Expert Systems With Applications 91 (2018) 150–158

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Reinforcement learning approaches for specifying ordering policies of perishable inventory systems

Ahmet Kara, Ibrahim Dogan*

Erciyes University, Industrial Engineering Department, Kayseri, Turkey

ARTICLE INFO

Article history:
Received 23 May 2017
Revised 23 August 2017
Accepted 24 August 2017
Available online 30 August 2017

Keywords:
Reinforcement learning
Inventory management system
Simulation-based optimization
Ordering management
Perishable item

ABSTRACT

In this study, we deal with the inventory management system of perishable products under the random demand and deterministic lead time in order to minimize the total cost of a retailer. We investigate two different ordering policies to emphasize the importance of the age information in the perishable inventory systems using Reinforcement Learning (RL). Stock-based policy replenishes stocks according to the stock quantities, and Age-based policy considers both inventory level and the age of the items in stock. The problem considered in this article has been modeled using Reinforcement Learning and the policies are optimized using Q-learning and Sarsa algorithms. The performance of the proposed policies compared with similar policies from the literature. The experiments demonstrate that the ordering policy which takes into account the age information appears to be an acceptable policy and learning with RL provides better results when demand has high variance and products has short lifetimes.

1. Introduction

One of the concerns of the companies producing perishable products is to have an effective inventory management system to improve the customer service and provide competitive advantages in the current global marketplace. However there are a number of challenges, such as stochastic customer demand, perishable nature of the products and need to trace the age of the products along the supply chain (Kouki & Jouini, 2015). In this study, we deal with a retailer's ordering replenishment problem under stochastic customer demand, limited shelf lifetime and fixed lead time situation to find the best balance between the outdating quantity and the shortage quantity. Fresh agri-products, dairy products, pharmaceuticals and human blood are typical examples of perishable products in which the age of the products are significantly important. For example platelet, one of the blood components can be used effectively for only 5 days. If the platelet stocks cannot be managed effectively by the age and quantity information, it is possible to observe a lot of stock-outs or wastage of platelets.

Classical inventory systems which do not consider the age of the products is not suitable for perishables and perishable inventory systems require different approaches (Tekin, Gürler, & Berk, 2001). Perishable inventory policies with stochastic demand have been commonly modeled using only quantity of stocks information. With the development of technology, age-based policies using the information about the age of stocks are evaluated in detail for the perishable items (Broekmeulen & van Donselaar, 2009). Heuristics approaches are proposed in the literature due to the modeling and solution complexity of the exact models. We suggest and evaluate approximate aged-based replenishment policies which take into account both the quantity and the age of the stocks. Moreover, this paper demonstrates that reinforcement learning methods can be used to solve the inventory control problem of perishable items with uncertain customer demand.

In the area of machine learning, Reinforcement learning offers the advantage of solving the complex sequential decision-making problems based on learning from the previous knowledge. Reinforcement learning provides a dynamic learning against the changing environment and no need to pre-determined model of environment (Rana & Oliveira, 2015). A variety of problems such as job-shop scheduling (Zhang & Dietterich, 1995), revenue management optimization (Gosavi, Ozkaya, & Kahraman, 2007), inventory management in the supply chain (Jiang & Sheng, 2009) and goal-regulation in manufacturing systems (Shin, Ryu, & Jung, 2012) are formulated using reinforcement learning under non-stationary and unknown conditions. In this article, we apply reinforcement learning approaches for indicating the importance of the age information of the perishable inventory system. The essential contribution of our research is the use of the Q-learning and Sarsa algorithm based on reinforcement learning to specify the near-optimal ordering replenishment policy of perishable products with stochastic customer demand and lead time.

* Corresponding author.
E-mail addresses: ahmet.kara@erciyes.edu.tr (A. Kara), idogan@erciyes.edu.tr (I. Dogan).

http://dx.doi.org/10.1016/j.eswa.2017.08.046
0957-4174/

The paper is organized as follows. In the Section 2, we provide related literature on the replenishment learning approach and the inventory management system of perishable items. Section 3 generally introduces the replenishment learning model. In the Section 4, the inventory replenishment models and proposed algorithms based on reinforcement learning are described. Details of the conducted experiments are explained in Section 5. The results of computational experiments, discussion and conclusion are presented in Section 6.

2. Literature review

In recent years, comprehensive studies related to the integration of reinforcement learning model and inventory control in the supply chain have been presented in the literature. Reinforcement learning methods have been implemented to specify near-optimal ordering policies in the entire supply chain.

Kim, Jun, Baek, Smith, and Kim (2005) suggested two inventory-control models with a nonstationary and unknown customer demand consists of centralized model and decentralized model. They employed a reinforcement learning approach called the action-value method for satisfying a predetermined target service level during the lead time. Chaharsooghi, Heydari, and Zegordi (2008) presented an inventory control system with the case of uncertain lead-times and uncertain customer demand to determine ordering policies of each echelon in the supply chain. The proposed model is formulated as a reinforcement learning approach so as to minimizing the total inventory cost including holding cost and backorder cost.

Dogan and Güner (2015) concentrated on the ordering and pricing problems of the supply chain with a multi-retailer environment and formulated the problem using a reinforcement learning method called Q-learning. Kim, Kwon, and Baek (2008) considered a learning model which updating the reward values of each action at each decision period for the supply chain with two-echelon under unstable demand environment. A similar learning model called asynchronous action-reward learning has been applied by Kwak, Choi, Kim, and Kwon (2009). It is discussed by Rana and Oliveira (2015) that how reinforcement learning may be implemented to estimate accurately the optimal pricing policies of perishable items and services utilizing.

Recently in the literature, reinforced learning has been employed to solve different problems. Almahdi and Yang (2017) suggested the recurrent reinforcement learning method in order to optimize financial investments. Zhou, Hao, and Duval (2016) use the reinforcement learning based local search (RLS) approach to solve grouping problems that are NP-hard. A Q-learning algorithm based on reinforcement learning for the train rescheduling problem has been addressed by Šemrov, Marsetič, Žura, Todorovski, and Srdic (2016). Okdinawati, Simatupang, and Sunitiyoso (2017) applied reinforcement learning to formulate to reduce transportation costs, raise visibility and develop agility in Collaborative Transportation Management. Mannion, Mason, Devlin, Duggan, and Howley (2016) proposed a Multi-Agent Reinforcement Learning (MARL) approach to a dispatch problem.

