Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

A DEEP Q-NETWORK BASED ON RADIAL BASIS FUNCTIONS FOR MULTI-ECHELON INVENTORY MANAGEMENT

Liqiang Cheng
Jun Luo

Shanghai Jiao Tong University
NO.1954 Huashan Rd
Shanghai, 200030, CHINA

Weiwei Fan

Tongji University
No.1239 Siping Rd
Shanghai, 200092, CHINA

Yidong Zhang
Yuan Li

Dchain Department, Alibaba Group
No.969 West Wen Yi Rd
Hangzhou, 311121, CHINA

ABSTRACT

This paper addresses a multi-echelon inventory management problem with a complex network topology where deriving optimal ordering decisions is difficult. Deep reinforcement learning (DRL) has recently shown potential in solving such problems, while designing the neural networks in DRL remains a challenge. In order to address this, a DRL model is developed whose Q-network is based on radial basis functions. The approach can be more easily constructed compared to classic DRL models based on neural networks, thus alleviating the computational burden of hyperparameter tuning. Through a series of simulation experiments, the superior performance of this approach is demonstrated compared to the simple base-stock policy, producing a better policy in the multi-echelon system and competitive performance in the serial system where the base-stock policy is optimal. In addition, the approach outperforms current DRL approaches.

1 INTRODUCTION

Supply chain management plays a crucial role in business operations, with inventory management being a core process within it. The focus of this paper is on the multi-echelon inventory ystem which consists of multiple stages or echelons that hold inventory (Gijsbrechts et al. 2022), because of its rising popularity in real-world supply chains. For instance, in Alibaba's supply chain, suppliers deliver inventory to a central warehouse, which then allocates inventory to downstream retailers in its region. In managing such inventory system, the manager may desire to dynamically determine the ordering decision at each period so that the total supply chain costs is minimized. The dynamic inventory management problem can be essentially formulated as a Markov Decision Process (MDP). To address this problem, many methods have been developed, dating back to Clark and Scarf (1960) and Sherbrooke (1968). However, these methods are specifically designed for multi-echelon systems with simple structure, e.g., the serial system. As pointed by Zipkin (2000), the optimal inventory policy for the general multi-echelon system is yet unknown, due to the complexity of systems. As a remedy, various approximation methods are proposed, while they often require certain assumptions, such as Poisson process demand or zero lead time. Qiu et al. (2022) investigated an integrated optimization problem of inventory management and transportation vehicle selection. They formulate the problem as a mixed-integer quadratically constrained program and established a convex approximation of the proposed formulation using Cauthy inequalities. More interesting review of these methods can be found in Simchi-Levi and Zhao (2012).

Recently, reinforcement learning (RL), also known as approximate dynamic programming (ADP), enjoys notable success in solving MDP. In a typical RL, an agent consistently interacts with the environment, where at each period, the agent observes the system's state, takes an action and receives the corresponding reward. Specifically, the action at each period is generated by optimizing the expected value of the total reward starting from the current state, termed by the Q-function. Classic RL approaches estimate the values of Q-function for all possible action-state pairs and store them in a lookup table named Q-table. Obviously, these methods are not suitable for our inventory management problem, because both our state (i.e., inventory level) and action (i.e., ordering decision) spaces can be large or even continuous. Instead, other approaches construct function approximations for the Q-function (Powell 2011). When the neural networks are used as the function approximator, the corresponding RL is called deep reinforcement learning (DRL) and the constructed approximation is called Q-network. DRL have gained significant attention for achieving human-like intelligence and even surpassing humans in some games, such as Go (Mcgrath et al. 2022) and Atari games (Mnih et al. 2013).

The appeal of the DRL approach arises from the strong approximation ability of neural networks. In light of this, our paper seeks to adopt the DRL approaches in solving our complicated inventory management problem.

DRL have been applied to solve the inventory management MDPs in different systems. Oroojlooyjadid et al. (2021) proposed a deep Q-network to play the beer game, which is a special serial system. The deep Q-network can achieve near-optimal solutions when playing with teammates who follow a base-stock policy. Wang et al. (2022) developed a double deep Q-network to the lost sales problem, which is a flexible solution that can be applied with different cost parameter settings. Van Roy et al. (1997) derived a neural network dynamic programming approach to solve a two-echelon system, where they manually developed 23 product features to construct the neural network. Gijsbrechts et al. (2022) exploited asynchronous advantage actor-critic algorithm (A3C) for solving lost sales, dual sourcing, and multi-echelon problems. The proposed A3C algorithm can match performance of state-of-the-art heuristics. Liu et al. (2022) applied a multi-agent DRL approach to multi-echelon inventory management, demonstrating that this approach can achieve lower costs and less significant bullwhip effects compared to single-agent DRL methods. They designed a recurrent neural network (RNN) to utilize historical information. Despite the success of DRL in inventory management, designing neural networks in DRL is complex, and tuning hyperparameters remains computationally burdensome (Gijsbrechts et al. 2022).

To alleviate the hyperparameter-tuning burden, we deploy a deep Q-network approach based on a radial basis functions (RBF) (Broomhead and Lowe 1988). The proposed RBF based deep Q-network is a special three-layered neural network, with its hidden layer neurons representing the states of the MDP and the activation function is kernel function. While the hidden layer neurons have a real meaning, the RBF based deep Q-network is easy to design and implement. Our simulation study demonstrates that the proposed RBF based deep Q-network approach achieves appealing performance compared to the base-stock policy and current DRL approaches. For the serial system with one warehouse and one retailer, the RBF based deep Q-network obtains a near-optimal solution (the optimal policy is the base-stock policy). For the multi-echelon system with one warehouse and multiple retailers, the RBF based deep Q-network outperforms the base-stock policy and current DRL approaches.

