Single-site Perishable Inventory Management under Uncertainties: A Deep Reinforcement Learning Approach

Kaixin Wang, Cheng Long, Darrell Joshua Ong, Jie Zhang, and Xue-Ming Yuan

Abstract—Online lot sizing for perishable materials in an uncertain environment is a fundamental problem for inventory planning and has been studied in the past several decades. In this paper, we study a novel setting of the lot sizing problem, considering perishable materials, multiple suppliers, uncertain demands and lead time (LS-PMU), which captures the inventory planning task in real life better than existing lot sizing problems. We present theoretical results of the best possible competitive ratio an online algorithm can achieve for LS-PMU problem. We then develop a reinforcement learning based algorithm called `RL4LS` to intelligently choose the supplier and decide the order quantity in each time period. We conduct extensive experiments on both real and synthetic datasets to verify that `RL4LS` outperforms existing algorithms in terms of effectiveness and efficiency, e.g., `RL4LS` improves the effectiveness by 44% and runs two orders of magnitude faster than the state-of-the-art algorithm `IBFA`.

Index Terms—supply chain optimization, inventory management, lot sizing, deep reinforcement learning

1 Introduction

Supply chain management is to manage the flow of raw materials, semi-finished and finished products. During the process of supply chain management, one essential task is *inventory planning*, which involves a critical problem called *lot sizing* [1]. Specifically, in a lot sizing problem, a decision-maker needs to determine in each period the orders to be placed and their quantities, considering the lead time (which means the time needed for producing and transporting the material from the supplier to the inventory), associated costs, products' quality, to minimize the total costs. Depending on the application scenarios, different assumptions, objectives, and constraints are adopted [2]. Most existing works [3], [4], [5], [6] assume a single-supplier scenario, where there is only one supplier available across different periods. Some recent works [7], [8] target a multiple-supplier scenario. Specifically, [7], [8] assume that the decision-maker can order materials from *multiple* suppliers in each *single* period. We observe that this multiple-supplier scenario does not capture the practice that well. For example, for many enterprises, such as ST Logistics, Alcon, LSH Electrical Engineering, they always view the order in a period as an integrated one and place it to the most suitable supplier, but not multiple ones for the following reasons.

First of all, it would save the ordering costs. Most suppliers charge a fixed cost, i.e., a setup cost, in addition to the costs of purchasing the materials, which usually cover the expenses incurred in production and shipment. Therefore, when we distribute the orders to multiple suppliers in a period, we may not only pay more money for buying these materials (since different suppliers charge different unit costs), but also have a higher expenditure for the production and transportation from different suppliers. Second, it would introduce less work of coordination when receiving orders. Imagine that we order from multiple suppliers in every period. Then for some periods, we may need to handle the situation that a number of receiving orders from different suppliers arrive at the inventory simultaneously, which makes the coordination a tough job. Third, it would make it convenient for after-sale services. When the customers want to return or exchange their products due to some reasons, it would be easier to contact the supplier.

In this paper, we propose a new problem, called the lot sizing with perishable materials, multiple suppliers and uncertain demands and lead time (LS-PMU). We adopt the setting that one only places the order of the material to at most one supplier in each planning period, which is different from that adopted in existing studies [7], [8]. Consider a planning scenario of T periods with a set of suppliers, denoted by S. Existing studies usually adopt a vector-based solution, which represents the policy as a $|S|$-dimensional vector in each planning period [7], [8], but they apply different search strategies. Specifically, `GA` [7] applies the genetic algorithm, where a possible solution is encoded as a chromosome-like data structure, represented by a $(|S| \times T)$-dimensional vector. `IBFA` [8] applies an improved bacteria foraging algorithm, where the raw solutions, intermediate results and the global optimum are all represented by $(|S| \times T)$-dimensional vectors. These solutions, however, cannot handle our task well since we do not allow to place orders to multiple suppliers in a single period. In addition, [7], [8] are both based on the nature-inspired heuristics, and it would take a long time to converge.

To address the LS-PMU problem effectively and effi-

- *K. Wang, C. Long and J. Zhang are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore Email: {kaixin.wang, c.long, zhangj}@ntu.edu.sg.*
- *D. J. Ong and X. Yuan are with Singapore Institute of Manufacturing Technology, Singapore. Email: {darrell_joshua_ong, xmyuan}@simtech.a-star.edu.sg.*
- *Corresponding Author: Cheng Long*

ciently at the same time, we consider the planning process of LS-PMU problem as a sequential decision making process. Then, we model the sequential decision making process as a Markov Decision Process (MDP) [9]. We carefully design the MDP including states, actions and rewards such that (1) the states capture the critical information and cheap to compute; (2) the *hybrid* actions capture the decision process of LS-PMU (i.e., first choosing a supplier, which is captured by a discrete value, and then deciding an order quantity, which is a continous value); and (3) the rewards are well aligned with the goal of the problem. Finally, we adopt a reinforcement learning (RL) algorithm, namely parameterized Deep-Q Network (P-DQN) [10], to learn the policy.

