Reinforcement Learning for Inventory Management

Shraddha Bharti, Dony S. Kurian and V. Madhusudanan Pillai

Abstract The decision of "how much to order" at each stage of the supply chain is a major task to minimize inventory costs. Managers tend to follow particular ordering policy seeking individual benefit which hampers the overall performance of the supply chain. Major findings from the literature show that, with the advent of machine learning and artificial intelligence, the trend in this area has been heading from simple base stock policy to intelligence-based learning algorithms to gain near-optimal solution. This paper initially focuses on formulating a multi-agent four-stage serial supply chain as reinforcement learning (RL) model for ordering management problem. In the final step, RL model for a single-agent supply chain is optimized using *Q*-learning algorithm. The results from the simulations show that the RL model with *Q*-learning algorithm is found to be better than Order-Up-To policy and 1–1 policy.

Keywords Supply chain · Ordering policy · Inventory management · Reinforcement learning · *Q*-learning

1 Introduction

A supply chain is an integrated network of multiple agents consisting of retailers, distributors, manufacturers, and suppliers. Each agent has to make replenishment decisions on "how much to order" with the aim of minimizing long-term total supply chain inventory cost. Furthermore, the decisions made by the human agents while operating the supply chain are often biased due to the behavior of agents [1, 2].

In a decentralized supply chain, each agent has to make an independent decision based on its interacting environment [3]. The order decision hence depends on factors such as the size of the downstream demand, quantity received from the

S. Bharti · D. S. Kurian · V. M. Pillai (✉)
Department of Mechanical Engineering, National Institute of Technology Calicut, Kozhikode 673601, India
e-mail: vmp@nitc.ac.in

BBVL. Deepak et al. (eds.), *Innovative Product Design and Intelligent Manufacturing Systems*, Lecture Notes in Mechanical Engineering,
https://doi.org/10.1007/978-981-15-2696-1_85

upstream, and inventory level. In addition, the complexity of the decision process is increased if the agents are subjected to uncertain system parameters (e.g., customer demand and lead time) and bullwhip effect [4, 5]. In such complex situations, agents must take decision based on the system's state rather than following a fixed decision rule.

Various approaches have been proposed by past researchers regarding the optimality of inventory order decisions. The base stock policy is found to be optimal in a multi-agent inventory system when several assumptions are incorporated in the model [6]. When the agent faces deterministic demand with penalty cost for unfulfilled orders, the best ordering policy is "pass order" or "one for one" (1–1) policy [7]. In 1–1 policy, each agent gives order to the upstream which is equal to the order received from the downstream.

The classical example of a decentralized supply chain is the MIT's beer distribution game. Sterman [2] points out that in a beer game environment where agents act irrationally there is no known optimal policy for an agent wishing to act optimally. In his work, a formula-based method to model the agent's decision-making behavior is proposed. Like Sterman [2], Mosekilde and Larsen [8] and Strozzi et al. [9] also adopted a formula-based approach which attempts to model agent-based decision-making. But these approaches fail to determine optimal decisions. The minimum supply chain cost under beer game settings is obtained when the different agents adopt different ordering policies rather than a single ordering policy [9, 10].

Interest in harnessing the power of learning algorithms in the supply chain has increased due to the emergence of artificial intelligence. Genetic algorithm (GA), an evolutionary learning algorithm, is used by Kimbrough et al. [7] to design a multi-agent system under beer game settings. Results from their study show that the artificial agents learned via GA are able to play a beer game effectively compared to human agents.

There are mainly three categories of machine learning techniques, namely supervised learning, unsupervised learning, and reinforcement learning (RL). Labeled set of accurate training data is needed for supervised learning. Unsupervised learning is used to find a hidden pattern from the collection of unlabeled data. Unlike other forms of machine learning, in RL, there is no exact action to perform; instead, a learning agent learns by trial and error based on its state and acts based on the current state.