The structure of this paper is as follows. Section 2 provides the system dynamics and MDP formulation of the multi-echelon inventory management problem considered in this paper. Section 3 introduces our RBF based deep Q-network approach to solve the inventory problem, and Section 4 presents the numerical results obtained from simulated scenarios. Finally, Section 5 concludes this paper.

2 MULTI-ECHELON INVENTORY MANAGEMENT

This section introduces the multi-echelon inventory management model and its corresponding MDP. Section 2.1 presents the dynamics of multi-echelon inventory management, including events that occur and their sequence. Based on the these events, a discrete event simulation model is established. Section 2.2 formulates the MDP for multi-echelon inventory management, and briefly introduces a Q-learning method for solving this MDP, based on which the approach is designed.

2.1 System Dynamics and Simulation Model

This section describes the multi-echelon system, which is a one-warehouse multiple-retailer system with K identical retailers. At each period of the infinite periods, random demands materialize at each retailer, and are fulfilled by inventory held at the retailers. Demands are independently and identically distributed through time and among different retailers. The retailers are replenished by a warehouse, while the warehouse is replenished by a supplier. There are delays in the transportation of orders both from the supplier to the warehouse and from the warehouse to each retailer. The delays are considered to be several periods. Hence, the system evolves in discrete time.

The inventory management is considered over an infinite number of periods. We use time points $t, t+1, \ldots$ to represent the beginning of each period, which also marks the end of the previous period. Without loss of generality, we focus on the multi-echelon inventory management process of one period between time point t and time point $t+1$. At time point t, the on-hand inventory of the warehouse and retailers is denoted by I_t^w and I_t^i, respectively (where $i = 1, \ldots, K$), and the pipeline inventory of the warehouse and retailers is denoted by $Q_t^w = (q_{t-1}^w, \ldots, q_{t-l_w}^w)$ and $Q_t^i = (q_{t-1}^i, \ldots, q_{t-l_r}^i)$ (where $i = 1, \ldots, K$). Figure 1 illustrates the on-hand inventory and pipeline inventory at time point t, the beginning of one period.

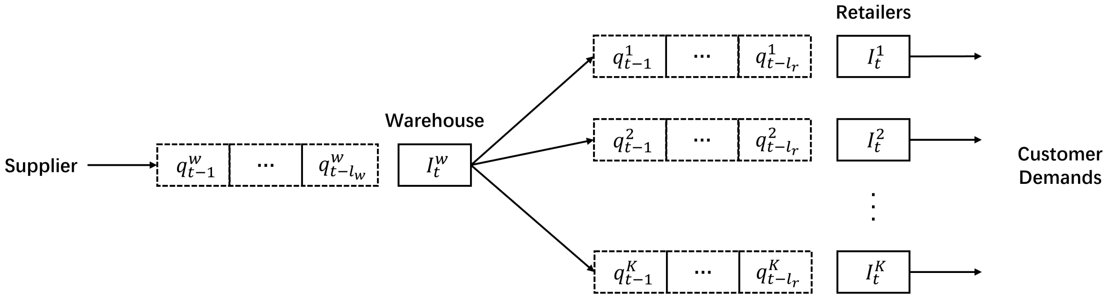


Figure 1: Inventory of the multi-echelon system at time point t.

The multi-echelon inventory management process between time point t and $t+1$ consists of five events: order arrival, demand fulfillment, special delivery, replenishment, and delivery and update. To provide a more precise description, we introduce virtual time points t_1, t_2, t_3 and t_4, at which the first four events occur. The delivery and update event occurs at time point $t+1$, which marks the end of the period. Each of the five events is described in detail below.

Orders Arrival: At time point t_1, the warehouse and K retailers receive delayed orders that were placed l_w and l_r periods ago, denoted as $q_{t-l_w}^w$ and $q_{t-l_r}^i, i = 1, \ldots, K$, respectively. These orders are then added to the on-hand inventory of the warehouse and retailers, denoted as I_t^w and I_t^i:

$$I_{t_1}^w = I_t^w + q_{t-l_w}^w,$$
$$I_{t_1}^i = I_t^i + q_{t-l_r}^i, 1, \ldots, K.$$

Demand Fulfillment: At time point t_2, each retailer i (where $i = 1, \ldots, K$) samples its demand d_t^i from a normal distribution with mean μ and standard deviation σ. The retailer then fulfills its demand using its on-hand inventory $I^i_{t_1}$. The on-hand inventory of the warehouse remains unchanged from its value at time point t_1 as $I^w_{t_2} = I^w_{t_1}$. While the on-hand inventory of the retailers is updated according to the inventory used to fulfill the demand:

$$I^i_{t_2} = (I^i_{t_1} - d^i_t)^+, 1, \ldots, K.$$

Special Delivery: At time point t_3, the special delivery event occurs only when the on-hand inventory of a retailer i is insufficient to meet the demand, i.e., $d^i_t > I^i_{t_1}$. In such a case, each unfulfilled demand either waits for a special delivery from the warehouse with probability P_w (if the warehouse on-hand inventory $I^w_{t_2}$ is not 0) or is lost with probability $1 - P_w$, resulting in lost sales. The total number of special deliveries at retailer i, denoted as B^i_t, follows a binomial distribution $B\left((d^i_t - I^i_{t_1})^+, P_w\right)$. The total number of special deliveries for all retailers is $B_t = \sum_{i=1}^K B^i_t$. During the special delivery event, the on-hand inventory of the retailers remains unchanged, i.e., $I^i_{t_3} = I^i_{t_2}$ for $i = 1, \ldots, K$. while the on-hand inventory of the warehouse is updated as follows:

$$I^w_{t_3} = I^w_{t_2} - B_t.$$

Any demands that are not fulfilled by the retailers' or warehouse's on-hand inventory leads to a shortage cost with a cost rate of p. Additionally, special deliveries from the warehouse incur an ordering cost with a cost rate of c_w. Therefore, the shortage cost and ordering cost at the period can be calculated as follows:

$$\text{shortage cost: } p[\sum_{i=1}^K (d^i_t - I^i_{t_1})^+ - B_t], \tag{1}$$

$$\text{ordering cost: } c_w B_t. \tag{2}$$

Replenishment: At time point t_4, the warehouse places an order q^w_t, and each retailer $i, i = 1, \ldots, K$ places an order q^i_t. The total order quantities of the retailers cannot exceed the on-hand inventory of the warehouse, i.e., $\sum_{i=1}^K q^i_t \le I^w_{t_3}$. Note that both the warehouse and retailers have limited capacities: (1) the maximum order quantity of the warehouse is C^m, (2) the warehouse inventory position $Z^w_{t_4} = I^w_{t_3} + \sum_{j=1}^{l_w} q^w_{t-j+1}$ cannot exceed C^w, and (3) each retailer inventory position $Z^i_{t_4} = I^i_{t_3} + \sum_{j=1}^{l_r} q^i_{t-j+1}, i = 1, \ldots, K$, cannot exceed C^r. These limited capacities result in a restriction on the order quantities of the warehouse and retailers. Additionally, at time point t_4, the orders of the warehouse and retailers are not delivered, so the on-hand inventory and pipeline inventory of the warehouse and retailers do not change, i.e., $I^w_{t_4} = I^w_{t_3}$, $I^i_{t_4} = I^i_{t_3}, i = 1, \ldots, K$.

Delivery and Update: At the end of the period, denoted by the virtual time point $t+1$, the orders placed during this period enter the pipeline inventory, and the pipeline inventory advances by one period. The on-hand inventory and pipeline inventory of the warehouse and retailers are then updated as follows:

$$\begin{aligned} I^w_{t+1} &= I^w_{t_4} - \sum_{i=1}^K q^i_t, \\ I^i_{t+1} &= I^i_{t_4}, i = 1, \ldots, K, \\ Q^w_{t+1} &= (q^w_{t-l_w+1}, \ldots, q^w_t), \\ Q^i_{t+1} &= (q^i_{t-l_r+1}, \ldots, q^i_t), i = 1, \ldots, K. \end{aligned}$$

Inventory held at the warehouse or retailers incurs a holding cost with a cost rate of h_w and h_r, respectively. The total holding cost at period t can be calculated as follows:

$$\text{holding cost: } h_w I^w_{t+1} + h_r \sum_{i=1}^K I^i_{t+1}. \tag{3}$$

To implement the simulation model, the sequence of these events is specified, and they occur in the following order at each period: $t < t_1 < t_2 < t_3 < t_4 < t+1$. The procedure of multi-echelon inventory management simulation for one period are illustrated in Figure 2.

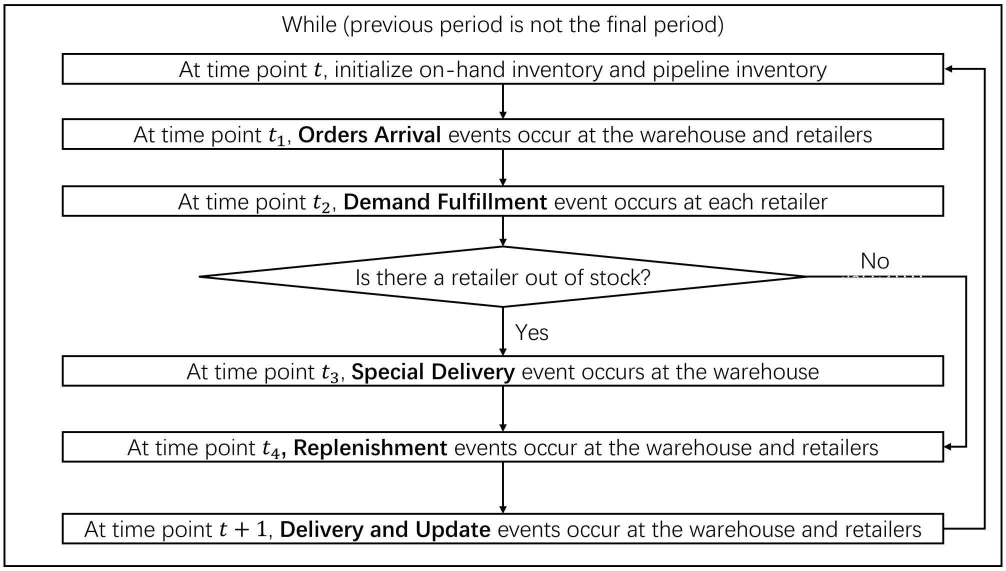

While (previous period is not the final period)
At time point t, initialize on-hand inventory and pipeline inventory
At time point t_1, **Orders Arrival** events occur at the warehouse and retailers
At time point t_2, **Demand Fulfillment** event occurs at each retailer
Is there a retailer out of stock?
No
Yes
At time point t_3, **Special Delivery** event occurs at the warehouse
At time point t_4, **Replenishment** events occur at the warehouse and retailers
At time point $t+1$, **Delivery and Update** events occur at the warehouse and retailers

Figure 2: The procedure of multi-echelon inventory management simulation for one period between time point t and $t+1$.