In summary, the main contributions of this paper are as follows. We propose the LS-PMU problem, in which we place an order to at most one supplier in each period, which is more aligned with real logistics applications. We present theoretical results on the competitive ratio of any given algorithm for LS-PMU problem, which could reflect the best possible quality guarantee for the algorithm. We propose a RL-based algorithm `RL4LS` for the LS-PMU problem. Compared with existing methods, our algorithm can capture more information of the inventory status and as a result, can generate the policies with smaller costs. We conduct extensive experiments on both real and synthetic datasets to verify that `RL4LS` outperforms all (resp. most) existing algorithms in terms of effectiveness (resp. efficiency). For example, `RL4LS` improves the effectiveness by 44% and simultaneously runs 226 times faster on the real dataset, compared with the state-of-the-art `IBFA`.

2 Related Work

Lot-sizing Problems. Lot sizing problems can be classified based on several criteria. Among them, [11], [12], [13], [14] target non-perishable materials. For perishable materials, earlier studies on the lot-sizing problem assume an *offline* setting, where the demands in each period are known in advance [15], [16], [17], [18]. Recently, some studies [3], [4], [5], [6], [19], [20], [21], [22] consider the online setting. However, [19], [20], [21], [22] target the materials whose quality deteriorates continuously over time, i.e., $N(t) = N(0) \cdot e^{-\lambda t}$, where $N(0)$ and $N(t)$ are the initial quality and the quality at time t, respectively. [3], [4], [5], [6] target the materials that have a fixed expiry date, but they assume a *single* supplier. The most related studies are [7], [8] since they also target the materials that have a fixed expiry date with multiple suppliers. Nevertheless, as explained in Section 1, these studies allow to place orders to more than one suppliers in a single period, which do not capture some real scenarios well. Interested readers are referred to a survey paper [2] for more details of other types of lot-sizing problems.

Reinforcement Learning for Inventory Planning. Some existing studies [5], [13], [14], [23] have tried to solve the inventory planning problem using reinforcement learning, they differ from our `RL4LS` algorithm in various aspects: (1) [13], [14] allow to place orders to more than one suppliers in a single period, and they focus on non-perishable materials; (2) [5] targets single-supplier setting and assumes zero or constant lead time; (3) [23] targets a multiple-echelon scenario that multiple inventories cooperate to minimize the total costs or maximize the revenue.

3 Problem Statement

3.1 Variables

Consider a horizon of time periods $1, 2, ..., T$, where each corresponds to a unit of time such as a month. Let S be a set of suppliers, from which we can order materials. During each period, the materials may be involved in some events, e.g., it is ordered from some supplier, delivered to the inventory, sold to customers, and disposed. The following variables represent the quantities involved in these events.

We denote by $o(i, j)$ and $l(i, j)$ the quantity and lead time of the material ordered from supplier $j \in S$ at the beginning of period $i \in [1, T]$, respectively. The lead time $l(i, j)$ is a random variable, meaning the amount of time it takes for the material that is ordered at the beginning of period i to be delivered from the supplier j to the inventory. Note that the order $o(i, j)$ will be received at $i' = i + l(i, j)$.

We denote by $v(i)$ the quantity of the material stored in the inventory at the beginning of period i. We further define $v(i, r)$ to be the quantity of material which (1) is stored in the inventory at the beginning of period i and (2) has the remaining life time $r \in [1, L]$, where L is the maximum life time of the material. Note that we have $v(i) = \sum_{r=1}^{L} v(i, r)$. We assume the initial inventory level is zero, i.e., $v(1) = 0$.

We denote by $a(i)$ the quantity of the material that arrives at the inventory at the beginning of period i. We assume no material expires before it arrives at the inventory, which is well aligned with the practice. This quantity depends on how the material is ordered and delivered before period i, i.e., $o(i', j)$ and $l(i', j)$ for $i' \leq i$ and $j \in S$. We also define $a(i, r)$ to be the quantity of material which (1) arrives the inventory at period i and (2) has the remaining life time $r \in [1, L]$. Similarly we have $a(i) = \sum_{r=1}^{L} a(i, r)$.

