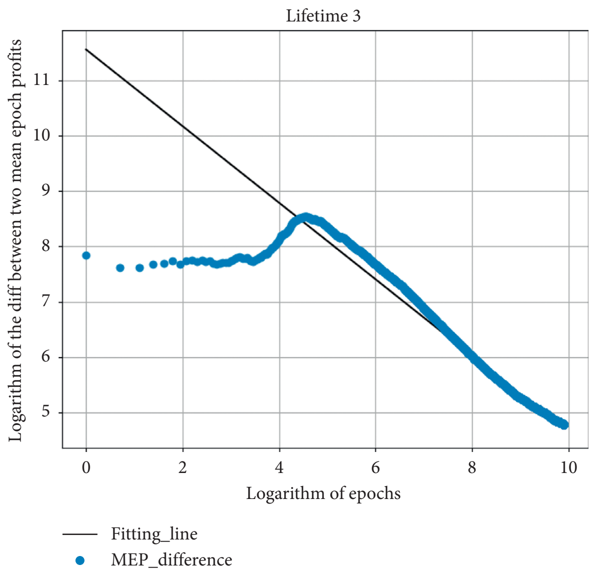


Figure 10: Log-log scale MEP difference for lifetime 3.

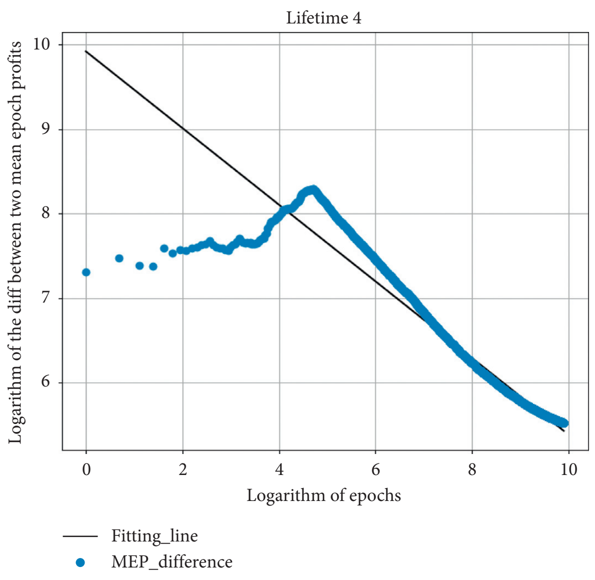


Figure 11: Log-log scale MEP difference for lifetime 4.

Table 14: Critical values for $L = 0$.

(K, L)	cv	1	2	3	4	5	6	7	8	9	10	11	12
(25, 0)	cv^1	−6	−6	−5	−6	−6	−6	−6	−7	−6	−6	−5	−5
	cv^2	22	23	23	24	23	22	24	23	24	23	23	24
		13	14	15	16	17	18	19	20	21	22	23	24
	cv^1	−6	−6	−6	−6	−5	−5	−6	−6	−5	−5	−7	−6
	cv^2	24	23	23	24	23	22	22	23	23	24	23	24
		25	26	27	28	29							
	cv^1	−5	−6	−5	−4	−5							
	cv^2	23	24	23	24	23							
(K, L)	cv	1	2	3	4	5	6	7	8	9	10	11	12
(50,0)	cv^1	−9	−10	−10	−11	−10	−9	−10	−10	−11	−10	−9	−11
	cv^2	12	13	13	13	14	13	12	13	13	13	13	13
		13	14	15	16	17	18	19	20	21	22	23	24
	cv^1	−10	−10	−10	−11	−10	−11	−10	−11	−9	−11	−11	−8
	cv^2	13	13	13	13	13	13	13	12	13	13	13	13
		25	26	27	28	29							
	cv^1	−9	−13	−10	−10	−10							
	cv^2	13	13	13	13	12							

Table 15: Critical values for $L = 1$.

(K, L)	cv	1	2	3	4	5	6	7	8	9	10	11	12
(25, 1)	cv^1	121	100	121	100	121	103	121	102	121	106	121	105
		13	14	15	16	17	18	19	20	21	22	23	24
	cv^1	121	108	121	110	121	114	121	116	121	116	121	118
		25	26	27	28	29							
	cv^1	121	121	121	121	121							
(K, L)	cv	1	2	3	4	5	6	7	8	9	10	11	12
(50, 1)	cv^1	62	62	63	63	64	65	63	64	63	64	64	64
		13	14	15	16	17	18	19	20	21	22	23	24
	cv^1	64	64	65	64	63	64	64	63	64	64	64	64
		25	26	27	28	29							
	cv^1	64	63	64	64	64							

Table 16: MEP with fixed ordering cost.

