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 for the lot sizing problem [2]. Most of existing works [3], [4], [5] for the lot sizing problem assume a single-supplier scenario, where there is only one supplier available across different periods. More recent works [6], [7] target a multiple-supplier scenario. Specifically, [6], [7] 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 [6], [7].

Consider a planning scenario of T periods with a set of suppliers, denoted by S. Existing solutions usually adopt a vector-based solution, which represents the ordering policy as a $|S|$-dimensional vector in each planning period [6], [7], but they apply different strategies to find the ordering policy with the minimum total costs. One state-of-the-art method is [6], called GA. This method applies the genetic algorithm to find the optimal solution, where a possible solution is encoded as a chromosome-like data structure, represented by a $(|S| \times T)$-dimensional vector. Another method is proposed in [7], called IBFA. This method applies an improved bacteria foraging algorithm to find the best solution, where the raw solutions, intermediate results and the global optimum are all demonstrated by $(|S| \times T)$-dimensional vectors. These existing solutions, however, cannot handle our task well since we do not allow to place orders to multiple suppliers

- *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.*

TABLE 1
Related literature of lot-sizing problems.

Reference	Material	Setting	Deterioration	Supplier	Method[1]
[8], [9], [10], [11], [12], [13], [14], [15], [16]	Non-perishable	*[2]	N.A.	*	*
[4], [17], [18], [19], [20], [21], [22], [23], [24]	Perishable	Offline	*	*	DP / Heuristics
[25], [26], [27], [28]	Perishable	Online	Continuous	*	Simulation / Heuristics
[5], [29], [30], [31]	Perishable	Online	Fixed Lifetime	Single	SP / Heuristics / RL
[6], [7]	Perishable	Online	Fixed Lifetime	Multiple	Nature-inspired Heuristics
Ours	Perishable	Online	Fixed Lifetime	Multiple[3]	RL

[1] DP represents dynamic programming; SP represents stochastic programming; RL represents reinforcement learning.
[2] We use * to denote that all types under the criteria (column) are covered by some of the references of the row.
[3] Our problem targets a different multiple-supplier setting compared with [6], [7].

in a single period. In addition, [6], [7] are both based on the nature-inspired heuristics, and it would take a long time for them to converge.

To address the LS-PMU problem effectively and efficiently 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) [32]. We carefully design the MDP including states, actions and rewards such that (1) the states capture the critical information of the inventory status and cheap to compute; (2) the actions are *hybrid* ones and 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 an existing reinforcement learning algorithm, namely parameterized Deep-Q Network (P-DQN) [33], to learn the policy for the MDP. In summary, the main contributions of this paper are as follows.

- We propose the LS-PMU problem, in which we place orders to at most one supplier in each period, which is more aligned with real logistics applications.
- We present theoretical results on the lower bound of the competitive ratio of any given algorithm for LS-PMU problem, which could reflects the best possible quality guarantee for any online algorithm to the LS-PMU problem.
- We propose a reinforcement learning based algorithm RL4LS for the LS-PMU problem. Compared with existing methods to the lot sizing problem, 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.

The rest of this paper is organized as follows. We provide the problem definition and theoretical results in Section 3. We present our RL4LS algorithm in Section 4 and show the experimental results in Section 5. We summarize the related work in Section 2. Finally, we conclude our paper in Section 6.

2 Related Work

Lot-sizing Problems. Lot sizing problems can be classified based on several criteria as presented in Table 1. Among them, [8], [9], [10], [11], [12], [13], [14], [15], [16] 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 [4], [17], [18], [19], [20], [21], [22], [23], [24]. Besides, most of them assume that the lead time is always zero. Recently, there exist some studies [5], [25], [26], [27], [28], [29], [30], [31] which consider the online setting. However, [25], [26], [27], [28] 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. [5], [29], [30], [31] target the materials that have a fixed expiry date, but they assume a *single* supplier. The most related studies are [6], [7] since they also target perishable materials 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. Given a specific environment, which is generally formulated as a Markov Decision Process (MDP) [32], reinforcement learning is to help the agents in the environment learn how to map the states to actions so as to maximise the accumulative rewards [34]. Some existing studies [5], [15], [16], [35] have tried to solve the inventory planning problem using reinforcement learning, they differ from our RL4LS algorithm in various aspects: (1) [15], [16] allow to place orders to more than one suppliers in a single period, and they focus on non-perishable materials; (2) [5] assumes zero or constant lead time; (3) [35] targets a multiple-echelon scenario that multiple inventories cooperate to minimize the total costs or maximize the revenue.