We denote by $d(i)$ the quantity of the material that is demanded during period i. The quantities for the future periods are unknown. We denote by $s(i)$ the quantity of the material that is in shortage during period i. Note that $s(i) = \{d(i) - v(i) - a(i)\}^+$, where $\{x\}^+ = \max\{0, x\}$. Here, we target the lost-sales scenario, where the unfilled demands would be dropped. We denote by $u(i)$ the quantity of the material that is used to serve the demands during period i. Note that $u(i) = \min\{d(i), v(i) + a(i)\}$. We denote by $p(i)$ the quantity of the material that is disposed at the end of period i. This quantity depends on the quantity of material that will expire at the end of this period, i.e., $v(i, 1) + a(i, 1)$, and the demand during this period, i.e., $d(i)$. Specifically, $p(i) = \{v(i, 1) + a(i, 1) - d(i)\}^+$. Here, we adopt the principle that material with shorter remaining life time is used to serve the demands first.

Note that the quantity of the material at the beginning of the next time period $i + 1$, i.e., $v(i + 1)$, depends on $v(i)$ (existing), $a(i)$ (newly arrived), $u(i)$ (sold), and $p(i)$ (disposed). Specifically, we have the following equation.

$$v(i + 1) = v(i) + a(i) - u(i) - p(i). \tag{1}$$

An illustration of the above events is shown in Figure 1.

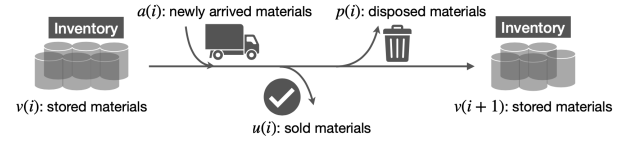


Fig. 1. An illustration of the events between two adjacent time periods.

3.2 Costs

Due to the life time of the material, various costs are incurred, including ordering cost, holding cost, disposal cost and shortage cost for each period i. We denote these costs by $C_o(i)$, $C_h(i)$, $C_d(i)$, and $C_s(i)$, respectively. Let $c_o(j)$, c_h, c_d, and c_s be the unit cost of ordering from supplier j, holding, disposing, and having a shortage of the material, respectively. Let $c_b(j)$ be the fixed setup cost that would be incurred when a certain quantity of material is ordered from the supplier j. Then, the costs can be computed as follows.

$$\begin{aligned} C_o(i) &= \sum\nolimits_{j\in S}(c_o(j)\cdot o(i,j) + c_b(j)\cdot I(o(i,j))) \\ C_h(i) &= c_h\cdot\{v(i)+a(i)-d(i)\}^+ \\ C_d(i) &= c_d\cdot\{v(i,1)+a(i,1)-d(i)\}^+ \\ C_s(i) &= c_s\cdot\{d(i)-v(i)-a(i)\}^+ \end{aligned} \tag{2}$$

where $I(x)$ is an indicator function, which is equal to 1 if $x > 0$ and 0 otherwise. Then, the cost for time period i, which we denote by $C(i)$, can be computed as follows.

$$C(i) = C_o(i) + C_h(i) + C_d(i) + C_s(i) \tag{3}$$

3.3 Problem Definition

In this paper, we study the problem of lot sizing with perishable materials, multiple suppliers, and uncertain demands and lead time (LS-PMU). Specifically, the problem is to decide for each time period a supplier and a quantity for ordering a material in an online fashion so as to minimize the total cost over all time periods subject to the inventory capacity constraint and the principle that during each time period, we order the material from at most one supplier. Mathematically, the problem could be formalized as follows.

$$\min_{o(i,j),i\in[1,T],j\in S} \sum\nolimits_{i=1}^{T} C(i) \quad \text{s.t.} \tag{4}$$

$$v(i) \le V \text{ for each } i \in [1,T] \tag{5}$$

$$\sum\nolimits_{j\in S} I(o(i,j)) \le 1 \text{ for each } i \in [1,T] \tag{6}$$

where V is the capacity for the material and $I(x)$ is an indicator function, which is equal to 1 if $x > 0$ and 0 otherwise. In addition, we summarize settings of our problem. (1) We target an *online* planning setting; (2) we target the *lost-sale* scenario, where the customers would not wait for the stock to be replenished and drop the unfilled demands; (3) the planning is in a *periodic-review* manner.

3.4 Competitive Ratio Analysis

In the following, we present a competitive ratio boundary that an algorithm is able to achieve for the LS-PMU problem.

Theorem 1. *Suppose that* $\min_{j\in S} c_o(j) \le c_s$ *(since otherwise the competitive ratio would be exactly 1). Given an algorithm* $A \in \mathcal{A}$ *for the LS-PMU problem with decision variables* $o(i,j)$ *for all* i *and* j*. Define* $o(i) = \sum_{j\in S} o(i,j)$ *and*

$$cr(i) = \max\left(\frac{f(o(i))}{f(0)}, \frac{c_s}{\min_{j\in S}\{c_o(j)\}}\right) \tag{7}$$

where $f(x) = \min_{j\in S}\{(c_h + c_d + c_o(j))\cdot x + c_b(j)\}$*, then the competitive ratio of algorithm* A *is at least* $\min_i cr(i)$*.*

We provide the proof of Theorem 1 and an offline optimal algorithm A^* in the technical report [24]. With A^*, we can compute the empirical competitive ratio of an algorithm.