In this study we handle the issue of inventory management of perishable product taking into account stochastic customer demand and fixed lead time. The detailed reviews have been published for perishable inventory management such as Bakker, Riezebos, and Teunter (2012), Beliën and Forcé (2012), Karaesmen, Scheller-Wolf, and Deniz (2011). In order to obtain the near-optimal inventory ordering policy of perishable items are studied with many different approaches. Haijema, van der Wal, and van Dijk (2007) provided a dynamic programming and simulation-based approach to manage the platelet inventory control at the blood bank. A Markov Decision Process is formulated to model the ordering policies of pharmaceutical items by Ana, Ivy, and King (2008). Hemmelmayr, Doerner, Hartl, and Savelsbergh (2010) proposed two solution model consist of an integer programming and a Variable Neighborhood Search approach. Our study is formulated using reinforcement learning to determine near-optimal inventory control policies of perishable products.

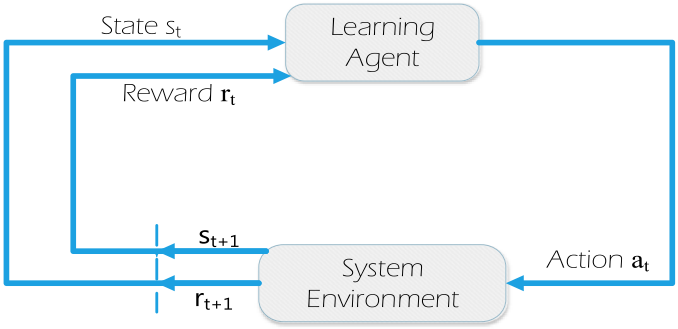

Fig. 1. The interaction of agent and environment (Sutton & Barto, 1998).

With respect to the supply chain inventory management, different models have been investigated using reinforcement learning. Pontrandolfo, Gosavi, Okogbaa, and Das (2002) considered production and distribution functions of global supply chain with multiple stages assuming a single item. The near-optimal procurement and distribution policy has been derived by a reinforcement learning model called Semi-Markov Average Reward technique (SMART). Kwon, Kim, Jun, and Lee (2008) focused on the solution of two significant situations consist of a non-stationary customer demand and a large state space. They presented a case-based myopic reinforcement learning algorithm for resolving inventory management problem of the supply chain with two-echelons. Jiang and Sheng (2009) studied on inventory management of a multi-agent supply chain structure. It has been suggested in Sun and Zhao (2012) that Q-learning is modeled as a reinforcement learning approach for specifying ordering policies of the supply chain with five stages.

3. Reinforcement learning

Reinforcement learning is a stochastic dynamic programing approach that concentrates on the concept of trial-and-error learning through the interaction of a decision-maker called agent and a dynamic environment. The structure of this approach generally consists of four basic elements: agent, state, action and reward (Das, Gosavi, Mahadevan, & Marchalleck, 1999). Fig. 1 depicts that agent-environment interaction in reinforcement learning approaches.

In each period, the learning concept in a reinforcement learning models takes place as below. When the system has a state s_t in time period t, agent chooses an action a_t among possible actions. The environment brings about a numerical reward r_{t+1} for a chosen action. The learning agent in each time period chooses an action with either the highest reward value based on its past experiences (exploitation) or a random action (exploration). The aim is to specify the optimal state-action pair that optimizes the long-run reward.

The basic issue of reinforcement learning is to resolve the problem of balancing the exploitation and exploration (Sutton & Barto, 1998). Initially, the learning agent frequently prefers the exploration strategy because it does not have enough details regarding environment. In the following phase, the agent mostly exploits the current information and the action with the highest reward value is preferred. Methods of action selection such as E3 technique (Kearns & Singh, 2002), the external-source technique (Smart & Kaelbling, 2000) and search-then-converge tech-

nique (Darken, Chang, & Moody, 1992) have been suggested and discussed in the literature for equalizing exploration and exploitation. In this study, we apply the search-then-converge procedure.

Reinforcement learning is mainly the combination of Monte Carlo method and dynamic programming and the approaches such as Q-learning (Watkins, 1989) and Sarsa algorithms (Rummery & Niranjan, 1994; Singh & Sutton, 1996) have been known to discover near-optimal solutions. The important task of these algorithms is to find $Q(s, a)$ values. $Q(s, a)$ is the expected reward value taking the action a when the system has in state s. This paper is proposed and compared the Q-learning and Sarsa algorithms to manage the perishable inventory control system in a stochastic environment.

4. Problem description and modeling

In this part, we identify characteristics of the perishable inventory management problem with a single product and stochastic demand. At the same time, it is demonstrated how Q-learning and Sarsa algorithms may be utilized to obtain a near-optimal ordering policy for the traditional order-up-to and stock-age dependent replenishment policies.

4.1. Perishable inventory management model

In this model, we study a single-product and a single-echelon inventory system for a retailer. We assume that the retailer sells a single perishable product to meet customer demands and place his order from a supplier with an infinite capacity. The retailer reviews stocks each period and receive orders after a fixed lead time of L periods. Each unit of the product has a constant and deterministic lifetime and it is equal to m periods. The lifetime of perishable item m is assumed to be larger than lead time L, $m > L$. The aging process starts when the products leave from the supplier. As a result, we follow up and update the remaining shelf life of stocks on hand and in-transit orders.

Customer demands are independent, identically distributed and it is determined by Gamma distributions with a scale parameter θ and a shape parameter k. The mean and the variance of Gamma distribution are $\mu = k*\theta$ and $\sigma\hat{}2 = k*\theta\hat{}2$, respectively. We presume that demand is satisfied from the oldest product on stock using the FIFO policy. The unmet demand is lost and is punished by a shortage cost, c_s, due to the opportunity loss and the erosion of customer goodwill. The unconsumed products left without remaining lives are discarded and the outdating cost, c_o, occurs.

The sequence of events for retailer in each period as follows. 1—Retailer receives in-transit orders and its inventory is updated. 2—Customer demand is satisfied from stocks and the demand that is not fulfilled from the stocks is lost. 3—The remaining lifetime of the products and the quantity of inventory is updated and the outdated products are discarded. 4—Retailer places on order according to its replenishment policy.

In this study, we compare stock-age based policy which takes into account both the inventory level and the remaining lifetime of stocks to the classical order-up-to policy without the age information. The parameters of the both policies are computed by Q-learning and Sarsa algorithms based on reinforcement learning model. The total inventory cost including the shortage cost resulting from the unsatisfied demands and the outdating cost resulting from discarded products is accepted as the performance measurement.

To compare the performance of the Q-learning and Sarsa algorithms, the same inventory replenishment policies including a stock-age based (called 2S) and a quantity dependent (called base policy) are modeled optimized using genetic algorithm. 2S policy considers the total remaining life of the items on stock and in transit. The total available remaining life is checked to place an order. The base policy is a classical order-up-to policy taking into account quantity of inventory.

4.2. Reinforcement learning model of perishable inventory problem

We describe the main elements of reinforcement learning consisting of the state variable, the action variable and the reward function for a near-optimal ordering policy of perishable inventory management.