2.2 Markov Decision Process

The multi-echelon system's dynamics can be formulated as a MDP. In this MDP, state $s_t = (I_t^w, Q_t^w, I_t^r, Q_t^r)$, where (I_t^w, Q_t^w) are warehouse on-hand and pipeline inventory at period t, and (I_t^r, Q_t^r) are retailers on-hand and pipeline inventory at period t. Action $a_t = (q_t^w, q_t^r)$, where (q_t^w, q_t^r) are warehouse and retailers order quantities at time t. The reward in this MDP is represented by the cost, denoted as $c_t(s_t, a_t)$. The cost can be broken down into three parts, shortage cost, ordering cost, and holding cost, which are defined in (1), (2) and (3), respectively. The objective of this MDP is to minimize the expected cumulative costs by controlling actions from period t to infinity:

$$\min_{a_{t+j}, j=0,1,\dots} \sum_{j=0}^{\infty} \gamma^j E[c_{t+j}(s_{t+j}, a_{t+j})], \tag{4}$$

where γ is a discount rate. With a transition probability $P(s_{t+1} \mid s_t, a_t)$, which describes the probability of the system to transit from state s_t to state s_{t+1} when picking action a_t, the Objective (4) can be achieved using linear programming or dynamic programming (Sutton and Barto 2018).

Q-learning is an another approach for solving the problem. In Q-learning, the action-value function $Q(s_t, a_t)$ represents the future expected cost of taking action a_t at state s_t and then following the optimal control from state s_{t+1}. For any state $s_t \in S$, the action-value function can be written as:

$$Q(s_t, a_t) = E[c_t(s_t, a_t)] + \min_{a_{t+j}, j=1,2,\dots} \sum_{j=1}^{\infty} \gamma^j E[c_{t+j}(s_{t+j}, a_{t+j})].$$

The Objective (4) can be achieved equivalently by minimizing $Q(s_t, a_t)$. Thus the optimal action can be calculated by:

$$a_t^\star = \arg\min_{a_t} Q(s_t, a_t).$$

The Q-learning approach begins by assigning an initial Q-value, typically set to 0, to all states and actions. It then iteratively update the Q-values using the following formula:

$$Q_{t+1}(s_t, a_t) = (1-\alpha) Q_t(s_t, a_t) + \alpha E[c_t(s_t, a_t) + \gamma \min_a Q_t(s_{t+1}, a)],$$

where α is the learning rate. The agent chooses actions using the ε-greedy method, which implies that the agent chooses an action randomly with a probability ε, and selects the action with the smallest Q-value with a probability of $1-\varepsilon$.

In a typical Q-learning process, the values of $Q(s_t,a_t)$ are stored in a lookup table, called Q-table. Solving MDP with a large state-action space by updating the Q-table values is impossible, which is known as the curse of dimensionality. To address this, Mnih et al. (2015) developed a deep Q-network (DQN) algorithm that uses a neural network as an approximation function $\hat{Q}(s_t,a_t;W_t)$ of action-value function $Q(s_t,a_t)$. Figure 3 shows the structure of a typical deep Q-network, which has $|A|$ outputs in the output layer representing the approximated values of $Q(s_t,a_t)$ for every possible action $a_t \in A$. Based on this structure, we propose a specialized and easy-to-design deep Q-Network for solving the multi-echelon inventory management MDP.

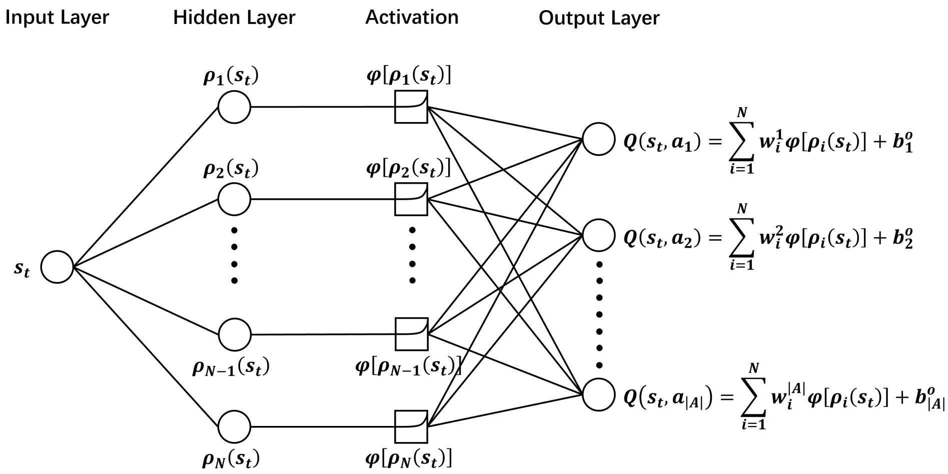

Input Layer
Hidden Layer
Activation
Output Layer
s_t
$\rho_1(s_t)$
$\rho_2(s_t)$
$\rho_{N-1}(s_t)$
$\rho_N(s_t)$
$\varphi[\rho_1(s_t)]$
$\varphi[\rho_2(s_t)]$
$\varphi[\rho_{N-1}(s_t)]$
$\varphi[\rho_N(s_t)]$
$Q(s_t,a_1)=\sum_{i=1}^{N} w_i^1\varphi[\rho_i(s_t)]+b_1^o$
$Q(s_t,a_2)=\sum_{i=1}^{N} w_i^2\varphi[\rho_i(s_t)]+b_2^o$
$Q(s_t,a_{|A|})=\sum_{i=1}^{N} w_i^{|A|}\varphi[\rho_i(s_t)]+b_{|A|}^o$

Figure 3: Structure of a deep Q-network. In RBF based deep Q-network, $\rho_i(s_t)=\|s_t-s_i\|$ is Euclidean distance and activation function $\varphi[\rho_i(s_t)]=k(\|s_t-s_i\|)$ is kernel function. While in deep Q-network constructed by other neural networks, $\rho_i(s_t)=\theta_i^T s_t+b_i^h$ is a linear transformation of its inputs s_t and activation function is typically the sigmoid function or the Rectified Linear Unit (ReLU) function.