4 METHODOLOGY

We observe that the planning task in the LS-PMU problem corresponds to a sequential decision process, i.e., it makes decisions on (1) choosing a supplier and (2) determining the order quantity for placing an order on the chosen supplier in each period sequentially. Therefore, we propose to use RL to help with the decision making process. Specifically, we model the LS-PMU problem as an MDP, adopt an existing deep RL method, P-DQN [10], to learn an optimal policy on the MDP, and then develop an algorithm called `RL4LS`, which uses the learned policy to solve the LS-PMU problem.

4.1 The LS-PMU Problem Modeled as an MDP

We model the LS-PMU problem as an MDP, which mainly consists of states, actions and rewards as defined as follows.

States. We denote the state in time period i by s_i. Intuitively, a state should capture essential information of the inventory status for determining a supplier and the order quantity in a period. We identify the following three types of information: (1) the inventory level, (2) open orders, which have been ordered but not received by the inventory and (3) the prediction of the lead time and demands.

To capture the first type of information of the inventory status at the beginning of period i, we take the values $v(i,r)$ for $r \in [1,L]$ and concatenate them together, denoted by s_i^v,

$$s_i^v = [v(i,1), \cdots, v(i,r), \cdots v(i,L)] \tag{8}$$

The rationale is that s_i^v provides an overview of the materials that can be directly consumed in the current time period.

To capture the second type of information of the inventory status, we maintain a list of open orders $\mathcal{O}$. Based on the assumptions that (1) no material expires before they arrive at the inventory and (2) we order the material from at most one supplier in each period, there would be at most $L-1$ open orders at the beginning of the time period i. Let $b(k)$ be a binary value, where $b(k) = 1$ indicates the order placed at time period $k < i$ is an open order and $b(k) = 0$ otherwise. Then, the open order information s_i^o is defined as follows,

$$s_i^o = [b(i-L+1)\cdot o(i-L+1), \cdots, b(i-1)\cdot o(i-1)] \tag{9}$$

where $o(\cdot) = \sum_{j\in S} o(\cdot, j)$. The rationale is that s_i^o captures the quantity of materials that would arrive at the inventory in the following time periods, which is important for planning the future orders. Note that for those $r \in [1, L-1]$ with $i - r \le 0$, we assume the open orders are equal to 0.

To capture the third type of information of the inventory status, we make predictions on the lead time and the demand. The rationales are as follow. Lead time prediction of a supplier j provides the information of how long it will take to deliver the materials if we place the order to j, which would affect the decision afterwards. Demand prediction provides the information of how many materials we need to meet in the following period, which would further affect the choice of the supplier since if there is a surge demand, we need to choose the supplier with a short lead time. Formally, the predicted lead times, denoted by $\hat{l}(i,j)$ for $j \in S$, the predicted demand, denoted by $\hat{d}(i)$, and corresponding state representations s_i^l and s_i^d are defined as follows.

$$s_i^l = [\hat{l}(i,1), \cdots, \hat{l}(i,|S|)], \quad s_i^d = [\hat{d}(i)] \tag{10}$$

We try different methods for lead time and demand predictions, e.g., ARIMA [25], LSTM [26] and sampling from known distribution. We compare the results and report them in the experiments. In summary, we define s_i as a $(2L+|S|)$-dimension vector, which captures the inventory level information, open order information and the predictions,

$$s_i = [s_i^v, s_i^o, s_i^l, s_i^d] \in \mathbb{R}^{2L+|S|} \tag{11}$$

Actions. We denote the action at time period i by a_i. Recall that at the beginning of the time period i, the state is s_i and we need to decide (1) a supplier and (2) the order quantity. Therefore, we define a_i as follows.

$$a_i = [j, q] \quad (j \in S, q > 0) \tag{12}$$

which means we place an order to the supplier j in which the quantity of the material is q. We note that the actions defined above are *hybrid* ones, which involve both a discrete value j (which indicates a supplier) and a continuous value q (which means the order quantity).