4.2.1. State variable

The system state variable provides the necessary information for the learning agents' decision making process. There are two different states to show the stock-age dependent (S_1) and quantity based (S_2) policies in this study. S_1 vector contains information regarding the total remaining life of the products on hand and in transit. The other S_2 vector describes only the inventory position.

$$S_1(t) = [\mathrm{IP}_t,\ \mathrm{AI}_t] \tag{1}$$

$$S_2(t) = [\mathrm{IP}_t] \tag{2}$$

where $S_1(t)$ represent the state variable of the stock-age dependent policy at time step t. $S_2(t)$ denotes the state variable of the quantity based policy the time period t. IP_t is the inventory position at time step t and AI_t denotes the total remaining life of the products on hand and in transit.

In order to obtain a near-optimal replenishment policy of perishable inventory management, IP_t element of $S_2(t)$ vector are discretized the form of $0 \le \mathrm{IP}_t \le 100$. The elements of $S_1(t)$ vector are discretized the form of $0 \le \mathrm{IP}_t \le 20$ and $0 \le \mathrm{AI}_t \le 5$, respectively.

4.2.2. Action variable

The action variable expresses the decision of learning agent. The action variable contains information order quantity.

$$a_t = [\ OQ_t] \tag{3}$$

where OQ_t denotes the amount of order the time period t. a_t vector is discretized the form of $0 \le a_t \le 40$.

4.2.3. Reward function

The proposed perishable inventory management aims to minimize the total inventory cost including the shortage cost and the outdating cost. Thus the reward function is determined as the total inventory cost. The reward function is determined as follows:

$$r_t = c_o * TO_t + c_s * TS_t \tag{4}$$

where r_t denotes the total inventory cost at time step t. c_o and c_s represent the unit outdating and shortage costs, respectively. TS_t and TO_t are the quantity of unmet demand and outdated quantities at time period t.

To overcome the problem of balancing exploration and exploitation, this study applies the search-then-converge procedure suggested by Darken et al. (1992). The learning rate α_t and the exploration rate ε_t are used balancing exploration and exploitation. These parameters are updated with regard to the search-then-converge procedure as follows.

$$\varphi_t = \frac{\varphi_0}{1+y} \tag{5}$$

where $y = \frac{t^2}{\partial_\gamma + t}$. The learning rate α_t and the exploration rate ε_t are updated using Eq. (5). Where ϕ_t denotes the values of ε_t and α_t at the time period t. ϕ_0 and ∂_γ are constants to use for smooth decaying purposes.

Q-learning and Sarsa algorithms are suggested to solve the perishable inventory management problem modeled by reinforcement

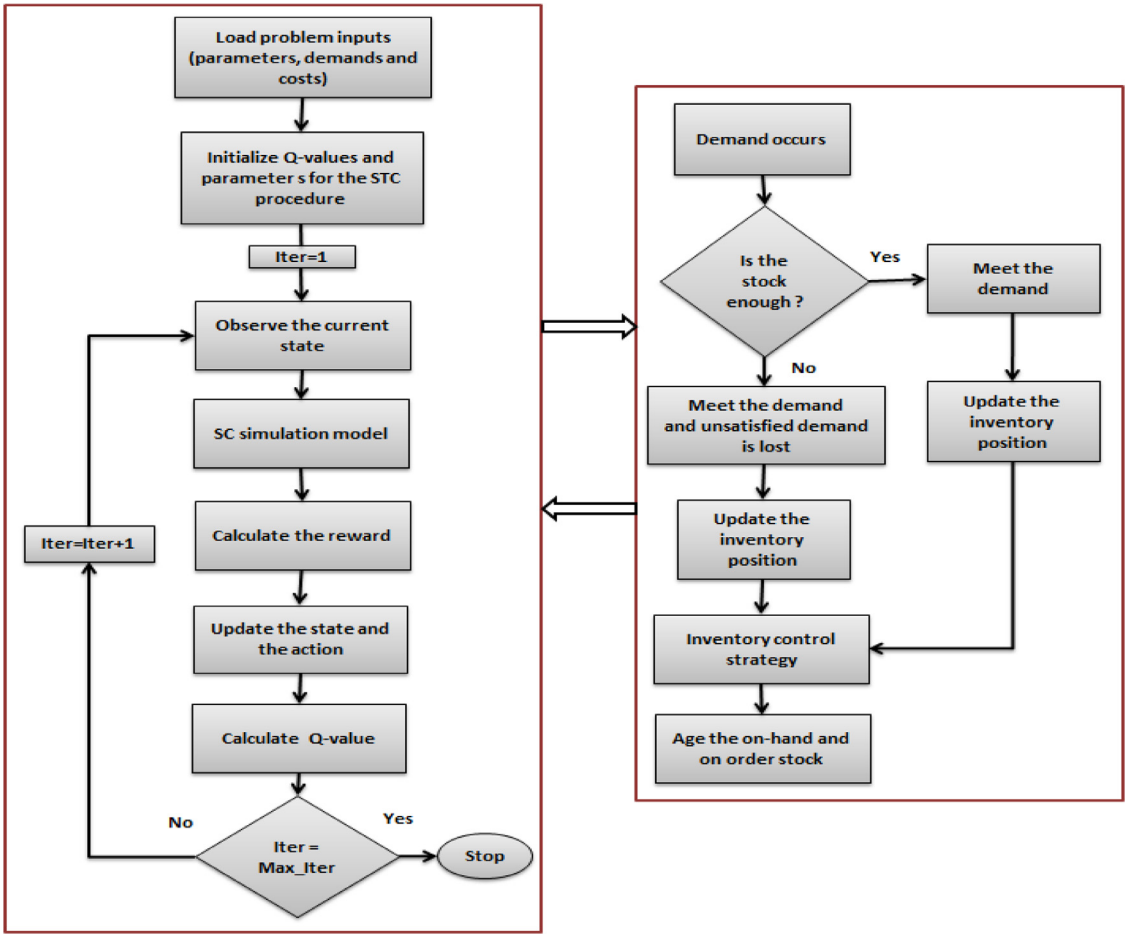

Fig. 2. Basics of the simulation with reinforcement learning.

learning. Fig. 2 demonstrates the main structure of the proposed Q-learning and Sarsa algorithms. Q-values, $Q(s,\ a)$ contain the immediately reward taking an action a in state s.

$$Q(s,\ a) \leftarrow Q(s,\ a) + \alpha\left[r + \gamma \max_{a'} Q\left(s,\ a'\right) - Q(s,\ a)\right] \tag{6}$$

$$Q(s,\ a) \leftarrow Q(s,\ a) + \alpha\left[r + \gamma Q\left(s,\ a'\right) - Q(s,\ a)\right] \tag{7}$$

Q-values of every state-action pair $Q(s, a)$ are stored for the Q-learning and Sarsa algorithms and updated throughout the learning proses. In the each time period, $Q(s,\ a)$ updating rule for Q-learning and Sarsa is conducted with Eq. (6) and Eq. (7), respectively. In Eqs. (6) and (7), α denotes the learning rate and r is the reward value. γ denotes the discount factor, $0 < \gamma < 1$. The different initial $Q(s, a)$ values are explored in simulation. Sutton and Barto (1998) indicate that different initial values can be used a way of exploration. In general the initial $Q(s, a)$ values are set to 0, ±5, ±10 plus a uniform random value between 0 and 1. Initial values other than zero allow algorithm to select different actions at the beginning with facilitating exploration and the random values can be used to break the ties.