3 DEEP Q-NETWORK BASED ON RADIAL BASIS FUNCTIONS

A radial basis function (RBF) network is used to construct the deep Q-network. The RBF based deep Q-network is a special three-layers network with N hidden neurons corresponding to states $\{s_1,s_2,\ldots,s_N\}$. As illustrated in Figure 3, the output of each hidden neuron is given by $\rho_i(s_t)=\|s_t-s_i\|$, which is the Euclidean distance between the current state s_t and the hidden neuron s_i. This is a key difference between the RBF based deep Q-network and Q-networks with other neural network architectures, such as fully connected neural networks (FCNs) or convolutional neural networks (CNNs), where the output of each hidden neuron is typically a linear transformation of its inputs s_t, given by $\rho_i(s_t)=\theta_i^T s_t+b_i^h$. Because the hidden layer neurons in the RBF based deep Q-network correspond to states, we can select lattice points from the minimum state to the maximum state to cover the entire state space. This makes the RBF based deep Q-network easy to design and implement in two ways. First, we don't need to determine the structure of hidden layers, such as the number of hidden layers and the number of neurons per hidden layer. Second, the real meaning of hidden layer neurons provides guidance for construction, in contrast to other neural network architectures which are primarily constructed based on intuition and experience. Overall, by using an RBF network to construct the deep Q-network, we aim to simplify the design and implementation process while leveraging the unique properties of RBF networks to effectively approximate the action-value function in the multi-echelon inventory management MDP.

Another key difference between RBF based deep Q-network and deep Q-networks with other neural network architectures is the activation function. In RBF based deep Q-network, the activation function $\varphi[\rho_i(s_t)] = k(\|s_t - s_i\|)$ is a kernel function that converts the Euclidean distance $\|s_t - s_i\|$ to a high dimensional space distance, while common activation functions used in other deep Q-networks are the sigmoid function or the Rectified Linear Unit (ReLU) function. Thus, each output of the RBF based deep Q-network, as shown in Figure 3, is a linear combination of N kernel functions, where each kernel function measures the high dimensional space distance between the current state s_t and the hidden neuron s_i. The most widely used kernel function is the radial basis kernel function, which is why it is called an "radial basis kernel" network. The radial basis kernel function is defined as:

$$k(s_t, s_i) = \exp\left(-\frac{\|s_t - s_i\|^2}{2\eta^2}\right).$$

The literature also suggests other kernel functions, for example, the Matérn(ν) kernel function. With gamma function $\Gamma(\cdot)$ and the modified Bessel function $K_\nu(\cdot)$, the Matérn(ν) kernel function is:

$$k_{\text{Matérn }(\nu)}(s_t, s_i) := \frac{1}{2^{\nu-1}\Gamma(\nu)}\left(\sqrt{2\nu}\left\|\eta^\top(s_t, s_i)\right\|\right)^\nu K_\nu\left(\sqrt{2\nu}\left\|\eta^\top(s_t, s_i)\right\|\right).$$

The Matérn kernel has a simplified form if ν is a half-integer: $\nu = p + \frac{1}{2}$ for some non-negative integer p, and the Matérn kernel becomes more differentiable as p increases. For instance, Matérn$(\frac{5}{2})$ kernel function is second-order differentiable:

$$k_{\text{Matérn }(\frac{5}{2})}(s_t, s_i) := \left(1 + \frac{\sqrt{5}\,\|s_t - s_i\|}{\eta} + \frac{5\,\|s_t - s_i\|^2}{3\eta^2}\right)\exp\left(-\frac{\sqrt{5}\,\|s_t - s_i\|}{\eta}\right). \tag{5}$$

η is a hyperparameter that determines the RBF based deep Q-network's smoothness. A smaller η results in a less smooth RBF based deep Q-network.

After constructing the RBF based deep Q-network, we train it by minimizing loss function $L(W)$, which is derived from Q-learning:

$$L(W) = \left(E[c_t(s_t, a_t)] + \gamma \min_a \hat{Q}(s_{t+1}, a; W) - \hat{Q}(s_t, a_t; W)\right)^2. \tag{6}$$

The loss function measures the difference between the current action value $\hat{Q}(s_t, a_t; W)$ and the predicted action value $E[c_t(s_t, a_t)] + \gamma \min_a \hat{Q}(s_{t+1}, a; W)$. The loss function is minimized via the gradient descent method. Considering a learning rate α, the weight vector W is updated by:

$$W_{t+1} = W_t + \alpha\left(E[c_t(s_t, a_t)] + \gamma \min_a \hat{Q}(s_{t+1}, a; W_t) - \hat{Q}(s_t, a_t; W_t)\right)\nabla\hat{Q}(s_t, a_t; W_t). \tag{7}$$

where $\nabla\hat{Q}(s_t, a_t; W_t)$ is the gradient of $\hat{Q}(s_t, a_t; W_t)$. As each output of RBF based deep Q-network is a combination of N kernel functions, it is easy to derive that the gradient is $(k(s_t, s_1), k(s_t, s_2), \ldots, k(s_t, s_N))$, a vector of N kernel function values.

Once the RBF based deep Q-network $\hat{Q}(s_t, a_t; W)$ is trained, the optimal order quantities at state s_t are selected by minimizing $\hat{Q}(s_t, a_t; W)$:

$$a_t^\star = \arg\min_{a_t} \hat{Q}(s_t, a_t; W) \tag{8}$$

4 SIMULATION STUDY

This section evaluates the performance of the proposed RBF based deep Q-network in three numerical experiments. The first experiment is a simple serial system with one warehouse and one retailer, while the other two experiments are complex systems involving multiple retailers. The difference between the two complex systems is that the third system's demands are more unstable than the second system, with a larger demand standard deviation and longer lead times. Previous studies have explored these systems using different DRL approaches: Van Roy et al. (1997) developed a neuro-dynamic programming approach for all three systems, and Gijsbrechts et al. (2022) applied the A3C algorithm to study the two complex systems. We adopt the same settings as these studies to compare with their DRL approaches.