Rewards. Consider that we take action a_i at a state s_i and then we arrive at a new state s_{i+1}. We define the reward, denoted by R_i, associated with this transition as follows.

$$R_i = -C(i) \tag{13}$$

The intuition is if a smaller cost is incurred in period i, the action a_i would be associated with a larger reward. It is worthy of mentioning that with the reward defined as above, the objective of the MDP, which is to maximize the accumulative rewards, would be equivalent to that of the LS-PMU problem, which is to minimize the costs over all periods. To see this, suppose that we go through a sequence of states $s_1, s_2, \cdots, s_{T+1}$ and correspondingly we receive a sequence of rewards $R_1, R_2, \cdots, R_T$. Then, the accumulative rewards without being discounted can be computed by

$$\sum_{i=1}^{T} R_i = -\sum_{i=1}^{T} C(i). \tag{14}$$

4.2 Policy Learning on the MDP

In our MDP, the states are high dimensional vectors and the actions are in a hybrid domain. Therefore, we adopt P-DQN [10] to solve our MDP as it targets the same setting. P-DQN involves a Q network and a policy network μ. The Q network, parameterized by ω, is to estimate the action-value function, and the policy network μ, parameterized by θ, is to map the state to an order quantity given a supplier in a deterministic way. The μ network given a supplier j is denoted by μ_j. Thus, given a state s_i in period i, we can select a proper action by first choosing a supplier j^* based on the Q network, and then calculating the order quantity q^* based on the μ networks, i.e.,

$$j^* = \arg\max_{j \in S} Q(s_i, j, \mu_j(s_i; \theta); \omega), \quad q^* = \mu_{j^*}(s_i; \theta) \tag{15}$$

We then take the action $a_i = [j^*, q^*]$ in period i. For the training process, we initialize two main networks $Q(\cdot; \omega)$ and $\mu(\cdot; \theta)$, which are used for selecting actions, and two target networks $Q'(\cdot; \omega')$ and $\mu'(\cdot; \theta')$, which are used for calculating the losses for training the main networks. During the training, we adopt the ϵ-greedy method, which takes $a = [j^*, q^*]$ with probability $(1-\epsilon)$ and $a = [j, q]$ other than $[j^*, q^*]$ with probability ϵ, to balance exploration and exploitation. We also maintain a replay buffer, containing the latest transitions. The training process is as follows. Consider N experiences sampled uniformly from the buffer, i.e., (s_i, a_i, R_i, s_{i+1}) for $i \in [1, N]$. For the Q network, we compute the loss by

$$L(\omega) = \frac{1}{N} \sum_{i=1}^{N} (y_i - Q(s_i, a_i; \omega))^2, \tag{16}$$

where $y_i = R_i + \gamma \cdot \max_{j' \in S} Q'(s_{i+1}, j', \mu'_{j'}(s_{i+1}; \theta'); \omega')$. For the μ network, we compute the loss by

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{|S|} Q(s_i, j, \mu_j(s_i; \theta); \omega) \tag{17}$$

Finally, we update θ and ω by gradient descent.

4.3 The RL4LS Algorithm

We put the pseudo-codes of RL4LS algorithm in the technical report [24]. We briefly introduce the main idea. At each time period i, RL4LS first observes the inventory status based on the three types of information, and constructs a state s_i. Then it takes an action $a_i = [j^*, o^*]$ based on the learned policy, i.e., Eq. (15). The plan for the time period i will be $o(i, j^*) = q^*$ and $o(i, j) = 0$ for other supplier j. Finally, we observe the costs and proceed to the next period.

Time complexity. The time complexity of the RL4LS algorithm is $O(T(L + |S|))$. The cost mainly comes from the computation of the states. Given a state $s = [s^v, s^o, s^l, s^d]$, we analyze the costs as follows. For s^v and s^o, they both take $O(L)$ time to construct. For s^l and s^d, they can be calculated in $O(|S|)$ and $O(1)$ time, respectively. Therefore, the complexity of the RL4LS is $O(T(L + |S|))$.

5 Experiment

5.1 Experiment Setup

Datasets. The real dataset is collected by Alcon Singapore Manufacturing Pte. Ltd. There are 76 different materials, and for each material, there are 76 different suppliers. Different materials have different demand distributions, maximum life time and associated costs. And different suppliers also have different lead time distributions. The planning horizon is one year. Since orders are placed on a monthly basis, we set $T = 12$. We also generate some synthetic datasets by following the existing study [27], which considers a sequence T of 12 periods and use different settings of $|S|$ for generating suppliers. For each setting, we further generate the same number of materials. Details of generating the suppliers and materials are referred to [27].

Baselines and Metrics. We compare our proposed model with four existing algorithms, namely IBFA [8], GA [7], PG4LS [13] and QL4LS [5], in terms of the total cost and the running time. IBFA, GA and PG4LS are all vector-based solutions, i.e., they allow to place orders to more than one supplier in a single period. The policy for a period i can be represented by a vector $[p_1, p_2, \cdots, p_{|S|}]$. To adapt them to our problem, we first set the order quantity to $q_i = \sum_{j=1}^{|S|} p_j$, and then we choose the supplier which has the minimum ordering cost, i.e., $j_i = \arg\min_{j \in S}\{c_o(j) \cdot q_i + c_b(j)\}$, so as

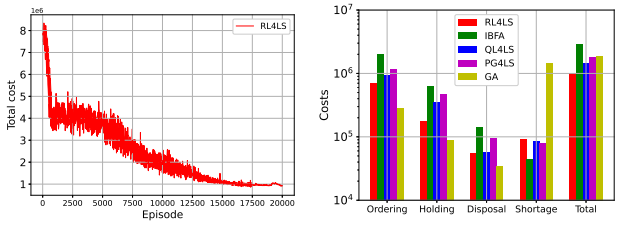
TABLE 1
Running time (seconds) on real dataset.