The basics of the simulation are shown in Fig. 2. The left column applies the logic of the RL learning as shown in Fig. 2. The result of the several repetitive experiments of the inventory policies at the second column is supplied as reward to the RL at each iteration, and RL tries to maximize these rewards by determining the appropriate ordering quantity based on the state knowledge of the agent. Then the new action values are supplied to the inventory simulation to identify the rewards and these rewards again fed to the RL. These cycles repeated until the convergence of the RL or the maximum number of iterations being performed. Fig. 3 depicts the steps of Q-learning and Sarsa algorithms.

Table 1
Parameter values for the simulation experiment.

Parameters	Values
Product lifetime $\boldsymbol{m}$	{4, 5, 8}
Gamma distribution shape and scale parameter ($\boldsymbol{k}$, $\boldsymbol{\theta}$)	$\left\{\begin{array}{l}(k=4,\theta=5),\ \mu=20,\sigma^2=100\\(k=2,\theta=10),\ \mu=20,\sigma^2=200\\(k=1,\theta=20),\mu=20,\sigma^2=400\end{array}\right\}$
Cost Ratio $\boldsymbol{c_o/c_s}$	{0.5, 1, 2}
Lead time $\boldsymbol{L}$	{1, 2}

5. Experimental details

We conduct the simulation studies to evaluate the performance of the proposed algorithms based on reinforcement learning and to understand the importance of stock-age dependent policy for perishable inventory systems. The stock-level dependent and the stock-age dependent policies for perishable products are solved using Q-learning, Sarsa and genetic algorithms. In the numerical experiments, the total average cost was considered as a measure of performance.

We examine the effects of replenishment policies by changing the values of demand variability, product lifetime, lead time and ordering cost and shortage cost ratio. The details of the experimental factors are given in Table 1. In the experimental design,

1) Let time step $t = 0$.
   a) Initialize Q-values, $Q(s, a)$ for all s and a.
   b) Initialize the related parameters for the Search-Then-Converge (STC) procedure. Observe the current state and set to $i \in s$. Start system simulation.
2) *While* $t < T_{max}$ *do*
   a) Calculate α_t and ε_t according to STC procedure.
   b) Select an action a with regard to STC procedure.
   c) Simulate the chosen action. Let the state at the next time be j. Receive an immediate reward as a result of taking action a in state i, $r(i, j, a)$.
   d) Update $Q(i, a)$ using Eq. 1 or Eq. 2

   Q-learning algorithm: $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s,a') - Q(s,a)\right]$

   Sarsa algorithm: $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s,a') - Q(s,a)]$

   d) Set current state i to new state j, and $t \leftarrow t + 1$.

Fig. 3. Q-learning and Sarsa algorithms for solving the perishable inventory problem.

Table 2
The constant values of Search-Then-Converge procedure.

Constants	Values
Decay parameter of exploration rate, $\boldsymbol{\varepsilon_t}$	1×10^{11}, 1×10^{12}
Decay parameter of learning rate, $\boldsymbol{\alpha_t}$	1×10^{11}, 1×10^{12}
Discount factor, $\boldsymbol{\gamma}$	0.90, 0.97
Initial learning rate, $\boldsymbol{\alpha_0}$	0.40, 0.60, 0.80
Initial exploration rate, $\boldsymbol{\varepsilon_0}$	0.40, 0.60, 0.80

shelf lives of the perishable products are assumed to be 4, 5 and 8 periods so that different degree of the perishability is considered in the analysis. It is presumed that the demand of each period is generated from a gamma distribution with mean, μ, equal to 20. By changing the scale and shape parameters of the gamma distribution, different levels of variation in the demand is obtained to analyze the effects of demand variation. The outdating cost and the shortage cost is set to 1 and 2 so that the cost ratio, $\frac{c_o}{c_s}$, equal to 0.5, 1 and 2.

The decay and the initial parameters of learning rate, exploration rate and discount factors are used in search-then-converge procedure to balance exploration and exploitation. As illustrated in Table 2, the different values are used to determine the effect on learning.

For perishable inventory systems, we analyze six situations:

- GA-Base: stock-level dependent policy modeled using genetic algorithm.
- Sarsa-Base: stock-level dependent policy modeled using Sarsa algorithm.
- QL-Base: stock-level dependent policy modeled using Q-learning algorithm.
- GA-Age: stock-age dependent policy modeled using genetic algorithm.
- Sarsa-Age: stock-age dependent policy modeled using Sarsa algorithm.
- QL-Age: stock-age dependent policy modeled using Q-learning algorithm.

The genetic algorithm was run with a population size of 70, a generation size of 200 and replication of each population with 10,000 days in order to obtain the improved parameters. After finding the optimized parameters, each setting was simulated 50 times with 250,000 periods.

5.1. Analysis of results

We present and investigate the results acquired from the computational experiments for inventory management of perishable products. Comparison of the outcomes was made according to the average cost performance. Table 3 compares all policies with GA-Base policy and reports the average percentage cost changes. Positive values represent the cost improvements comparing to GA-Base policy. The first observations from the results are that the age based policies performs better than the quantity based policies. Table 3 shows that the age information of the stocks has significant impact on the inventory policy and inventory cost. The cost differences decrease with higher values of product lifetimes. In other words the value of age information decreases when the product have higher life-cycles. The highest cost improvement is observed when the value of the demand variance is high, the product lifetime is low, the lead time is high and the cost ratio is low. On the other hand, we observed mixed results when comparing the performance of the algorithms. RL based algorithms performs better with high variance and with small product lifetime values. Sarsa-Age policy gives the highest improvements on the average with lower values of product lifetime and higher demand variance. Cost improvement is 10.05% with Sarsa-Age policy when the value of product lifetime is 4. Fig. 4 also shows the average lost, perishable and total costs for policies across for all experiments. Sarsa-Age policy results in the lowest average total and lost costs. However GA-Age policy produces the lowest cost for average perishable cost.