3 Problem Statement

3.1 Variables

We consider a horizon of time periods $1, 2, ..., T$, where each corresponds to a unit of time such as a day, a week, or a month. Let S be a set of suppliers, from which we can order materials. During each time period, the material of some quantity may be involved in different events, e.g., it is ordered from some supplier, delivered to the inventory, sold to customers, and disposed. We define the following variables to represent the quantities involved in different 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 time period $i \in [1, T]$, respectively. Here, the lead time $l(i, j)$ means the amount of time it takes for the

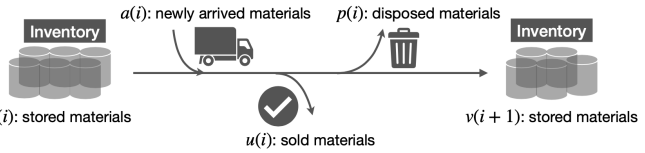


Fig. 1. An illustration of the variables and events between two adjacent time periods.

ordered material to be transported from the supplier j to the inventory starting at the beginning of time period i. $l(i,j)$ usually depends on the various transportation factors such as weather and is unknown for future time periods. Note that the order $o(i,j)$ will be received at the beginning of the time period $i' = i + l(i,j)$.

We denote by $v(i)$ the quantity of the material stored in the inventory at the beginning of time 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 time 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 that 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 time period i. We assume that no material expires before it arrives at the inventory, which is well aligned with the practice. Thus, this quantity depends on how the material is ordered and transported before time 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 by customers during time period i. This quantity for the future time periods is unknown. We further denote by $s(i)$ the quantity of the material that is in shortage during time period i. This quantity is equal to $\{d(i) - v(i) - a(i)\}^+$, where $\{x\}^+ = \max\{0, x\}$. Here, we target the lost-sales scenario, where the customers would drop the unfilled demands incurring some shortage cost, which will be introduced in the next subsection.

Besides, we denote by $u(i)$ the quantity of the material that is used to serve customers' demands during the time period i. Note that we have $u(i) = \min\{d(i), v(i) + a(i)\}$.

Finally, we denote by $p(i)$ the quantity of the material that is disposed at the end of time period i. In this case, this quantity depends on the quantity of material that will expire at the end of this time period, i.e., $v(i,1) + a(i,1)$, and the customer demand during this time period, i.e., $d(i)$. Specifically, we have $p(i) = \{v(i,1) + a(i,1) - d(i)\}^+$. Here, we adopt the principle that material with shorter remaining life time is used to serve customers' demands first (i.e., First-Expired-First-Out (FEFO)).

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.

3.2 Costs

Due the life time of the material, various costs are incurred, including ordering cost, holding cost, disposal cost and shortage cost for each time 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 cost that would be incurred when a certain quantity of material is ordered from the supplier j, i.e., the fixed setup costs. Then, the costs can be computed as follows.

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

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 only. Mathematically, the problem could be formalized as follows.

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

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

$$\textstyle\sum_{j \in S} I(o(i,j)) \leq 1 \text{ for each } i \in [1,T] \tag{6}$$

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 $\min_{j \in S} c_o(j) \leq c_s$ (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*

$$o(i) = \sum_{j \in S} o(i,j) \tag{7}$$

and

$$cr(i) = \max\left(\frac{(c_h + c_d)\cdot o(i) + f(o(i))}{f(0)}, \frac{c_s}{\min_{j\in S}\{c_o(j)\}}\right), \tag{8}$$

where

$$f(x) = \min_{j\in S}\{c_o(j)\cdot x + c_b(j)\}, \quad x \geq 0 \tag{9}$$

then the competitive ratio of algorithm A is at least $\min_i cr(i)$.

We provide the proof of Theorem 1 and also an offline optimal algorithm A^* in the technical report [36]. With A^*, we can compute the empirical competitive ratio of an algorithm. Theorem 1 provides the bounds under the worst-case, but in practice, we observe that the ratio of the total costs returned by our algorithm to the total costs returned by A^* is only around 2.1 on our real datasets.

4 METHODOLOGY

We observe that the inventory 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 reinforcement learning to help with the decision making process. Specifically, we model the LS-PMU problem as a Markov Decision Process (MDP), adopt an existing deep reinforcement learning method, namely P-DQN [33], for learning an optimal policy on the MDP, and then develop an algorithm called RL4LS, which uses the learned policy for solving 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 three components, namely states, actions, and rewards as defined as follows.