Alg.	RL4LS	IBFA	QL4LS	PG4LS	GA
Time (s)	2.39	542.27	1.78	2.77	843.74

to align with the objective of LS-PMU. QL4LS is the state-of-the-art RL based algorithm for the perishable material under the single-supplier setting. The policy for a period is a real number representing the order quantity, say q_i. To adapt QL4LS to our problem, we first set the order quantity to q_i, and then we choose the supplier which generates the least ordering costs, i.e., $j_i = \arg\min_{j \in S}\{c_o(j) \cdot q_i + c_b(j)\}$.

Model Training and Hyperparameter Setting. They can be found in the technical report [24]. Our codes can be found via https://github.com/wangkaixin219/RL4LS.

5.2 Experiment Results



(a) Learning Curve (b) Cost Decomposition

Fig. 2. Results on real dataset.

(1) Results on Real Dataset. The average time of training a satisfactory model is around 1.4 hours and the learning curve for RL4LS on real dataset is presented in Figure 2(a). We can see that the total cost gradually decreases over the training process, and the algorithm converges after 18,000 episodes. Consider the total costs (Figure 2(b)), RL4LS outperforms all existing algorithms. For all but GA, the costs mainly come from the holding costs and ordering costs, which are almost 80% of the total costs. When we order the materials more than the quantity we really need, it incurs higher purchasing costs and holding costs. For GA, the cost mainly comes from the shortage costs since it orders much fewer materials but they cannot meet the demands. The possible reasons are (1) GA gradually stucks at the local optimum and (2) the unit cost of having a shortage of the material, i.e., c_s, is around 7-9 times of that of purchasing, i.e., $c_k(j)$. Consider the efficiency (Table 1). QL4LS runs the fastest, which could be explained by that (1) GA and IBFA need a large amount of time for trial-and-error and (2) the state and the action are both cheaper to compute compared with RL4LS and PG4LS. In addition, RL4LS runs the second fast, only slightly slower than QL4LS.

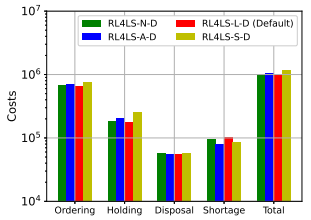
(2) Ablation Study on Prediction Models (Real Dataset). Recall that in Section 4.1, we make the predictions on the lead time and the demand in our state definition. We choose three models, namely ARIMA [25], LSTM [26] and Gaussian Sampling. For ARIMA(p, d, q), we set $p = 1, d = 0, q = 1$ for both predictions. For the demand prediction (resp. lead time prediction of a supplier) using LSTM, the input is a 12-dimensional (resp. 3-dimensional) vector, which consists of the demand history data of the last 12 periods (resp. the last 3 lead time history data of the same supplier). For Gaussian Sampling, we assume that the demand and lead time both follow some Gaussian distributions. We denote the variants by RL4LS-N/A/L/S-D/L, where N (A, L or S) represents we use a null (ARIMA, LSTM or Sampling) prediction model, and D (L) represents the model is for the demand (lead time) prediction. Consider the costs (Figure 3(a) and Figure 3(b)). For the demand prediction, the model with LSTM performs the best, which could be explained by the fact that LSTM can make a more accurate prediction, and this information could be used to decide the actions with the least costs. For lead time prediction, however, using sampling outperforms the others. We observe that LSTM and ARIMA do not work for lead time prediction. The reason is that the lead time is always affected by some unpredictable factors, such as weather or traffic, which makes the prediction unreliable. First four rows of Table 2 shows all variants are efficient.

TABLE 2
Running time (seconds) on ablation study.

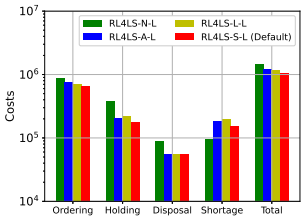
Alg.	RL4LS-N-D	RL4LS-A-D	RL4LS-L-D	RL4LS-S-D
Time (s)	2.11	2.26	2.39	2.18
Alg.	RL4LS-N-L	RL4LS-A-L	RL4LS-L-L	RL4LS-S-L
Time (s)	1.98	2.57	2.74	2.39
Alg.	$S_1 + A_1$	$S_1 + A_2$	$S_2 + A_1$	$S_2 + A_2$
Time (s)	2.39	2.24	2.90	2.77
Alg.	$S_1 + A_1$	$S_1 + A_3$	$S_3 + A_1$	$S_3 + A_3$
Time (s)	2.39	2.26	1.95	1.78