Fig. 5 shows the comparison of magnitude of different factors' main effects for Sarsa-Age Policy through the experiments. Similar to the analysis above, for the higher values of demand variance and lead time and the lower values for product lifetime the average total cost increases.

Table 4 compares the results of the Sarsa-Age policy with other policies in more detail based on the average cost performance across the 54 experiments varied different values of the parameters. The column labeled as "count" represents the how many times the Sarsa-Age policy performs better than the other policy out of the 18 experiments if the parameter has three levels or out of the 27 experiments if the parameter has two levels. For example, Sarsa-Age policy performs better than the GA-Base policy in 14 out of 18 experiments (77.8%). Table 4 shows that the ordering policy taking into account the age information performs better than the quantity based policy in 47 out of 54 (87%) experiments. When the lifetime of the products decreases and the variability of demand increases, similar to the Table 3, the stock-age dependent ordering policy indicates better results.

Sarsa-Age performs better than Q-Age and GA when the demand variance is high and the products are highly perishable with small product lifetime. In other words Sarsa performs better when the environment is extremely dynamic. We believe that it is related to how the method updates the Q values during learning phase. Sarsa is an on-policy method where algorithm considers and explores the policy while moving. On the other hand, Q learn-

Table 3
The average cost differences of policies compared to the GA-Base policy (%).

Parameter	Level	Number of experiments	QL-Base (%)	Sarsa-Base (%)	GA-Age (%)	QL-Age (%)	Sarsa-Age (%)
Demand Variance	100	18	1.99	−5.09	6.85	6.24	5.06
	200	18	3.92	1.49	7.78	7.89	7.82
	400	18	5.20	3.14	6.98	7.73	8.25
Product lifetime m	4	18	6.80	3.91	8.52	9.94	10.05
	5	18	3.50	1.04	7.05	7.06	7.85
	8	18	−5.85	−9.63	0.40	−3.74	−5.90
Cost ratio c_o/c_s	0.5	18	4.05	2.68	7.52	8.18	8.02
	1	18	4.69	1.65	7.16	7.35	8.11
	2	18	4.56	0.28	6.88	7.10	7.05
Lead time L	1	27	3.37	0.09	4.43	3.64	3.85
	2	27	4.96	2.37	8.70	9.72	9.79

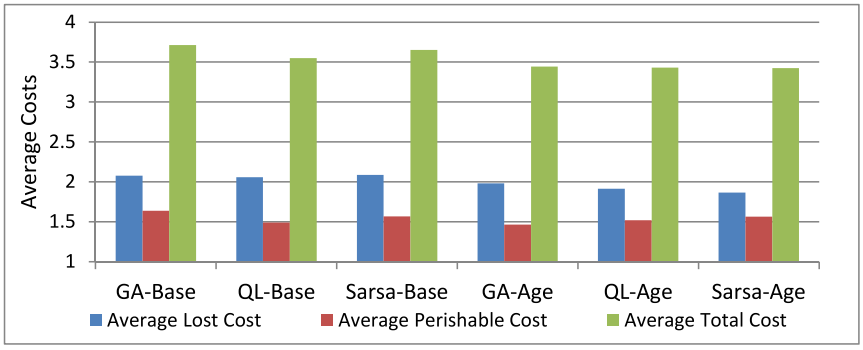

Fig. 4. Average lost, perishable and total costs for all policies.

Table 4
The average cost performances of Sarsa-Age algorithm compared to the other algorithms.

Parameter	Level	Sarsa-Age/ GA-Base		Sarsa-Age/ QL-Base		Sarsa-Age/ Sarsa-Base		Sarsa-Age/ GA-Age		Sarsa-Age/ QL-Age	
		Count	%	Count	%	Count	%	Count	%	Count	%
Demand Variance	100	8	44.4%	8	44.4%	13	72.2%	5	27.8%	6	33.3%
	200	12	66.7%	12	66.7%	17	94.4%	9	50.0%	9	50.0%
	400	14	77.8%	16	88.9%	17	94.4%	13	72.2%	14	77.8%
Product lifetime m	4	17	94.4%	15	83.3%	18	100.0%	15	83.3%	12	66.7%
	5	15	83.3%	14	77.8%	17	94.4%	11	61.1%	13	72.2%
	8	2	11.1%	7	38.9%	12	66.7%	1	5.6%	4	22.2%
Cost ratio c_o/c_s	0.5	11	61.1%	11	61.1%	15	83.3%	8	44.4%	6	33.3%
	1	12	66.7%	14	77.8%	15	83.3%	10	55.6%	13	72.2%
	2	11	61.1%	11	61.1%	17	94.4%	9	50.0%	10	55.6%
Lead Time L	1	14	51.9%	13	48.1%	23	85.2%	9	33.3%	13	48.1%
	2	20	74.1%	23	85.2%	24	88.9%	18	66.7%	16	59.3%

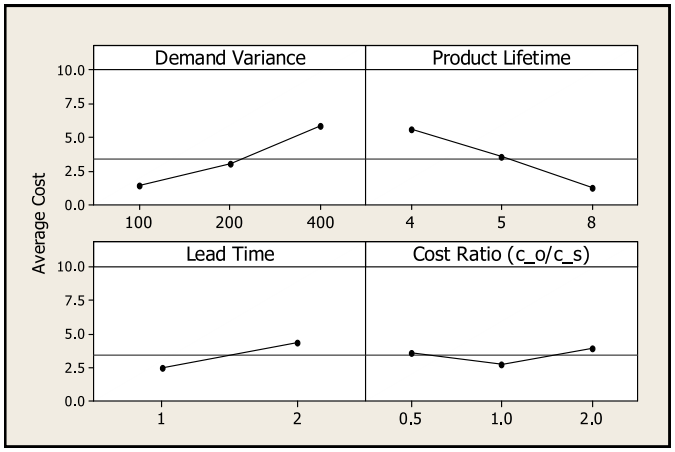

Fig. 5. Main effects for Sarsa-Age policy.

ing is an off-policy method where the agent learns the optimal policy without the policy followed (Sutton & Barto, 1998). The difference between these methods can be seen in the updating equation in Q-learning with max operator. In reality the update learning rules for both SARSA (Eq. (7)) and Q-learning (Eq. (6)) is minimal, however, the effects of the algorithms can be significant. Furthermore, we believe that the higher values of the product lifetime and the lead time cause delayed rewards in RL and this complicates the learning phase to adapt stable and efficient policies.