States. We denote the state in time period i by s_i. Intuitively, the state s_i should capture essential information of the inventory status for making decisions of choosing a supplier and setting the order quantity in a period. We identify the following three types of information at the beginning of the time period i, which are critical for making the decisions: (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 time period i, we take the values $v(i,r)$ for $r \in [1, L]$ and concatenate them together, denoted by s_i^v, that is,

$$s_i^v = [v(i,1), \cdots, v(i,r), \cdots v(i,L)] \tag{10}$$

The rationale is that s_i^v provides a clear 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, we use s_i^o, as defined below, to capture the open order information,

$$s_i^o = [b(i-L+1)\cdot o(i-L+1), \cdots, b(i-1)\cdot o(i-1)] \tag{11}$$

where $o(\cdot) = \sum_{j\in S} o(\cdot, j)$. The rationale is that s_i^o clearly 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 $i - r \leq 0$, we assume the open orders are equal to 0.

Finally, to capture the third type of information of the inventory status, we make some predictions on the lead time and the demands. Firstly, the supply lead time is stochastic and usually varies among different suppliers. Moreover, we may choose different suppliers in different periods. Therefore, 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. Formally, based on the ordering history of different suppliers, we predict the lead times for different suppliers, which are denoted by $\hat{l}(i,j)$ for $j \in S$. Then, we concatenate the predicted lead times for different suppliers and denote it by s_i^l,

$$s_i^l = [\hat{l}(i,1), \cdots, \hat{l}(i,j), \cdots \hat{l}(i,|S|)], \quad j \in S \tag{12}$$

Similarly, demand prediction provides the information of how many materials we need to meet in the following time 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 demand, denoted by $\hat{d}(i)$, and the corresponding state representation s_i^d are defined as follows.

$$s_i^d = [\hat{d}(i)] \tag{13}$$

We try different methods for lead time and demand prediction, e.g., ARIMA [37], LSTM [38] and sampling from known distribution. We compare the results and report them in the experiments. In summary, we define state s_i to be a $(2L + |S|)$-dimension vector, which captures the inventory level information, open order information and the future predictions, i.e.,

$$s_i = [s_i^v, s_i^o, s_i^l, s_i^d] \in \mathbb{R}^{2L+|S|} \tag{14}$$

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) how much quantity we order and (2) which supplier we choose from. Formally, we define a_i as follows.

$$a_i = [j, q] \quad (j \in S) \tag{15}$$

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

associated with this transition from s_i to s_{i+1}, denoted by R_i, as follows.

$$R_i = -C(i) \tag{16}$$

The intuition is that if a smaller cost is incurred in period i, the action a_i would be associated with a larger reward, and vice versa. 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 $s_1, s_2, \cdots, s_T$ and correspondingly we receive a sequence of rewards R_1, R_2, R_{T-1}. Then, the accumulative rewards without being discounted can be computed by

$$\sum_{i=1}^{T} R_i = -\sum_{i=1}^{T} C(i). \tag{17}$$

4.2 Policy Learning on the MDP

Considering that our MDP has a high dimensional continuous state space and a hybrid action space, we adopt P-DQN [33], which is capable of dealing with continuous states and hybrid actions, to learn a policy from our MDP. 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.,

$$\begin{aligned} j^* &= \arg\max_{j \in S} Q(s_i, j, \mu_j(s_i; \theta); \omega) \\ q^* &= \mu_{j^*}(s_i; \theta), \end{aligned} \tag{18}$$

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 the action $a = [j^*, q^*]$ with a probability of $(1 - \epsilon)$ and a random action $a = [j, q]$ other than $[j^*, q^*]$ with a probability of ϵ, to balance exploration and exploitation. We also maintain a replay memory, which contains the latest several transitions that are used for training the networks. Then, the training process is as follows. Consider N experiences sampled uniformly from the replay memory, i.e., (s_i, a_i, R_i, s_{i+1}) for $i = 1, 2, \cdots N$. For the Q network, we compute the loss by

$$L(\omega) = \frac{1}{N} \sum_{i=1}^{N} \Big(y_i - Q(s_i, a_i; \omega) \Big)^2 \tag{19}$$

where

$$y_i = R_i + \gamma \cdot \max_{j' \in S} Q'(s_{i+1}, j', \mu'_{j'}(s_{i+1}; \theta'); \omega'). \tag{20}$$

For the policy network μ, we compute the loss by

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{|S|} Q(s_i, j, \mu_j(s_i; \theta); \omega). \tag{21}$$

Algorithm 1 The RL4LS algorithm

Require: The demands $d(i)$ and the lead time $l(i, j)$ for all i and j, where they arrive in an online fashion;
Ensure: The ordering plan $o(i, j)$ for all i, j, total costs C;
1: Initialize the open order list $O \leftarrow \phi$;
2: $C \leftarrow 0$;
3: **for** each time period i where $i \in [1, T]$ **do**
4: $s_i \leftarrow$ observe the inventory status as Eq. (14);
5: Take an action $a_i = [j^*, q^*]$ which satisfies Eq. (18);
6: Output $o(i, j^*) = q^*$ and $o(i, j) = 0$ for $j \neq j^*$;
7: $C(i) \leftarrow$ total costs in the time period i;
8: $C \leftarrow C + C(i)$;
9: **end for**
10: **return** C

Finally, we update the parameters θ and ω by gradient descent.