Lifetime	Lead time	K	$u/h+u$ (%)	MEP	$\text{MEP}_{\text{benchmark}}$	$\text{MEP}/\text{MEP}_{\text{benchmark}} * 100\%$
4	0	50	98	15378.025	16127.167	95.355
	1	50	98	14484.18	15576.99	92.98
4	0	25	98	15753.94	16507.79	95.43
	1	25	98	14844.676	15622.779	95.019

we can see that the new state contains more information almost all performs better than the single current inventory state, which means the current decisions of the agent are influenced by not only the current inventory state but also the inventory states of the previous periods. In Table 10, MEP_{S^1} and MEP_{S^2}, respectively, represent the inventory state and action information of the previous period and the previous two periods added to the current state. From the table, we find that the final performance did not get better and better with the continuous addition of the historical information, which also confirms that the dimensions of the state mentioned above continue to increase with the addition of information, which may have a negative impact on learning.

6. Conclusions

In this paper, we investigate a joint pricing and inventory control problem and obtain near-optimal pricing and replenishment policies for stochastic perishable inventory systems with positive lead time by deep reinforcement learning algorithms. Through our designed algorithms, we show that, in a perishable inventory control problem, the expected profit is maximized by adjusting the price according to the availability of inventory and the remaining lives of the items. We consider the case of no fixed ordering cost and the one involving a fixed ordering cost and find near-optimal policies for both cases. Our findings when a fixed ordering cost is involved contribute to the literature of inventory control for perishables, which has not been studied before. In this paper, we only focus on a single agent's joint pricing and inventory control problem. However, multiple agents are usually involved in supply chains, and their interactions may have a big impact on each agent's pricing and inventory decisions. Therefore, the study of the competition and cooperation of participants under complete and incomplete information is an interesting topic for future research.

Appendix

This section is for the discussion about lost-sales case in Section 5.

A. Experiment on Dynamic Pricing with Positive Lead Time

Table 11 shows that it is better to adjust the price in a dynamic way so that the price can be adjusted according to the availability of inventory and the remaining life of the product and maximize the profits. From Table 12, we can also see our proposed method PAQ-DQN achieves better performance than PAQ-A2C when lead time is positive and our proposed deep reinforcement learning methods obviously perform better than the Q-learning method.

B. Experiments on the Performance of Proposed Algorithms

In this case, we compute the mean epoch profits for the optimal policy and proposed PAQ-DQN and PAQ-A2C with zero lead time. Table 13 shows the computed results after twenty thousand simulations. From the table, we can see our proposed algorithms achieve a good performance for three different lifetimes, where the benchmark is a loose upper bound from the real demand D^t and the difference from the average optimal profit is almost always less than the highest possible profit per unit. And the algorithm PAQ-A2C is slightly better than algorithm PAQ-DQN. Figures 9–11 show the scatter plots of the profits' difference between the optimal policy and the proposed PAQ-DQN algorithm for three different lifetimes. To better show the convergence rate, the figure is drawn on a log-log scale. From three figures, we can see three MEP differences all begin to decrease rapidly after about ninety simulations; this demonstrates our deep reinforcement learning method works, the agent gradually learns how to order, and price is optimal. Besides, the fitting lines in the figures are used to depict the convergence rate, and the following fitting line functions are for lifetime 2, 3, and 4:

$$\log\left(\text{MEP}_{\text{diff}}\right) \approx -0.401\log(\text{epochs}) + 9.862\left(r^2 = 0.960\right), \tag{B.1}$$

$$\log\left(\text{MEP}_{\text{diff}}\right) \approx -0.677\log(\text{epochs}) + 11.531\left(r^2 = 0.982\right), \tag{B.2}$$

$$\log\left(\text{MEP}_{\text{diff}}\right) \approx -0.429\log(\text{epochs}) + 9.973\left(r^2 = 0.960\right). \tag{B.3}$$

B.1. Experiments on Dynamic Ordering and Pricing with no Fixed Ordering Cost. In this section, under the same settings as the backlogging case, we get the same results where the learned order quantity is nonincreasing in both outstanding and on-hand inventory levels. When $L = 0$, the learned price is always equal to the price that achieves highest expected revenue, and when $L > 0$, the learned price is most sensitive to the oldest on-hand inventory.