We also examine the effects of the initial and decay parameters on RL performance. The results are compared according to the achievement of learning process with different values of the discount factor (γ), decay parameters of learning (α_t) and exploration rates (ε_t), and the initial values of learning (α_0) and exploration rates (ε_0) as shown in Table 2. Fig. 6 compares the mean total cost obtained by QL-Base and QL-Age for different parameter values of discount factor, initial values of learning and exploration rates across 54 experiments. It can be observed that the learning parameters have a more significant effect on QL-Base. In all cases,

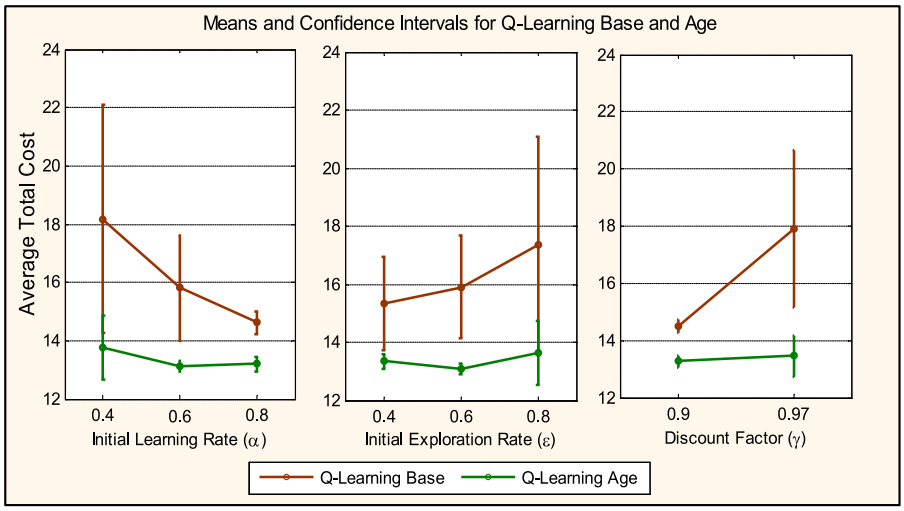

Fig. 6. The effect of different parameters on average total cost for Q-learning algorithm.

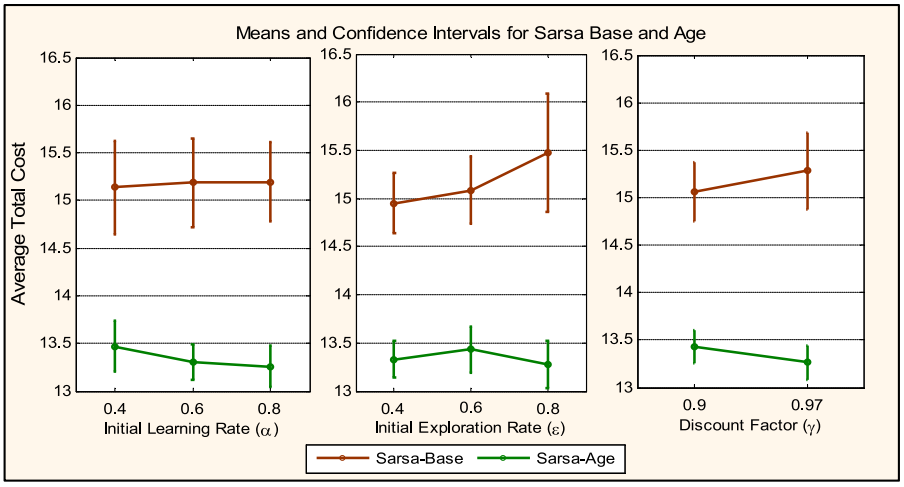

Fig. 7. The effect of different parameters on average total cost for Sarsa algorithm.

the average total cost of QL-Age is lower compared with QL-Base. The performance of learning for QL-Base gives better results when the initial exploration rate and discount factor is lower and the initial learning rate is higher. On the other hand, we revealed that QL-Age has generally similar results for different parameter values. Fig. 6 depicts that QL-Age policy is less sensitive and more robust to the changes in discount factor, and initial values of learning and exploration rates. Similarly, Fig. 7 illustrates the effect of learning parameters on the mean total costs derived from Sarsa-Base and Sarsa-Age algorithms. It is found that the smaller values of initial exploration rate and discount factor result in lower average costs for Sarsa-Base algorithm. However, the mean total cost decreases with the higher values of initial learning rate and discount factor for Sarsa-Age algorithm.

Fig. 8 shows the performance of Sarsa and Q-Learning algorithms on the average costs related to the converge characteristics. In reality, the learning phase takes longer than simulation time in Fig. 8. Rather, we scaled the x-axes and showed the average of consecutive periods during the learning phase to better explore the differences among polices. It is evident from the Fig. 8 that initially Sarsa-Age algorithm performs better, then Sarsa-Base policy performs better but eventually Sarsa-Age performs better than Sarsa-Age policy. Sarsa-Age policy has larger state space than the Sarsa-Base policy. Therefore, Sarsa-Age policy explores more than the Sarsa-Base and this results in a worse performance at the beginning and a better performance at the end. Furthermore, the state information related to the age in Sarsa-Age policy provides valuable information and results in lower costs. Similarly for Q learning, firstly QL-Base improves faster than the QL-Age and performs better, but eventually QL-Age performs better than the QL-Base. Fig. 9 shows the convergence results of the outdating, shortage and total costs for one of the experiment during the learning phase. Initially, RL starts with random policy with larger costs. We can observe that costs significantly reduce once RL learns better policies. Here, the tackle for the RL is to find the better policies to balance the costs of shortage and outdating. This figure clearly shows that outdating cost plays significant role in total cost and it is considerably reduced during the learning.

Fig. 10 shows the learned action-value function, $Q(s,a)$, for one of the experiment using QL-Base policy. The vertical axes shows the state information (s) which is only inventory position and horizontal axes shows the action space (a) which is the ordering quantity in this case. Although the problem on hand is a minimization problem, we employ negative rewards values which allow us to still use maximization updating rule. As a consequence, the steady-state $Q(s,a)$ values are resulted in negative values. After learning has taken place, it has been concluded that it is better not to place large orders when the inventory position is high since these

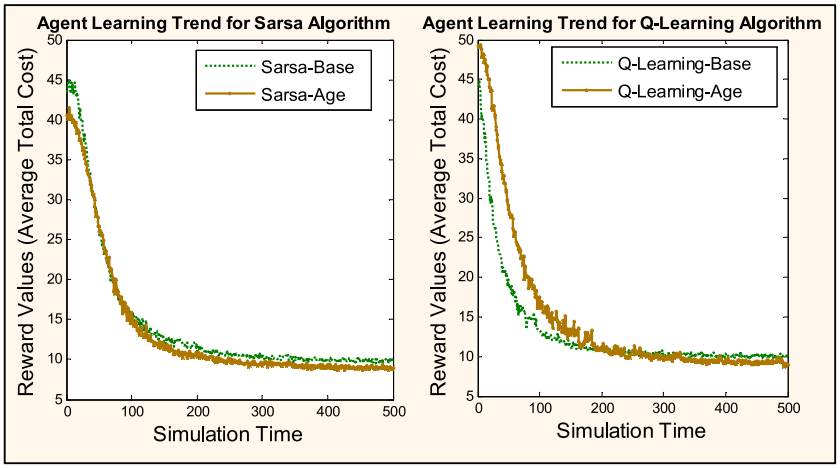

Fig. 8. Q-learning and Sarsa algorithms learning trend.