Many practical problems are found to be stochastic in nature, and such discrete-time stochastic processes are formulated as Markov decision process (MDP). Reinforcement learning (RL) is a technique to solve MDP [11]. In recent years, RL has been implemented to solve several problems in supply chain management. The RL framework is used to address the coordination problem of global supply chains [12]. Preliminary work on RL for ordering management has been proposed by Giannoccaro and Pontrandolfo [13]. They have employed a semi-markov average reward technique to solve inventory decision problem in a three-stage supply chain. In another study, Chaharsooghi et al. [5] have applied Q-learning method, widely used RL algorithm, to optimize inventory order decisions of four-stage supply chain. Kara and Dogan [14] have addressed ordering

policies for perishable products using Q-learning and SARSA algorithm based on RL. Performance of both the algorithms proves to be better than the genetic algorithm. Recently, Oroojlooyjadid et al. [15] proposed a RL algorithm based on Deep Q-Networks (DQN) and a transfer learning approach to find a near-optimal ordering policy in the beer game environment. Collectively, these studies provide evidence that by addressing the supply chain inventory management problem as RL problem, it is possible to achieve near-optimal ordering policy.

This paper aims to put forth a modified learning mechanism for an artificial agent to make inventory decisions on ordering size. In the previous works [5, 13, 15], the agent has adopted $X + Y$ rule as an ordering policy where X represents the downstream demand while Y indicates the quantity determined by the learning agent. This paper primarily centers on formulating a multi-agent four-stage supply chain as RL model for deriving near-optimal solutions. Subsequently, a Q-learning algorithm with modified ordering rule is adopted to solve a single-agent supply chain to obtain minimum inventory cost.

The remainder of this paper is organized as follows. Section 2 provides a brief description of RL and MDP. Section 3 discusses the problem formulation using RL approach. Q-learning algorithm is employed to solve the single-agent supply chain problem in Sect. 4 and the results are discussed in Sect. 5.

2 Reinforcement Learning

The fundamental concept behind RL is the interaction between the learning agent and its environment. During the learning process, an agent at time-step t selects an action a_t based on the environmental state s_t. As an outcome of its action, the agent receives a reward r_{t+1} and transits from state s_t to new state s_{t+1} (see Fig. 1). In the long term, the objective of any RL agent is to maximize the cumulative reward by learning what to do and how to map situations [16].

There should be Markovian property associated with every state of the RL model, and even if the states are non-Markov yet it is appropriate to approximate it as a Markov state [14]. Markov decision process (MDP) is a RL satisfying the Markov property. MDP, in brief, is a sequential decision-making model which

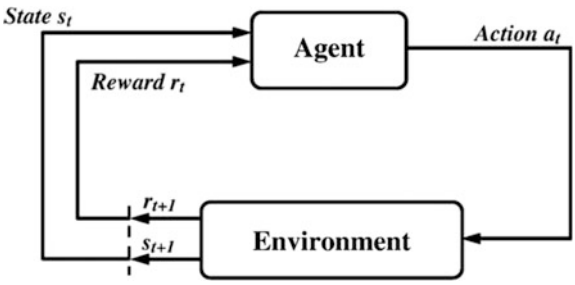


Fig. 1 Agent and environment interaction in RL [16]

consists of decision periods, system states, available actions, rewards or costs, and transition probabilities. The information provided by the system state assists the decision-maker to choose an action at each decision epoch and gets the corresponding reward. Combined previous states and actions or present state determines the action to be chosen in the present state, and this is termed as a decision rule. A decision rule forms policy, and a reward is acquired by implementing the policy. The aim is to maximize the reward sequence by choosing a suitable policy [17].

3 Problem Description and Formulation

A serial four-stage supply chain as shown in Fig. 2 is considered for the study. It consists of one retailer ($i = 1$), one wholesaler ($i = 2$), one distributor ($i = 3$), and one factory ($i = 4$). The retailer agent faces the stochastic demand ($D(t)$) from its customer. It is assumed that the factory has an unlimited production capacity and whatever quantity demanded by the factory is released ($R(t)$) after production. Each agent i of the supply chain tries to place independent decisions on ordering size ($O_i(t)$) to its upstream agent, and shipping orders ($S_i(t)$) placed by its downstream agent over a series of time period $t = 1, 2, 3 \ldots, T$. The parameters of the RL including reward function, state variable, and agent's policy are described in the next subsection.