Similar to Van Roy et al. (1997) and Gijsbrechts et al. (2022), we apply a baseline method and compare our approach's improvements against it. The baseline method is the base-stock policy, which means that for an installation (warehouse or retailer) with a base-stock level s, if the inventory position is less than s, the installation places orders to increase the inventory position to s as close as possible. It should be noted that in a serial system, the base-stock policy is optimal (Clark and Scarf 1960), while in a complex multi-echelon system, the optimal policy is unknown.

We select lattice states from the minimum state to the maximum state in hidden layers. The higher the state dimension, the more hidden layer neurons are used. Therefore, we reduce the state's dimension to reduce the hidden layer neurons. We set the state $s_t = (Z_t^w, Z_t^r)$, where $Z_t^w = I_t^w + \sum_{j=1}^{l_w} q_{t-j}^w$ is the warehouse's inventory position, and $Z_t^r = \sum_{i=1}^{K}(I_t^i + \sum_{j=1}^{l_r} q_{t-j}^i)$ is the total inventory position of all retailers. Thus, we reduce the states to two dimensions. We also reduce actions to two dimensions in a similar way. Actions are $a_t = (q_t^w, q_t^r)$, where q_t^w is the warehouse's order quantity, and q_t^r is every retailer's order quantity, which are the same for all retailers.

The kernel function in our RBF based deep Q-network is the Matérn($\frac{5}{2}$) kernel given by (5). The hyperparameter η that determines the RBF based deep Q-network's smoothness is set to 1 for all three experiments, suggesting that the RBF based deep Q-network is very unsmooth.

The simulation programs for the three experiments are implemented in C++. Details of the procedures are discussed in Section 2.1. The programs for the RBF-based deep Q-network algorithm are implemented in Python, and the ctypes library is used to call the C++ simulation programs. All experiments run on a 64-bit Linux machine with a 20×2.50GHz CPU and 12×16GB RAM.

4.1 Experiment With One Warehouse One Retailer Serial System

In the first experiment, the system consists of only one warehouse and one retailer. Furthermore, there is no lead time for the warehouse, and the retailer has only one period lead time. A detailed list of parameters is presented in Table 1.

Table 1: Settings of a serial system with one warehouse and one retailer.

	μ	σ	l_w	l_r	K	h_w	h_r	c_w	p	P_w	C^m	C^w	C^r
Setting 1	5	8	0	1	1	1	2	10	50	1	10	50	50

We selected lattice states as $\{(Z_i^w, Z_i^r) : Z_i^w = 0, 5, \ldots, 50, Z_i^r = 0, 5, \ldots, 50\}$. Since they are also hidden layer neurons, the hidden layers have $N = 121$ neurons. Regarding actions, we set the warehouse order quantity $q_t^w \in [0, 10]$ and retailer order quantity $q_t^r \in [0, 10]$. There are 121 possible actions in total, implying that the output layer's dimension is 121.

Figure 4 displays the average cost evolution during the training process, which takes a total of 1,837 seconds. As shown, the average cost of the RBF based deep Q-network (blue solid line) decreases in the first 2,000,000 periods and stabilizes near the cost of the base-stock policy (red dashed line). There is a slight gap between the RBF based deep Q-network and the base-stock policy in the serial system, where the base-stock policy is optimal.

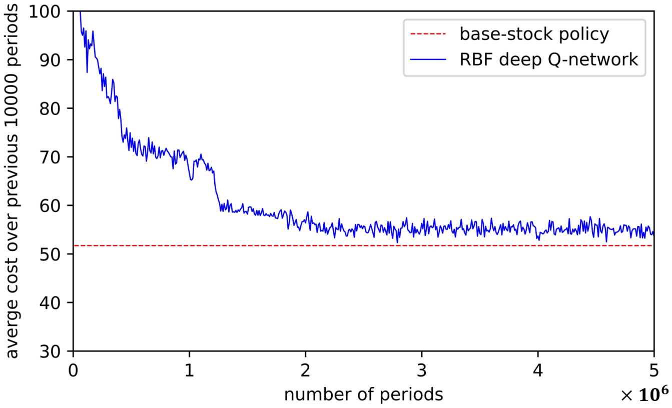


Figure 4: Average cost evolution during training.

We also compare the on-hand inventory of base-stock policy and RBF based deep Q-network. Table 2 report the average warehouse and retailer on-hand inventory of base-stock policy $(\bar{I}^w_{\text{BS}}, \bar{I}^r_{\text{BS}})$ and the average warehouse and retailer on-hand inventory of RBF based deep Q-network $(\bar{I}^w_{\text{RBF}}, \bar{I}^r_{\text{RBF}})$, respectively. As shown in the table, their on-hand inventory is similar, especially for the retailer on-hand inventory. Moreover, the average relative difference between their on-hand inventory is also reported in Table 2, and the difference is small. This indicates that the RBF based deep Q-network approach not only reduces costs to a near-optimal level but also orders and controls on-hand inventory like the base-stock policy. Thus, the RBF based deep Q-network learns a near-optimal solution in the serial system.

Table 2: Average on-hand inventory of base-stock policy and RBF based deep Q-network, and average relative difference between their on-hand inventory.

$\bar{I}^w_{\text{BS}}$	$\bar{I}^w_{\text{RBF}}$	$\bar{I}^r_{\text{BS}}$	$\bar{I}^r_{\text{RBF}}$	Average of $\frac{\lvert I^w_{\text{BS}} - I^w_{\text{RBF}}\rvert}{I^w_{\text{BS}}}$	Average of $\frac{\lvert I^r_{\text{BS}} - I^r_{\text{RBF}}\rvert}{I^r_{\text{BS}}}$
4.71	6.05	13.28	12.28	34.88 %	9.66 %

We calculate relative gap between RBF based deep Q-network and base-stock policy, and compare the result with Van Roy et al. (1997). Their approach has a 1.74% gap to base-stock policy, while our gap is 2.87%. It should be noted that Van Roy et al. (1997) manually developed three features of the system as state, while our approach can achieve a similar near-optimal solution without manual feature engineering.