4.3 The RL4LS Algorithm

The RL4LS algorithm, presented in Algorithm 1, is based on the learned policy for the MDP that models the LS-PMU problem. Specifically, it proceeds periodically (line 3). At each time period i, it observes the inventory status based on the three types of information, and constructs a state s_i (line 4). Then, it takes an action $a_i = [j^*, o^*]$ based on the learned networks $Q(s, a; \omega)$ and $\mu(s; \theta)$ (line 5-6). Then, the ordering plan for the time period i will be (1) $o(i, j^*) = q^*$ and (2) $o(i, j) = 0$ for other supplier j. After that, we observe the costs and move to the next period (lines 7-8).

Time complexity. The time complexity of the RL4LS algorithm is $O(T(L + |S|))$, where T denotes the maximum planning horizon and L is the maximum life time. The cost mainly comes from the transition 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 [39]. 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 $T = 12$. The dataset contains some private commercial information, thus, we omit the detailed descriptions of this dataset in this paper. We also generate some synthetic datasets by following the existing study [40], 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 [40].

Baselines and Metrics. We compare our proposed model with four existing algorithms, namely IBFA [7], GA [6], PG4LS [15] and QL4LS [5], in terms of the total cost

TABLE 2
Running time (seconds) on real dataset.

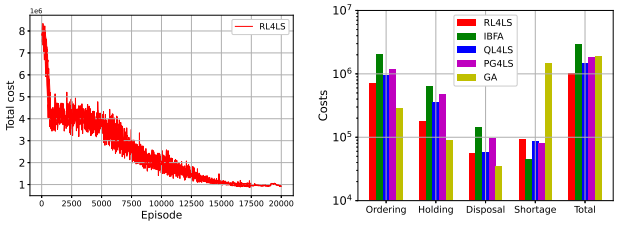
Alg.	RL4LS	IBFA	QL4LS	PG4LS	GA
Time	2.39	542.27	1.78	2.77	843.74

and the running time. IBFA and GA are two state-of-the-art algorithms for the lot-sizing problem with perishable materials under the multiple-supplier setting, which use two different nature-inspired heuristics. PG4LS is the most recent reinforcement learning based algorithm for non-perishable inventory planning under the multiple-supplier setting. These three algorithms are all vector-based solutions, i.e., they allow to place orders to more than one supplier in a single period. We adapt them to our problem by the following steps. Suppose the output for a period i can be represented by a vector $[p_1, p_2, \cdots, p_{|S|}]$. First, we set the order quantity to $q_i = \sum_{j=1}^{|S|} p_j$. Second, we choose the supplier which has the minimum ordering cost, i.e., $j_i = \min_{j \in S}\{c_o(j) \cdot q_i + c_b(j)\}$, so as to align with the objective of minimizing the total cost. QL4LS is the state-of-the-art reinforcement learning based algorithm for the perishable lot-sizing problem under the single-supplier setting. In QL4LS, a state is defined as a pair of two real numbers, one equal to the inventory level of the materials and the other equal to the sum of the remaining life times of materials in the inventory, and an action is a real number representing the order quantity, say q_i. To adapt QL4LS to our problem, we choose the supplier which generates the least ordering costs, i.e., $j_i = \min_{j \in S}\{c_o(j) \cdot q_i + c_b(j)\}$ (s.t. the problem becomes one under the single-supplier scenario), apply the QL4LS algorithm based on the chosen supplier, and set the order quantity from the chosen supplier to q_i in each time period.

Hyperparameter Selection and Model Training. For the baseline algorithms, all hyperparameters are set to be the same as those in existing studies. For all algorithms, we regard each material as an independent agent for each problem instance of size $|M| \times |S|$. For the dataset, which consists of data spanning over 6 years, from 2015 to 2020, we use the first five years for training and the data of the last year for testing. The network $\mu(s;\theta)$ is composed of one hidden layer and one output layer. We use the tanh function as the activation function with 64 neurons in the hidden layer, and we use sigmoid function with $|S|$ neurons in the output layer corresponding to different actions. The network $Q(s, a; \omega)$ consists of 2 layers. In the first layer, we use the tanh function as the activation function with 64 neurons, and in the second layer, we use a linear function with $|S|$ neurons as the output corresponding to different actions. For training, we set the discount factor γ to 1, and use the Adam optimiser with constant learning rate 10^{-4}. We also adopt $\epsilon = 0.1$ for the ϵ-greedy process involved in training process. Other parameters follow the default settings in PyTorch. We train the networks for 20,000 iterations. The hardware we use is a machine with Intel Core i9-10940X CPU and a single Nvidia GeForce 2080Ti GPU. Our codes can be found via https://github.com/wangkaixin219/RL4LS.