(3) Ablation Study on State and Action Definitions (Real Dataset). To evaluate the effectiveness of the state and action definitions of RL4LS, separately and collectively, we replace each of them with some alternative definitions, namely those adopted in PG4LS [13] and QL4LS [5]. We denote the state and action definitions of RL4LS (PG4LS, QL4LS) by S_1 and A_1 (S_2 and A_2, S_3 and A_3). In this experiment, we explore two groups of combinations. In group one, we explore four combinations of S_1, S_2 and A_1, A_2. In group two, we explore four combinations of S_1, S_3 and A_1, A_3, Consider the effectiveness (Figure 3(c) and Figure 3(d)). We observe that $S_1 + A_1$ (i.e., RL4LS) performs the best in both groups while $S_2 + A_2$ (i.e., PG4LS) and $S_3 + A_3$ (i.e., QL4LS) perform the worst in each group, respectively. For both groups, RL4LS has the least ordering and holding costs, which dominate the total costs. This could be possibly explained by that a state of RL4LS captures richer and more relevant information of the inventory and the action orders the materials timely and adequately. Consider the efficiency (last four rows of Table 2). For group one, we observe that $S_1 + A_2$ runs the fastest, which could be explained by that (1) S_1 is cheaper to compute than S_2 and (2) compared with A_2, A_1 needs to decide the quantity and choose the best supplier, which will increase computation load. For group two, we observe that RL4LS is not so efficient which could be explained that S_3 and A_3 are both cheaper to compute.

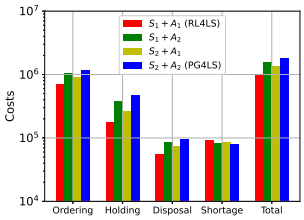
(4) Case Study (Real Dataset). We choose two typical materials. Material-I has a high unit shortage cost c_{s_1} but its disposal cost c_{d_1} is low while Material-II does not cost too much when shortage happens, i.e., c_{s_2} is much smaller, but discarding will incur a large punishment c_{d_2}. These two materials have the same fixed lifetime. The results are presented in Figure 4. The blue lines of these two figures represent the inventory change while the bars represent the order quantity on the corresponding state. For Material-I, maintaining a high inventory level is a good choice since low inventory level sometimes may encounter the problem of demand unmet. Thus, in Figure 4(a), we can see that the inventory level will almost maintain at a level over 80% of the maximum capacity. The policy for Material-II



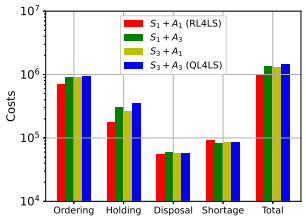
(a) Prediction on demands



(b) Prediction on lead time



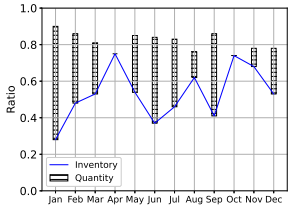
(c) Combination with S_2 and A_2



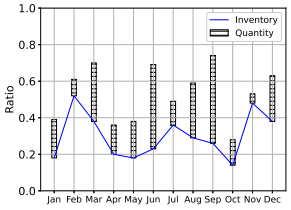
(d) Combination with S_3 and A_3

Fig. 3. Results on ablation study.

tells another story in Figure 4(b). Since discarding means a large punishment, the order quantity and the inventory level maintain at a low level.



(a) Case study for Material-I



(b) Case study for Material-II

Fig. 4. Results on real dataset (case study).

(5-7) Experiments on Synthetic Dataset. They can be found in the technical report [24].

6 Conclusion

In this paper, we study a novel problem, namely LS-PMU. We propose a RL-based algorithm `RL4LS`. Compared with existing algorithm, our algorithm can intelligently choose a supplier and decide an order quantity so as to minimize the overall costs. Extensive experiments on real and synthetic datasets demonstrate that `RL4LS` is effective and efficient. It is worthy of noting that we do not consider a fill rate (fr), i.e., the ratio of the satisfied demands to the total demands, in our model. If the inventory has such a goal, e.g., $fr \geq 95\%$, our solution would not guarantee to achieve it.

Acknowledgments. This research is supported by the Ministry of Education, Singapore, under its Academic Research Fund (Tier 1 Award (RG77/21)). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore. This work is also supported by the A*STAR Cyber-Physical Production System (CPPS) - Towards Contextual and Intelligent Response Research Program, under the RIE2020 IAF-PP Grant A19C1a0018, and Model Factory@SIMTech.