3.1 Reward Function

At time-step t, the agent observes the state and takes an action. Agent receives a reward as a feedback corresponding to the action taken. The reward is the payoff for taking the right decision. The aim of the inventory ordering management problem is to minimize the cost; therefore, the reward function is defined as a loss function and it is estimated as in Eq. (1).

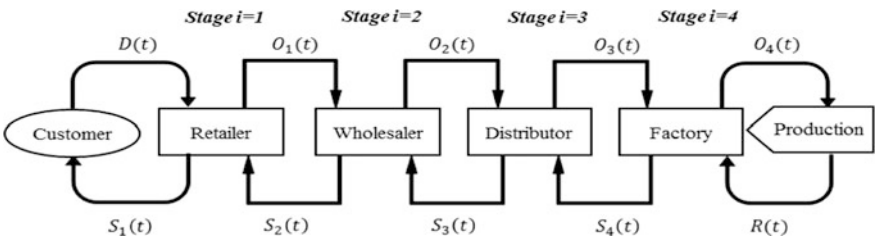


Fig. 2 Order and shipment flow in a supply chain

$$r(t+1) = \sum_{i=1}^{4} \left[C_i^h \times I_i^+(t) + C_i^s \times I_i^-(t) \right] \quad (1)$$

The reward function $r(t + 1)$ is a function of the holding cost and lost sales cost occurring at period t. In the Q-learning algorithm, at every epoch, the value of Q has to be updated. For this purpose, reward value (R) is required at each step as shown in Fig. 3. Reward value (R) depends upon the reward function ($r(t + 1)$). If the value of the current period reward function is less than or equal to the value of the previous period reward function, then R equals to +1 else R takes −1. In the above equation, C_i^h is the unit holding cost and C_i^s is the unit lost sales cost. $I_i^+(t)$ represents inventory at the end of the period t, and $I_i^-(t)$ is the lost sales quantity.

3.2 State Variable

State variable defines the state of the system and provides appropriate information to the decision-maker. In this study, inventory position ($IP_i(t)$) at time-step t is considered as a state variable of the agent i. Inventory position at time-step t is the sum of end-period inventory at t and on-order inventory at t. State variable vector $s(t)$ at time-step t is represented as follows:

$$s(t) = [IP_1(t), IP_2(t), IP_3(t), IP_4(t)] \quad (2)$$

1-*Set the initial conditions*
Iteration = 0, t = 0, Q(s,a) = 0
2-*While iteration ≤ Maximum_iteration*
Set the inventory position of retailer:
IP = 2
While t < weeks
(a) Select an action vector according to ε-greedy policy with probability of exploration
(b) Observe next state (s′)
(c) Calculate immediate reward (r(t+1)) and corresponding R
(d) Update Q(s,a) using:
$Q(s,a) \leftarrow Q(s,a) + \alpha [R + max_a Q(s',a') - Q(s,a)]$
(e) $s \leftarrow s'$
(f) $t \leftarrow t'$
t = 0
Iteration = Iteration + 1

Fig. 3 Q-learning algorithm

3.3 Ordering Policy

A modified $X + Y$ ordering policy is proposed in this study in contrast to ordering rule discussed in the literature [5, 15]. According to this modified policy, X indicates the quantity determined via Order-Up-To (OUT) decision rule and Y denotes the agent's policy which takes the values positive, negative, or zero implying that the agent can order more, less, or equal to X. OUT inventory policy is a periodic review type inventory system where OUT level is the maximum (target) inventory level. In OUT policy, order quantity (X) is determined by taking the difference between OUT level and inventory position only if the OUT level is greater than inventory position; otherwise, X is equal to zero [18]. At every time-step t, agent orders $X + Y$ quantity to his upstream member.

The modified $X + Y$ ordering rule helps to limit the state space under consideration; that is, inventory position variation can be limited to a finite number which in turn helps the agent to learn faster.

3.4 Agent's Policy

The agent's policy $Y_s(t)$ for the state s is given as:

$$Y_s(t) = [Y_{1s}(t), Y_{2s}(t), Y_{3s}(t), Y_{4s}(t)] \tag{3}$$

where $Y_{is}(t)$ represents the value of Y determined by the agent i for the state s at time-step t.