4.2 Experiment With One Warehouse Multiple Retailers Multi-echelon System

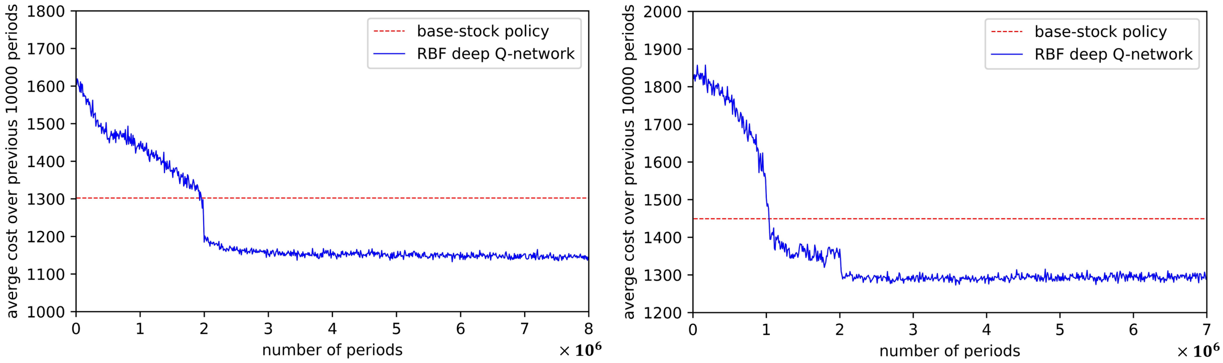
Next, we evaluate our approach in two systems, both with one warehouse and K identical retailers. Table 3 lists the parameter settings. In the system with Setting 2, the demands are more stable, with a smaller demand standard deviation and shorter lead times. In the system with Setting 3, the demands are very unstable, with a demand mean of zero and a very large demand standard deviation.

Table 3: Settings of one warehouse and multiple retailers system.

	μ	σ	l_w	l_r	K	h_w	h_r	c_w	p	P_w	C^m	C^w	C^r
Setting 2	5	14	2	2	10	3	3	0	60	0.8	100	1,000	100
Setting 3	0	20	5	3									

The selected lattice states (hidden neurons) for Setting 2 are $\{(Z_i^w, Z_i^r) : Z_i^w = 200, 220, \ldots, 400, Z_i^r = 100, 120, \ldots, 400\}$, and for Setting 3 are $\{(Z_i^w, Z_i^r) : Z_i^w = 300, 320, \ldots, 600, Z_i^r = 100, 120, \ldots, 300\}$. Each setting has $N = 176$ neurons in the hidden layer. The action ranges are set to $q_t^w \in [50, 100]$ and $q_t^r \in [0, 15]$ in Setting 2, while $q_t^w \in [40, 100]$ and $q_t^r \in [0, 15]$ in Setting 3. The output layer's dimensions are 816 and 976 in Setting 2 and Setting 3, respectively.

Figure 5 illustrates the average cost evolution of Setting 2 and Setting 3 during training, which takes a total of 3,832 seconds and 4,656 seconds, respectively. In both figures, the average costs of the RBF based deep Q-network (blue solid lines) initially reduce, then stabilize, and finally become lower than the base-stock policy costs (red dashed lines). This implies that the base-stock policy is no longer optimal for complex multi-echelon systems with multiple retailers.



(a) Average cost of Setting 2

(b) Average cost of Setting 3

Figure 5: Average cost evolution during training.

We also compare the on-hand inventory of base-stock policy and RBF based deep Q-network. Table 4 reports the average warehouse and retailer on-hand inventory of base-stock policy $(\bar{I}^w_{\text{BS}}, \bar{I}^r_{\text{BS}})$ and the average warehouse and retailer on-hand inventory of RBF based deep Q-network $(\bar{I}^w_{\text{RBF}}, \bar{I}^r_{\text{RBF}})$ in Setting 2 and Setting 3. In both settings, the average warehouse on-hand inventory of RBF based deep Q-network is lower than the base-stock policy. This implies warehouses controlled by RBF based deep Q-network order less and can achieve lower warehouse holding costs. Regarding the average retailer on-hand inventory, in Setting 2, the RBF based deep Q-network and base-stock policy are the same, while in Setting 3, the base-stock policy is lower.

Table 4: Average on-hand inventory of base-stock policy and RBF based deep Q-network in two settings.

	$\bar{I}^w_{\text{BS}}$	$\bar{I}^w_{\text{RBF}}$	$\bar{I}^r_{\text{BS}}$	$\bar{I}^r_{\text{RBF}}$
Setting 2	157	109	106	106
Setting 3	154	123	110	134

It is worth noting that we find in both Setting 2 and Setting 3, the larger the warehouse on-hand inventory, the more retailers will order. This reduces warehouse holding costs because once the warehouse on-hand inventory is large, the retailers will order more to reduce warehouse on-hand inventory. This also implies that a retailer controlled by RBF based deep Q-network considers both the warehouse and retailer inventory when making decisions, while a retailer following the base-stock policy only considers its own inventory to make decisions. Hence, the RBF based deep Q-network achieves lower costs by learning more information.