5.2 Experiment Results

(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).



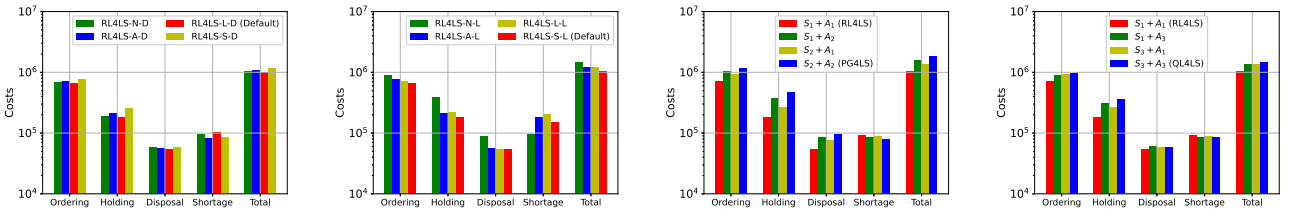
(a) Learning Curve (b) Cost Decomposition

Fig. 2. Results on real dataset.

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 2). We observe that 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 both the lead time and the demands in our state definition. We choose three models, namely ARIMA [37], LSTM [38] and Gaussian Sampling. For ARIMA(p, d, q) prediction method, we set $p = 1, d = 0, q = 1$ for both the demand and the lead time predictions. For the demand prediction (resp. lead time prediction of a supplier) using LSTM, the input of LSTM 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 are both following some Gaussian distributions. We denote these variants by RL4LS-$N/A/L/S$-D/L, where N represents we do not use prediction, A (L or S) represents the prediction model is using ARIMA (LSTM or Sampling), 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, using LSTM performs the best among these four variants, which could be explained by the fact that using LSTM can make a more accurate prediction than other methods, and this information could be used to decide the actions with the least costs. For lead time prediction, however, using sampling method 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 factors that unpredictable, such as weather or traffic, which makes the prediction unreliable. Consider the running time (Table 3). We observe that all algorithms are efficient (i.e. the running time $<$ 3s).

(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,



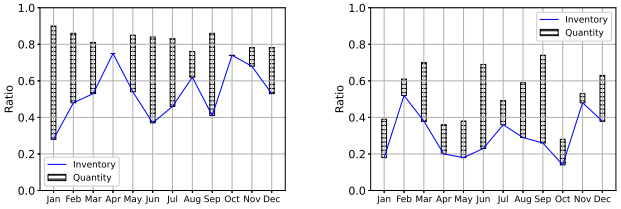
(a) Prediction on demands (b) Prediction on lead time (c) Combination with S_2 and A_2 (d) Combination with S_3 and A_3

Fig. 3. Results on ablation study.

TABLE 3
Running time (seconds) on ablation study (demand and lead time).

Alg.	RL4LS-N-D	RL4LS-A-D	RL4LS-L-D	RL4LS-S-D
Time	2.11	2.26	2.39	2.18
Alg.	RL4LS-N-L	RL4LS-A-L	RL4LS-L-L	RL4LS-S-L
Time	1.98	2.57	2.74	2.39

namely those adopted in PG4LS [15] 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 (1) $S_1 + A_1$ (i.e., RL4LS), (2) $S_1 + A_2$, (3) $S_2 + A_1$, and (4) $S_2 + A_2$ (i.e., PG4LS). In group two, we explore (1) $S_1 + A_1$ (i.e., RL4LS), (2) $S_1 + A_3$, (3) $S_3 + A_1$, and (4) $S_3 + A_3$ (i.e., QL4LS). 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 (Table 4). 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 than S_1 and A_1.



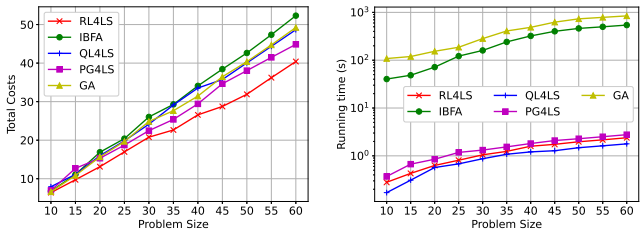
(a) Case study for Material-I (b) Case study for Material-II

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

(4) Case Study (Real Dataset). We choose two typical materials. Material-I has an extremely 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 than c_{s_1}, but discarding will result in 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 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. From the result we also observe another interesting phenomenon. From July to October, the inventory level decreases gradually. Instead of ordering a large amount of materials at October when the inventory level is the lowest, it orders the least at that time period. The explanation is that we have ordered those materials in August and September in advance. Since we have encoded the information of in-transit material in our state vector, the algorithm will know that it does not have to order again.