4 Q-Learning Algorithm for Single-Agent Supply Chain

Most widely used algorithm for solving RL model is Q-learning algorithm. This is because of the model-free nature of the algorithm and it does not require complete knowledge of the system [16].

Q-learning is a temporal difference method and it learns from experience [16]. After a certain number of episodes (iterations), the algorithm finds the best state-action pair values ($Q(s, a)$) which are called Q-values. Q-values are stored in Q-table and get updated through iterations. The rows of the table represent states (s) and columns represent actions (a). The table elements are initialized as zero and then get updated as algorithm proceeds. At the end of the learning, a table with learned Q-values is obtained. In addition, the best course of action is selected for each state based on the Q value. In this paper, Q-learning method is proposed to solve the RL ordering model for a single-agent supply chain and the algorithm is described in Fig. 3.

The retailer faces stochastic demand from the customer which is randomly generated from a uniform distribution between [1, 4]. The generated sequence of demand distribution is [1–4]. Lead time considered is zero and the period of operation is from 1 to 25 weeks. The inventory position is the sum of beginning inventory and the quantity ordered in the previous period but not yet received (on-order quantity). Beginning inventory at the start of every iteration for the agent is initialized by 2 based on mean demand and review period. The algorithm is run for 10^5 iterations with a ε-greedy policy where ε is the probability of exploration which indicates that the agent performs random action with probability ε and performs greedy action (exploitation) with $(1 - \varepsilon)$ probability. In every iteration, the supply chain operates for 25 weeks. In this model, the probability of exploration is reduced linearly with increase in iteration number. Probability of exploration is taken as 98% in the first episode and 10% in the last episode. In each specific episode also, it is getting reduced from 1st week to 25th week linearly to 2% as reported in the literature [5]. Learning rate (α) can take values from 0 to 1 where the value of one indicates that the agent tries to learn everything, while the value of zero means the agent learn nothing. Learning rate (α) of 0.3 is found to be appropriate on trying out different values ranging from 0.1 to 0.7.

Results from simulation indicate that the proposed ordering policy facilitates in limiting the state space. Thus, for the given demand distribution with OUT level equal to three and for the agent's action $Y = [-1, 0, 1]$, the range of inventory position lies between [0, 3]; that is, the states [0, 1, 2, 3] are suitable for this case. Unit holding cost and unit lost sales cost are considered to be 5 and 10, respectively. After learning, Q-table gives the optimal policy when greedy action is taken for each state.

5 Experimental Result and Validation

The entire programming for solving the single-agent supply chain ordering decision problem is carried out in Python 3.5 language. The aim of this study is to minimize the total inventory cost (TC) for 25 weeks which is the sum of holding cost and lost sales cost. The convergence of the total cost is obtained by solving the RL model using Q-learning algorithm. The convergence obtained after the training can be shown by plotting a graph between the number of iterations and its corresponding cost obtained and is depicted in Fig. 4. For the same supply chain, order management is simulated under 1–1 policy and OUT policy using Microsoft Excel. The performance of the three order management policies is compared. The comparison has been carried out using the total inventory cost for different policies and is shown in Table 1.

The average cost of 25 weeks for last 100 episodes obtained from RL model is better than the cost obtained from the other two policies. The result obtained from the RL model outperforms OUT policy by 5% and 1–1 policy by 20.83%. For instance, a graph is plotted indicating cumulative cost obtained from three policies for 25 weeks and is shown in Fig. 5.

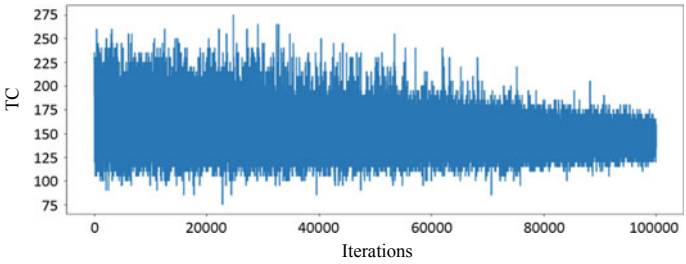
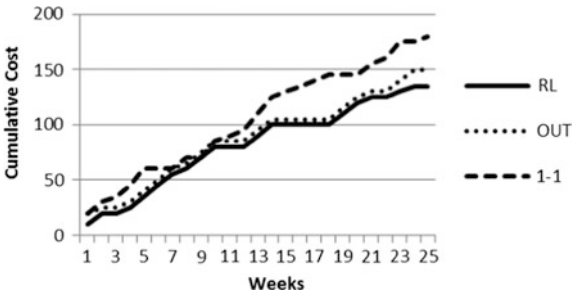