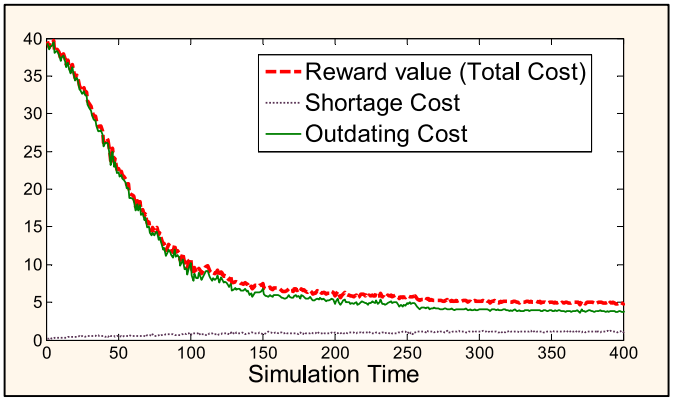

Fig. 9. The trend of shortage cost, outdating cost and total cost during the learning for Sarsa-Age.

state action values have smaller $Q(s, a)$ values. On the other hand, when the inventory position is low, $Q(s, a)$ values suggest to release larger orders. Furthermore, it is evident from the figure that the learned action-value function shows smooth color transitions and that is a sign of successful exploitation and exploration balance and the accomplished learning.

6. Conclusion remarks

We investigate the inventory management problem obtaining a near-optimal replenishment policy for a perishable product with a stochastic demand and a fixed lead time. Our study proposed two different ordering replenishment policies including stock-age dependent policy and quantity dependent policy. The ordering policies are modeled using Q-learning and Sarsa algorithms based on reinforcement learning approach and the genetic algorithm. In this article, we conducted the cost analysis for the different values of demand variability, lead time, cost ratio and product lifetime. Our computational results demonstrated that the ordering policy con-

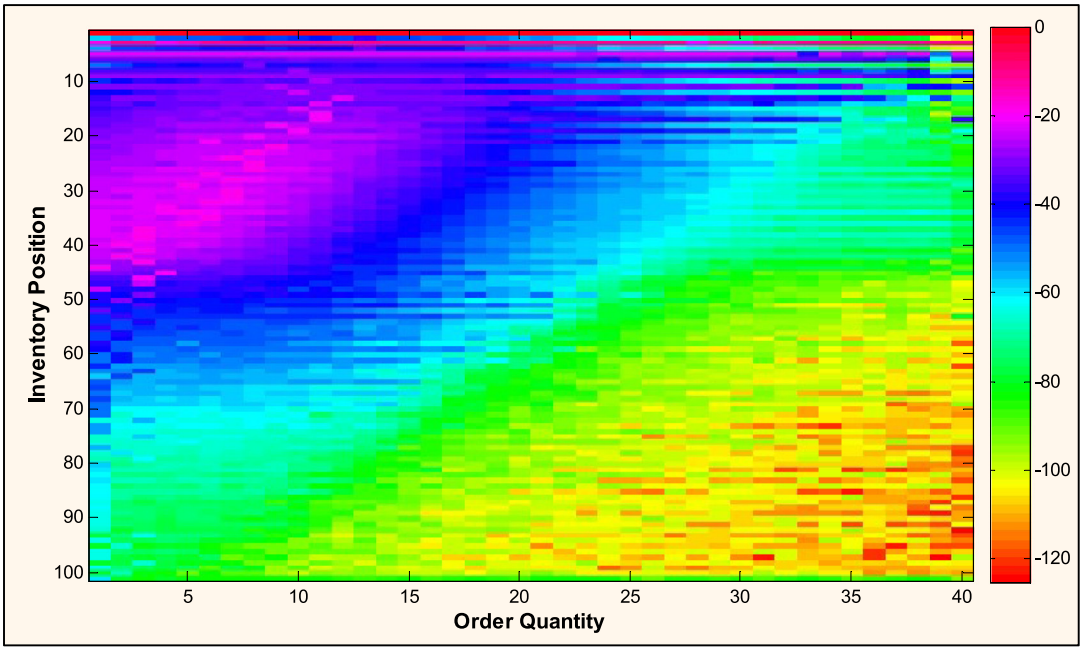

Fig. 10. Q-values for learning agent.

sidering both the age information and the inventory quantity gave better results compared with the quantity dependent policies using reinforcement learning and genetic algorithms. In the ordering policy taking into account the information about age of stock, we observe that Sarsa algorithm is better in terms of cost performance compared to the other algorithms. Furthermore, Q-learning algorithm for the base policy indicated better results compared to Sarsa algorithm and genetic algorithm.

Furthermore we analyze how the discount factor, learning and exploration rates affect the learning performance. The results show that interaction of initial-decay values of learning rate and interaction of initial-decay values of exploration rate have an effect on the learning. The proposed approach can be extended to a multi-agent and multi-echelon supply chain network. Furthermore, the non-stationary and unknown customer demand can be considered to evaluate the influence on the quality of learning and the perishable inventory management problem.

References

Almahdi, S., & Yang, S. Y. (2017). An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications, 87*, 267–279. https://doi.org/10.1016/j.eswa.2017.06.023.

Ana, V. P. R., Ivy, J. S., & King, R. E. (2008). A simulation-based approach for inventory modeling of perishable pharmaceuticals. In *Proceedings – Winter simulation conference* (pp. 1532–1538). https://doi.org/10.1109/WSC.2008.4736234.

Bakker, M., Riezebos, J., & Teunter, R. H. (2012). Review of inventory systems with deterioration since 2001. *European Journal of Operational Research, 221*(2), 275–284. https://doi.org/10.1016/j.ejor.2012.03.004.

Beliën, J., & Forcé, H. (2012). Supply chain management of blood products: A literature review. *European Journal of Operational Research, 217*(1), 1–16. https://doi.org/10.1016/j.ejor.2011.05.026.

Broekmeulen, R. A. C. M., & van Donselaar, K. H. (2009). A heuristic to manage perishable inventory with batch ordering, positive lead-times, and time-varying demand. *Computers and Operations Research, 36*(11), 3013–3018. https://doi.org/10.1016/j.cor.2009.01.017.

Chaharsooghi, S. K., Heydari, J., & Zegordi, S. H. (2008). A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems, 45*(4), 949–959. https://doi.org/10.1016/j.dss.2008.03.007.