TABLE 4
Running time (seconds) on ablation study (state and action).

Alg.	$S_1 + A_1$	$S_1 + A_2$	$S_2 + A_1$	$S_2 + A_2$
Time	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	2.39	2.26	1.95	1.78



(a) Total costs ($\times 10^7$) (b) Running time

Fig. 5. Results on synthetic dataset.

(5) Results on Synthetic Dataset. The results of the synthetic datasets are present in Figure 5. Since the demands and lead time generated in [40] follow a uniform distribution, we use the same distribution to sample the demand and lead time for the predictions in the state definition. Consider the effectiveness, our proposed algorithm outperforms other algorithms under most cases. GA has the second best results when the problem size is small (i.e., problem size $< 20 \times 20$) since GA could generate a large search space due to the chromosome structure, and it is easy to extract a good solution. However, when the problem size increases, it becomes harder to search the whole solution space, and as a result, the performance becomes worse than the RL-based algorithms. Compared with other RL-based algorithms, our algorithm is also competitive since the our proposed state definition captures more information than that of QL4LS and PG4LS. As for the running time, GA and IBFA need a large amount of time by trial-and-error while RL-based algorithms are efficient as the problem size becomes larger. QL4LS runs the fastest, which could be explained by the fact that the state is cheaper to compute.

(6-7) Results on other parameter studies (zero lead time, constant demand and varying the maximum time period). They can be found in the technical report [36].

6 CONCLUSION

In this paper, we study a novel problem, namely LS-PMU, where we aim to minimise the total costs for perishable materials under various uncertainties. We propose a reinforcement learning based algorithm, i.e., RL4LS. Compared with existing algorithm, our algorithm can intelligently choose a supplier and decide an order quantity so as to minimise the overall costs. Extensive experiments on real and synthetic datasets demonstrate that our algorithm is effective and efficient. It is worthy of noting that our proposed solution is not suitable for the materials whose associated costs vary significantly in different time periods since those costs are considered consistent in our problem. Also, since 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.

REFERENCES

[1] H. M. Wagner and T. M. Whitin, "Dynamic version of the economic lot size model," *Management science*, vol. 5, no. 1, pp. 89–96, 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] E. Berk and Ü. Gürler, "Analysis of the (q, r) inventory model for perishables with positive lead times and lost sales," *Operations Research*, vol. 56, no. 5, pp. 1238–1246, 2008.

[4] 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.

[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] 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.

[7] 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.

[8] Y. Pochet and L. A. Wolsey, *Production planning by mixed integer programming*. Springer, 2006, vol. 149, no. 2.

[9] N. Brahimi, S. Dauzere-Peres, N. M. Najid, and A. Nordli, "Single item lot sizing problems," *European Journal of Operational Research*, vol. 168, no. 1, pp. 1–16, 2006.

[10] N. Absi, S. Kedad-Sidhoum, and S. Dauzère-Pérès, "Uncapacitated lot-sizing problem with production time windows, early productions, backlogs and lost sales," *International Journal of Production Research*, vol. 49, no. 9, pp. 2551–2566, 2011.

[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] S. H. R. Pasandideh, S. T. A. Niaki, and S. M. Mousavi, "Two metaheuristics to solve a multi-item multiperiod inventory control problem under storage constraint and discounts," *The International Journal of Advanced Manufacturing Technology*, vol. 69, no. 5-8, pp. 1671–1684, 2013.

[14] I. Giannoccaro and P. Pontrandolfo, "Inventory management in supply chains: a reinforcement learning approach," *International Journal of Production Economics*, vol. 78, no. 2, pp. 153–161, 2002.

[15] 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.

[16] 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.

[17] G. J. Van Zyl, "Inventory control for perishable commodities," North Carolina State University. Dept. of Statistics, Tech. Rep., 1963.

[18] S. Nahmias, "The fixed-charge perishable inventory problem," *Operations Research*, vol. 26, no. 3, pp. 464–481, 1978.

[19] Nahmias and Steven, "Higher-order approximations for the perishable-inventory problem," *Operations Research*, vol. 25, no. 4, pp. 630–640, 1977.

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

[21] H. N. Chiu, "A heuristic (r, t) periodic review perishable inventory model with lead times," *International Journal of Production Economics*, vol. 42, no. 1, pp. 1–15, 1995.