B.2. Experiments on Dynamic Ordering and Pricing with Fixed Ordering Cost. Firstly, we introduce the steps to get the critical values mentioned in the backlogging case. We first extract the latest n epochs with stable mapping, denote by dataset $\mathbf{p} = \left\{[x^{i1}, A^{i1}, \ldots, x^{it}, A^{it}, \ldots, x^{iT}, A^{iT}]_{i=1}^{n}\right\}$, where x^{it} is the inventory state vector for epoch i and period t. To facilitate discussion, we will use the fragment $\widetilde{\mathbf{p}}^{t} = \left\{[x^{it}, A^{it}]_{i=1}^{n}\right\}$ $(t = 1, \ldots, T)$ extracted from $\widetilde{\mathbf{p}}$. Based on the fragment $\widetilde{\mathbf{p}}^{t} = \left\{[x^{it}, A^{it}]_{i=1}^{n}\right\}$ $(t = 1, \ldots, T)$, we can observe the relationship between the inventory level and the order quantity and price. In the backlogging case, when $l = 4$, $L \in [0, 1]$, $K \in [25, 50]$, and $u/h + u = 98\%$, we get the following ordering and pricing policies. In each epoch, we set the first period as a zero initial inventory, so we cannot observe the critical values, and the following values are obtained from the second period. Tables 14 and 15 show the critical values in the different settings. When $L = 0$ and $K = 25$, the price is always 32 and $q_1 = 80, q_2 = 65$, and $q_3 = 35$. When $L = 0$ and $K = 50$, the price is always 32 and $q_1 = 95, q_2 = 65$, and $q_3 = 50$; when $L = 1$ and $K = 50$, the price is 32 except for the first period and $q_1 = 120$ and $q_2 = 0$; when $L = 1$ and $K = 25$, when inventory position is less than the critic value cv^1, the price is 32; otherwise, the price is 33, $q_1 = 75$, and $q_2 = 45$. In the lost-sales case, Table 16 shows the MEP for the different settings. And the same learned convergent policies structure as backlogging case can be obtained from this lost-sales case.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] I. Z. Karaesmen, A. Scheller-Wolf, and B. Deniz, "Managing perishable and aging inventories: review and future research directions," in *Planning Production and Inventories in the Extended Enterprise*, pp. 393–436, Springer, Berlin, Germany, 2011.

[2] X. Chen, Z. Pang, and L. Pan, "Coordinating inventory control and pricing strategies for perishable products," *Operations Research*, vol. 62, no. 2, pp. 284–300, 2014.

[3] S. K. Chaharsooghi, J. Heydari, and S. H. Zegordi, "A reinforcement learning model for supply chain ordering management: an application to the beer game," *Decision Support Systems*, vol. 45, no. 4, pp. 949–959, 2008.

[4] I. Dogan and A. R. Güner, "A reinforcement learning approach to competitive ordering and pricing problem," *Expert Systems*, vol. 32, no. 1, pp. 39–48, 2015.

[5] A. Kara and I. Dogan, "Reinforcement learning approaches for specifying ordering policies of perishable inventory systems," *Expert Systems with Applications*, vol. 91, pp. 150–158, 2018.

[6] J. Ke, F. Xiao, H. Yang, and J. Ye, "Optimizing online matching for ride-sourcing services with multi-agent deep reinforcement learning," 2019, http://arxiv.org/abs/1902.06228.

[7] S. A. M. Shihab, C. Logemann, D.-G. Thomas, and P. Wei, "Autonomous airline revenue management: a deep reinforcement learning approach to seat inventory control and overbooking," 2019, http://arxiv.org/abs/1902.06824.

[8] E. Presman and S. P. Sethi, "Inventory models with continuous and Poisson demands and discounted and average costs," *Production and Operations Management*, vol. 15, no. 2, pp. 279–293, 2006.

[9] O. Caliskan-Demirag, Y. Chen, and Y. Yang, "Ordering policies for periodic-review inventory systems with quantity-dependent fixed costs," *Operations Research*, vol. 60, no. 4, pp. 785–796, 2012.

[10] O. Alp, W. Tim Huh, and T. Tan, "Inventory control with multiple setup costs," *Manufacturing & Service Operations Management*, vol. 16, no. 1, pp. 89–103, 2013.