Furthermore, we calculate the relative improvement of RBF based deep Q-network compared to the base-stock policy and compare our relative improvement with current DRL approaches. Table 5 shows the relative improvement of different DRL approaches. The RBF based deep Q-network is slightly better than both neuro-dynamic programming and A3C in Setting 2 and is as good as A3C in Setting 3. It's worth noting that Van Roy et al. (1997) manually developed 23 features for the neuro-dynamic programming approach, while both the RBF based deep Q-network and A3C do not require manual feature engineering. Additionally, the process of designing the neural network in A3C is complex. As pointed out by Gijsbrechts et al. (2022), tuning to select the number of hidden layers and neurons per layer remains computationally burdensome. The tuning and training time for A3C can be days or even weeks. In contrast, the RBF based deep Q-network does not require special design for the neural network structure. Thus, the RBF based deep Q-network is easier to implement. The training process for Setting 2 and Setting 3 takes only 3832 seconds and 4656 seconds, respectively, which is significantly less than the A3C algorithm.

Table 5: Relative improvement of different DRL approaches.

	RBF Based Deep Q-network	Neuro-dynamic Programming (Van Roy et al. 1997)	A3C (Gijsbrechts et al. 2022)
Setting 2	12 %	10 %	9 %
Setting 3	12 %	10 %	12 %

5 CONCLUSION

This paper proposes a deep Q-network approach based on RBF to solve dynamic inventory management for general multi-echelon systems. The RBF based deep Q-network has a simple structure and can be easily constructed. Simulation studies show that our method performs better than the base-stock policy in multi-echelon systems with multiple retailers. Meanwhile, we also compared the RBF based deep Q-network with current DRL approaches and find that the RBF based deep Q-network has appealing performance compared to existing DRL approaches and is easier to design. These demonstrate the potential use of our RBF based deep Q-network for solving practical inventory management problems.

ACKNOWLEDGEMENT

This work was partially supported by the National Natural Science Foundation of China [No. 72031006, 72071146] and Alibaba Innovative Research (AIR) Project.

REFERENCES

Broomhead, D. S., and D. Lowe. 1988. "Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks". *No. RSRE-MEMO-4148*:1–34.

Clark, A. J., and H. Scarf. 1960. "Optimal Policies for a Multi-Echelon Inventory Problem". *Management Science* 6(4):475–490.

Gijsbrechts, J., R. N. Boute, J. V. Mieghem, and D. J. Zhang. 2022. "Can Deep Reinforcement Learning Improve Inventory Management? Performance on Lost Sales, Dual-Sourcing, and Multi-Echelon Problems". *Manufacturing and Service Operations Management* 24(3):1349–1368.

Liu, X., M. Hu, Y. Peng, and Y. Yang. 2022. "Multi-Agent Deep Reinforcement Learning for Multi-Echelon Inventory Management". *Working Paper*. https://www.researchgate.net/publication/365025212_Multi-Agent_Deep_Reinforcement_Learning_for_Multi-Echelon_Inventory_Management, accessed 31st August 2023.

Mcgrath, T., A. Kapishnikov, N. Tomaev, A. Pearce, D. Hassabis, B. Kim, U. Paquet, and V. Kramnik. 2022. "Acquisition of Chess Knowledge in AlphaZero". *Proceedings of the National Academy of Sciences* 119(47):e2206625119.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 2013. "Playing Atari with Deep Reinforcement Learning". *arXiv preprint arXiv:1312.5602*. https://arxiv.org/abs/1312.5602.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglo, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. 2015. "Human-level control through deep reinforcement learning". *Nature* 518(7540):529–533.

Oroojlooyjadid, A., M. R. Nazari, L. V. Snyder, and M. Taká. 2021. "A Deep Q-Network for the Beer Game: Deep Reinforcement Learning for Inventory Optimization". *Manufacturing & Service Operations Management* 24(1):285–304.

Powell, W. B. 2011. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York: John Wiley & Sons.

Qiu, J., J. Xia, J. Luo, Y. Liu, and Y. Liu. 2022. “Integrated Inventory Placement and Transportation Vehicle Selection Using Neural Network”. In *IEEE 18th International Conference on Automation Science and Engineering (CASE)*. 20th–24th August, Mexico City, Mexico, 1601–1608.

Sherbrooke, C. C. 1968. “Metric: A Multi-Echelon Technique for Recoverable Item Control”. *Operations Research* 16(1):122–141.

Simchi-Levi, D., and Y. Zhao. 2012. “Performance Evaluation of Stochastic Multi-Echelon Inventory Systems: A Survey”. *Advances in Operations Research* 2012(5):126254.

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. Cambridge, USA: MIT press.

Van Roy, B., D. P. Bertsekas, Y. Lee, and J. N. Tsitsiklis. 1997. “A Neuro-Dynamic Programming Approach to Retailer Inventory Management”. In *Proceedings of the 36th IEEE Conference on Decision and Control*. 12th December, San Diego, CA, 4052–4057.

Wang, Q., Y. Peng, and Y. Yang. 2022. “Solving Inventory Management Problems through Deep Reinforcement Learning”. *Journal of Systems Science and Systems Engineering* 31(6):677–689.

Zipkin, P. H. 2000. *Foundations of Inventory Management*. New York: McGraw-Hill Companies.

AUTHOR BIOGRAPHIES

LIQIANG CHENG is a master student in the Antai College of Economics and Management at Shanghai Jiao Tong University. His email address is chengzi1998@sjtu.edu.cn.

JUN LUO is a professor in the Antai College of Economics and Management at Shanghai Jiao Tong University. His primary research interest are simulation optimization and statistics. His email address is jluo_ms@sjtu.edu.cn.

WEIWEI FAN is an associate professor in the Advanced Institute of Business and School of Economics and Management at Tongji University. Her primary research interest are simulation optimization and robust optimization. Her email address is wfan@tongji.edu.cn.

YIDONG ZHANG is a director of the B2C Supply Chain Optimization team and the Supply Planning Optimization team at Alibaba Group. His email address is tanfu.zyd@alibaba-inc.com.

YUAN LI is an algorithm engineer in the Dchain Department at Alibaba Group. His email address is yuan.lya@alibaba-inc.com.