[22] Chiu and H. Neng, "An approximation to the continuous review inventory model with perishable items and lead times," *European Journal of Operational Research*, vol. 87, no. 1, pp. 93–108, 1995.

[23] ——, "A good approximation of the inventory level in a (q, r) perishable inventory system," *RAIRO-Operations Research*, vol. 33, no. 1, pp. 29–45, 1999.

[24] 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.

[25] 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.

[26] 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.

[27] D. C. Nnadi, "A decision model for the design and operation of inventory programmes in a manufacturing industry," 2017.

[28] F. Jing and X. Chao, "A dynamic lot size model with perishable inventory and stockout," *Omega*, vol. 103, p. 102421, 2021.

[29] M. Önal, "The two-level economic lot sizing problem with perishable items," *Operations Research Letters*, vol. 44, no. 3, pp. 403–408, 2016.

[30] 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.

[31] N. Arslan, "Lot sizing with perishable items," Ph.D. dissertation, Bilkent Universitesi (Turkey), 2019.

[32] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[33] 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.

[34] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[35] 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.

[36] K. Wang, C. Long, D. J. Ong, J. Zhang, and X. Yuan, "Single-site perishable inventory management under uncertainties: A deep reinforcement learning approach (technical report)," https://wangkaixin219.github.io/usage/inventoryTR.pdf.

[37] 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.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] Alcon. https://www.alcon.com/.

[40] 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.

[41] A. Atamtürk and S. Küçükyavuz, "An $o(n^2)$ algorithm for lot sizing with inventory bounds and fixed costs," *Operations Research Letters*, vol. 36, no. 3, pp. 297–299, 2008.

[42] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

APPENDIX A
PROOF OF THEOREM 1

Proof. We prove by constructing an instance with a material with the maximum life time equal to one, i.e., $L = 1$, and $|S|$ suppliers over T periods. Since the maximum life time of this material is only one, the materials that are not used to meet the demands will be disposed at the end of each period. Thus, the inventory level at the beginning of each period will be zero, i.e., $v(i) = 0$ for all $i \in [1, T]$. Suppose that the order arrives instantaneously, i.e., $l(i, j) = 0$ for all i and j. Now we first consider the costs for a single period. Then, we extend the results to multiple periods. For a period i, there are two possible cases: (1) $d(i) \leq o(i)$, and (2) $d(i) > o(i)$.

Case 1 $d(i) \leq o(i)$. Since the ordering quantity is larger than the demand, we need to pay the holding costs, disposal costs and ordering costs in this time period. Define

$$g(x) = \min_{j \in S}\{c_o(j) \cdot x + c_b(j) \cdot I(x)\}, \quad x \geq 0. \tag{22}$$

Then, the costs for time period i would be,

$$\begin{aligned} C(i) &= C_h(i) + C_o(i) + C_d(i) \\ &\geq (c_h + c_d) \cdot (o(i) - d(i)) + g(o(i)) \\ &= (c_h + c_d) \cdot o(i) + g(o(i)) - (c_h + c_d) \cdot d(i). \end{aligned} \tag{23}$$

Since the order arrives instantaneously, the offline optimal solution would be $o(i) = d(i)$, and we choose the supplier that provides the least ordering costs with respect to $o(i)$. Thus, the objective is

$$C^*(i) = g(d(i)). \tag{24}$$

Thus, the competitive ratio can be computed by

$$\begin{aligned} cr(i) &= \max_{x \in \mathcal{X}} \frac{obj(A(x))}{obj(A^*(x))} \\ &\geq \max_{d(i) \geq 0} \frac{(c_h + c_d) \cdot o(i) - (c_h + c_d) \cdot d(i) + g(o(i))}{g(d(i))} \\ &= \lim_{x \to 0^+} \frac{(c_h + c_d) \cdot o(i) - (c_h + c_d) \cdot x + g(o(i))}{g(x)} \\ &= \frac{(c_h + c_d) \cdot o(i) + f(o(i))}{f(0)} \end{aligned} \tag{25}$$

Case 2 $o(i) < d(i)$. Since the ordering quantity is less than the demand, the costs are composed by the ordering and shortage costs,

$$\begin{aligned} C(i) &= C_o(i) + C_s(i) \\ &\geq g(o(i)) + c_s \cdot (d(i) - o(i)) \\ &= \min_{j \in S}\{(c_o(j) - c_s) \cdot o(i) + c_b(j) \cdot I(o(i))\} + c_s \cdot d(i), \end{aligned} \tag{26}$$

If $c_o(j) > c_s$ for all j, then $C(i)$ has the minimum when $o(i) = 0$. That means that if all unit ordering costs provided by different suppliers are larger than the shortage cost, which are known before running the algorithm, then the ordering policy for any algorithm which induces the minimum costs will be $o(i, j) = 0$ for all supplier j, which is the same as the offline optimal solution. Thus, the competitive ratio would be exactly 1.