[11] A. T. Almaktoom, "Stochastic reliability measurement and design optimization of an inventory management system," *Complexity*, vol. 2017, Article ID 1460163, 9 pages, 2017.

[12] R. Azghandi, J. Griffin, and M. S. Jalali, "Minimization of drug shortages in pharmaceutical supply chains: asimulation based analysis of drug recall patterns and inventory policies," *Complexity*, vol. 2018, Article ID 6348413, 14 pages, 2018.

[13] C. Li, H. Guo, Y. Zhang, S. Deng, and Y. Wang, "An improved differential evolution algorithm for a multicommodity location-inventory problem with false failure returns," *Complexity*, vol. 2018, Article ID 1967398, 2018.

[14] X. Gan, S. P. Sethi, and L. Xu, "Simultaneous optimization of contingent and advance purchase orders with fixed ordering costs," *Omega*, vol. 89, pp. 227–241, 2019.

[15] S. Nahmias and W. P. Pierskalla, "Optimal ordering policies for a product that perishes in two periods subject to stochastic demand," *Naval Research Logistics Quarterly*, vol. 20, no. 2, pp. 207–229, 1973.

[16] S. Nahmias, "Optimal ordering policies for perishable inventory-II," *Operations Research*, vol. 23, no. 4, pp. 735–749, 1975.

[17] B. E. Fries, "Optimal ordering policy for a perishable commodity with fixed lifetime," *Operations Research*, vol. 23, no. 1, pp. 46–61, 1975.

[18] S. Nahmias, "Perishable inventory theory: a review," *Operations Research*, vol. 30, no. 4, pp. 680–708, 1982.

[19] G. P. Prastacos, "Blood inventory management: an overview of theory and practice," *Management Science*, vol. 30, no. 7, pp. 777–800, 1984.

[20] M. E. Ferguson and O. Koenigsberg, "How should a firm manage deteriorating inventory?," *Production and Operations Management*, vol. 16, no. 3, pp. 306–321, 2007.

[21] Q. Chen, Q. Xu, and W. Wang, "Optimal policies for the pricing and replenishment of fashion apparel considering the effect of fashion level," *Complexity*, vol. 2019, Article ID 9253605, 12 pages, 2019.

[22] Y. Li, A. Lim, and B. Rodrigues, "Note-pricing and inventory control for a perishable product," *Manufacturing & Service Operations Management*, vol. 11, no. 3, pp. 538–542, 2009.

[23] C. Li and M. Lu, "Joint price and inventory optimization under minimax regret," *SSRN Electronic Journal*, 2017.

[24] Y. Li, B. Cheang, and A. Lim, "Grocery perishables management," *Production and Operations Management*, vol. 21, no. 3, pp. 504–517, 2012.

[25] S. Bhattacharyya, V. Snasel, A. Hassanien, S. Saha, and B. Tripathy, *Deep Learning: Research and Applications, De Gruyter Frontiers in Computational Intelligence*, De Gruyter, Berlin, Germany, 2020, https://books.google.com/books?id=yEj2DwAAQBAJ.

[26] I. Giannoccaro and P. Pontrandolfo, "Inventory management in supply chains: a reinforcement learning approach,"

International Journal of Production Economics, vol. 78, no. 2, pp. 153–161, 2002.

[27] C. Jiang and Z. Sheng, "Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6520–6526, 2009.

[28] Z. Sui, A. Gosavi, and L. Lin, "A reinforcement learning approach for inventory replenishment in vendor-managed inventory systems with consignment inventory," *Engineering Management Journal*, vol. 22, no. 4, pp. 44–53, 2010.

[29] M. H. F. Zarandi, S. V. Moosavi, and M. Zarinbal, "A fuzzy reinforcement learning algorithm for inventory control in supply chains," *International Journal of Advanced Manufacturing Technology*, vol. 65, no. 1–4, pp. 557–569, 2013.

[30] R. Rana and F. S. Oliveira, "Dynamic pricing policies for interdependent perishable products or services using reinforcement learning," *Expert Systems with Applications*, vol. 42, no. 1, pp. 426–436, 2015.

[31] C. J. C. H. Watkins, *Learning from delayed rewards*, Ph.D. thesis, 1989.

[32] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[33] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in *Proceedings of the 1992 IEEE Workshop on Neural Networks for Signal Processing II*, pp. 3–12, IEEE, Helsingoer, Denmark, 1992.