References

[1] H. M. Wagner and T. M. Whitin, "Dynamic version of the economic lot size model," *Management science*, vol. 5, no. 1, 1958.
[2] N. Brahimi, N. Absi, S. Dauzère-Pérès, and A. Nordli, "Single-item dynamic lot-sizing problems: An updated survey," *European Journal of Operational Research*, vol. 263, no. 3, pp. 838–863, 2017.
[3] M. Önal, "The two-level economic lot sizing problem with perishable items," *Operations Research Letters*, vol. 44, no. 3, pp. 403–408, 2016.
[4] A. Acevedo-Ojeda, I. Contreras, and M. Chen, "Two-level lot-sizing with raw-material perishability and deterioration," *Journal of the Operational Research Society*, vol. 71, no. 3, pp. 417–432, 2020.
[5] A. Kara and I. Dogan, "Reinforcement learning approaches for specifying ordering policies of perishable inventory systems," *Expert Systems with Applications*, vol. 91, pp. 150–158, 2018.
[6] N. Arslan, "Lot sizing with perishable items," Ph.D. dissertation, Bilkent Universitesi (Turkey), 2019.
[7] E. Ahmadi, D. T. Masel, S. Hostetler, R. Maihami, and I. Ghalehkhondabi, "A centralized stochastic inventory control model for perishable products considering age-dependent purchase price and lead time," *Top*, vol. 28, no. 1, pp. 231–269, 2020.
[8] A. K. Sinha and A. Anand, "Optimizing supply chain network for perishable products using improved bacteria foraging algorithm," *Applied Soft Computing*, vol. 86, p. 105921, 2020.
[9] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
[10] J. Xiong, Q. Wang, Z. Yang, P. Sun, L. Han, Y. Zheng, H. Fu, T. Zhang, J. Liu, and H. Liu, "Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space," *arXiv preprint arXiv:1810.06394*, 2018.
[11] I. Saracoglu, S. Topaloglu, and T. Keskinturk, "A genetic algorithm approach for multi-product multi-period continuous review inventory models," *Expert Systems with Applications*, vol. 41, no. 18, pp. 8189–8202, 2014.
[12] A. Akbalik, B. Penz, and C. Rapine, "Capacitated lot sizing problems with inventory bounds," *Annals of Operations Research*, vol. 229, no. 1, pp. 1–18, 2015.
[13] Z. Peng, Y. Zhang, Y. Feng, T. Zhang, Z. Wu, and H. Su, "Deep reinforcement learning approach for capacitated supply chain optimization under demand uncertainty," in *2019 Chinese Automation Congress (CAC)*. IEEE, 2019, pp. 3512–3517.
[14] L. Kemmer, H. von Kleist, D. de Rochebouët, N. Tziortziotis, and J. Read, "Reinforcement learning for supply chain optimization," in *European Workshop on Reinforcement Learning 14*, 2018, pp. 1–9.
[15] G. J. Van Zyl, "Inventory control for perishable commodities," North Carolina State University. Dept. of Statistics, Tech. Rep., 1963.
[16] Nahmias and Steven, "Higher-order approximations for the perishable-inventory problem," *Operations Research*, vol. 25, no. 4, pp. 630–640, 1977.
[17] E. Tekin, Ü. Gürler, and E. Berk, "Age-based vs. stock level control policies for a perishable inventory system," *European Journal of Operational Research*, vol. 134, no. 2, pp. 309–329, 2001.
[18] F. Olsson and P. Tydesjö, "Inventory problems with perishable items: Fixed lifetimes and backlogging," *European Journal of Operational Research*, vol. 202, no. 1, pp. 131–137, 2010.
[19] A. Barman and P. De, "A multi-item deteriorating inventory model under stock level-dependent, time-varying, and price-sensitive demand," in *Recent Trends in Applied Mathematics*. Springer, 2021, pp. 1–12.
[20] L. Feng, J. Zhang, and W. Tang, "Dynamic joint pricing and production policy for perishable products," *International Transactions in Operational Research*, vol. 25, no. 6, pp. 2031–2051, 2018.
[21] D. C. Nnadi, "A decision model for the design and operation of inventory programmes in a manufacturing industry," 2017.
[22] F. Jing and X. Chao, "A dynamic lot size model with perishable inventory and stockout," *Omega*, vol. 103, p. 102421, 2021.
[23] A. Oroojlooyjadid, M. Nazari, L. Snyder, and M. Takáč, "A deep q-network for the beer game: A deep reinforcement learning algorithm to solve inventory optimization problems," *arXiv preprint arXiv:1708.05924*, 2017.
[24] https://wangkaixin219.github.io/usage/inventoryTR.pdf.
[25] G. E. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.
[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[27] H. Soleimani, M. Seyyed-Esfahani, and M. A. Shirazi, "Designing and planning a multi-echelon multi-period multi-product closed-loop supply chain utilizing genetic algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 68, no. 1-4, pp. 917–931, 2013.