Darken, C., Chang, J., & Moody, J. (1992). Learning rate schedules for faster Stochastic gradient search. In *Neural networks for signal processing - Proceedings of the IEEE workshop* (pp. 3–12). https://doi.org/10.1109/NNSP.1992.253713

Das, T. K., Gosavi, A., Mahadevan, S., & Marchalleck, N. (1999). Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science, 45*(4), 560–574. https://doi.org/10.1287/mnsc.45.4.560.

Dogan, I., & Güner, A. R. (2015). A reinforcement learning approach to competitive ordering and pricing problem. *Expert Systems, 32*(1), 39–48. https://doi.org/10.1111/exsy.12054.

Gosavi, A., Ozkaya, E., & Kahraman, A. F. (2007). Simulation optimization for revenue management of airlines with cancellations and overbooking. *OR Spectrum, 29*(1), 21–38. https://doi.org/10.1007/s00291-005-0018-z.

Haijema, R., van der Wal, J., & van Dijk, N. M. (2007). Blood platelet production: Optimization by dynamic programming and simulation. *Computers & Operations Research, 34*(3), 760–779. https://doi.org/10.1016/j.cor.2005.03.023.

Hemmelmayr, V., Doerner, K. F., Hartl, R. F., & Savelsbergh, M. W. P. (2010). Vendor managed inventory for environments with stochastic product usage. *European Journal of Operational Research, 202*(3), 686–695. https://doi.org/10.1016/j.ejor.2009.06.003.

Jiang, C., & Sheng, Z. (2009). Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system. *Expert Systems with Applications, 36*(3–2), 6520–6526. https://doi.org/10.1016/j.eswa.2008.07.036.

Karaesmen, I. Z., Scheller-Wolf, A., & Deniz, B. (2011). Managing perishable and aging inventories: Review and future research directions. In International Series in Operations Research and Management Science, 151, (pp. 393–436) https://doi.org/10.1007/978-1-4419-6485-4_15.

Kearns, M., & Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning, 49*(2–3), 209–232. https://doi.org/10.1023/A:1017984413808.

Kim, C. O., Jun, J., Baek, J. K., Smith, R. L., & Kim, Y. D. (2005). Adaptive inventory control models for supply chain management. *International Journal of Advanced Manufacturing Technology, 26*(9–10), 1184–1192. https://doi.org/10.1007/s00170-004-2069-8.

Kim, C. O., Kwon, I. H., & Baek, J. G. (2008). Asynchronous action-reward learning for nonstationary serial supply chain inventory control. *Applied Intelligence, 28*(1), 1–16. https://doi.org/10.1007/s10489-007-0038-2.

Kouki, C., & Jouini, O. (2015). On the effect of lifetime variability on the performance of inventory systems. *International Journal of Production Economics, 167*, 23–34. https://doi.org/10.1016/j.ijpe.2015.05.007.

Kwak, C., Choi, J. S., Kim, C. O., & Kwon, I. H. (2009). Situation reactive approach to vendor managed inventory problem. *Expert Systems with Applications, 36*(5), 9039–9045. https://doi.org/10.1016/j.eswa.2008.12.018.

Kwon, I. H., Kim, C. O., Jun, J., & Lee, J. H. (2008). Case-based myopic reinforcement learning for satisfying target service level in supply chain. *Expert Systems with Applications, 35*(1–2), 389–397. https://doi.org/10.1016/j.eswa.2007.07.002.

Mannion, P., Mason, K., Devlin, S., Duggan, J., & Howley, E. (2016). Multi-objective dynamic dispatch optimisation using multi-agent reinforcement learning. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems* (pp. 1345–1346). International Foundation for Autonomous Agents and Multiagent Systems.

Okdinawati, L., Simatupang, T. M., & Sunitiyoso, Y. (2017). Multi-agent reinforcement learning for value co-creation of Collaborative Transportation Management (CTM). *International Journal of Information Systems and Supply Chain Management, 10*(3). https://doi.org/10.4018/IJISSCM.2017070105.

Pontrandolfo, P., Gosavi, A., Okogbaa, O. G., & Das, T. K. (2002). Global supply chain management: A reinforcement learning approach. *International Journal of Production Research, 40*(813), 1266–1317. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.1096.

Rana, R., & Oliveira, F. S. (2015). Dynamic pricing policies for interdependent perishable products or services using reinforcement learning. *Expert Systems with Applications, 42*(1), 426–436. https://doi.org/10.1016/j.eswa.2014.07.007.

Rummery, G., & Niranjan, M. (1994). On-line Q-learning using connectionist systems. 37. *University of Cambridge, Department of Engineering*. http://mi.eng.cam.ac.uk/reports/svr-ftp/auto-pdf/rummery_tr166.pdf.

Šemrov, D., Marsetič, R., Žura, M., Todorovski, L., & Srdic, A. (2016). Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research Part B: Methodological, 86*, 250–267. https://doi.org/10.1016/j.trb.2016.01.004.

Shin, M., Ryu, K., & Jung, M. (2012). Reinforcement learning approach to goal-regulation in a self-evolutionary manufacturing system. *Expert Systems with Applications, 39*(10), 8736–8743. https://doi.org/10.1016/j.eswa.2012.01.207.

Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning, 22*(1–3), 123–158. https://doi.org/10.1007/BF00114726.

Smart, W. D., & Kaelbling, L. P. (2000). Practical reinforcement learning in continuous spaces. In *Proceedings of the seventeenth international conference on machine learning* (pp. 903–910).

Sun, R., & Zhao, G. (2012). Analyses about efficiency of reinforcement learning to supply chain ordering management. In *IEEE international conference on industrial informatics (INDIN)* (pp. 124–127). https://doi.org/10.1109/INDIN.2012.6301163.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning – An introduction*. Cambridge, Massachusetts, London, England: MIT Press.

Tekin, E., Gürler, Ü., & Berk, E. (2001). Age-based vs. stock level control policies for a perishable inventory system. *European Journal of Operational Research, 134*(2), 309–329. https://doi.org/10.1016/S0377-2217(00)00250-2.

Watkins, C. (1989). Learning from delayed rewards. *Cambridge University*. https://doi.org/10.1016/0921-8890(95)00026-C.

Zhang, W., & Dietterich, T. (1995). A reinforcement learning approach to job-shop scheduling. In IJCAI, 95, pp. (1114–1120). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.3850&rep=rep1&type=pdf.

Zhou, Y., Hao, J. K., & Duval, B. (2016). Reinforcement learning based local search for grouping problems: A case study on graph coloring. *Expert Systems with Applications, 64*, 412–422. https://doi.orwg/10.1016w/j.eswwa.2016.07.047.