Fig. 4 Convergence plot (number of iterations versus total cost)

Table 1 Total inventory cost comparison

Policy	1–1 policy	OUT policy	RL model
Total inventory cost (TC)	180	150	142.5

Fig. 5 Cumulative cost versus weeks



6 Conclusion

This paper is focused on supply chain ordering decision problem. Initially, a four-stage serial supply chain is formulated as a RL model for inventory management. As a next step, the RL model for a single-agent supply chain where the agent faces stochastic demand from the customer is solved using Q-learning. This study particularly has made use of a modified ordering rule which is much more effective than the ordering policy employed in the literature. Results from the study show that the RL model is found to be efficient in deciding order size. The total inventory cost obtained using the RL model is lesser than the supply chains simulated for 1–1 policy and OUT policy. The achieved results through RL model are promising, and there is a good scope of solving inventory decision problems using RL approaches in a multi-agent supply chain.

References

1. Lee HL, Padmanabhan V, Whang S (1997) Information distortion in a supply chain: the bullwhip effect. Manag Sci 43(4):546–558
2. Sterman JD (1989) Modeling managerial behavior: misperceptions of feedback in a dynamic decision making experiment. Manag Sci 35(3):321–339
3. Claus C, Boutilier C (1998) The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the fifteenth national conference on artificial intelligence. AAAI, Madison, Wisconsin, pp 746–752
4. Forester JW (1961) Industrial dynamics, 1st edn. MIT Press; Wiley, New York
5. Chaharsooghi SK, Heydari J, Zegordi SH (2008) A reinforcement learning model for supply chain ordering management: an application to the beer game. Decis Support Syst 45(4):949–959
6. Clark AJ, Scarf H (1960) Optimal policies for a multi-echelon inventory problem. Manag Sci 6(4):475–490
7. Kimbrough SO, Wu DJ, Zhong F (2002) Computers play the beer game: can artificial agents manage supply chains? Decis Support Syst 33(3):323–333
8. Mosekilde E, Larsen ER (1986) Deterministic chaos in the beer production-distribution model. Syst Dyn Rev 4(1–2):131–147
9. Strozzi F, Bosch J, Zaldivar JM (2007) Beer game order policy optimization under changing customer demand. Decis Support Syst 42(4):2153–2163
10. Edali M, Yasarcan H (2016) Results of a beer game experiment: should a manager always behave according to the book? Complexity 21(S1):190–199
11. Gosavi A (2009) Reinforcement learning: a tutorial survey and recent advances. INFORMS J Comput 21(2):178–192
12. Pontrandolfo P, Gosavi A, Okogbaa OG, Das TK (2002) Global supply chain management: a reinforcement learning approach. Int J Prod Res 40(6):1299–1317
13. Giannoccaro I, Pontrandolfo P (2002) Inventory management in supply chains: a reinforcement learning approach. Int J Prod Econ 78(2):153–161
14. Kara A, Dogan I (2017) Reinforcement learning approaches for specifying ordering policies of perishable inventory systems. Expert Syst Appl 91:150
15. Oroojlooyjadid A, Nazari M, Snyder L, Takáč M (2017) A deep *Q*-network for the beer game: a reinforcement learning algorithm to solve inventory optimization problems. arXiv preprint arXiv:1708.05924 [cs. LG]
16. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction, 1st edn. MIT Press, Cambridge
17. Puterman ML (1994) Markov decision processes: Discrete stochastic dynamic programming. Wiley, New York
18. Daniel JSR, Rajendran C (2005) A simulation-based genetic algorithm for inventory optimization in a serial supply chain. Int Trans Oper Res 12(1):101–127