Another scenario is that the supplier s, which provides the least $C(i)$, satisfies that $c_o(s) \leq c_s$, then the offline optimal solution would be $o(i, s) = d(i)$, and $o(i, j) = 0$ for all suppliers $j \neq s$. Then, the objective is

$$C^*(i) = g(d(i)). \tag{27}$$

Thus, the competitive ratio can be computed by

$$\begin{aligned} cr(i) &= \max_{x \in \mathcal{X}} \frac{obj(A(x))}{obj(A^*(x))} \\ &\geq \max_{d(i) \geq 0} \frac{g(o(i)) + c_s \cdot (d(i) - o(i))}{g(d(i))} \\ &= \lim_{x \to +\infty} \frac{g(o(i)) + c_s \cdot (x - o(i))}{g(x)} \\ &= \frac{c_s}{\min\limits_{j \in S}\{c_o(j)\}} \end{aligned} \tag{28}$$

Since the demand $d(i)$ is not known in advance, thus the competitive ratio for a single period is

$$cr(i) \geq \max\left(\frac{(c_h + c_d) \cdot o(i) + f(o(i))}{f(0)}, \frac{c_s}{\min\limits_{j \in S}\{c_o(j)\}}\right) \tag{29}$$

Then, for a planning task with T horizon, since each period is similar, then the competitive ratio will be

$$\begin{aligned} cr(A) &= \max_{x \in \mathcal{X}} \frac{obj(A(x))}{obj(A^*(x))} \\ &= \max_{\boldsymbol{d} \geq \boldsymbol{0}} \frac{\sum_{i=1}^{T} C(i)}{\sum_{i=1}^{T} C^*(i)} \\ &= \frac{\sum_{i=1}^{T} cr(i) \cdot C^*(i)}{\sum_{i=1}^{T} C^*(i)} \\ &\geq \min_i cr(i). \end{aligned} \tag{30}$$

Therefore, we conclude that the algorithm A cannot achieve a competitive ratio better than $\min_i cr(i)$. □

APPENDIX B
OFFLINE OPTIMAL ALGORITHM

To evaluate an online algorithm A, it is important to know the gap between the result returned by A and the result returned by the optimal solution. Thus, we present an optimal solution in which the demands and the lead time are known in advance.

Lemma 1. *$C_d(i) = 0$ for all $i \in [1, T]$.*

Proof. We prove it by contradiction. Assume that there is an optimal solution S^* returned by A^* in which there exists a period i such that $C_d(i) > 0$. We denote the quantity of the materials being disposed in period i by $m > 0$. Since these materials reaches their lifetime in period i, then they must be ordered in the period $i' = i - (L - 1)$ and there exists a supplier j satisfying $o(i', j) \geq m$. We propose another solution S. For all periods except period i', the ordering policy remains the same as S^* and for the period i', we order $o(i', j) = o^*(i', j) - m$ from the same supplier j. Since the ordering quantity in period i' becomes less and the inventory does not need to hold these m materials between the period $i' + l(i', j)$ and period i, the costs incurred by solution S will be less than that of S^*, which is contradicted by the assumption that S^* is the optimal solution. □

Since there is no costs coming from disposing in the offline optimal solution, it reduces to general lot-sizing problem with positive lead time. To solve the problem, we first review some optimal conditions of order quantity under the zero-lead-time setting, and then we choose the suitable supplier by (1) satisfying the optimal conditions of the order quantity and (2) considering the lead time constraints.

Definition 1 (Block [41]). *Let $v(T+1) = 0$. A block, denoted by $[j, k]$ $(j \leq k)$, represents a consecutive sequence of time periods $[j, k]$ if $v(j) \in \{0, V\}$, $v(k+1) \in \{0, V\}$ and $0 < v(i) < V$ for all period $i \in [j+1, k]$ if any.*

We extend the optimal conditions summarized in [41] to our problems. The whole planning horizon under the optimal solution can be divided into several blocks. Within each block, there is at most one period with positive quantity of the receiving materials. There are four types of the block, determined by the values of inventory levels $v(j)$ and $v(k)$. There are two useful observations of any block $[j, k]$.

- If $v(j) = 0$ and the quantity of the receiving materials $a(i) > 0$ for some $i \in [j, k]$, then $i = j$.
- If $v(k+1) = V$ and the quantity of the receiving materials $a(i) > 0$ for some $i \in [j, k]$, then $i = k$;

We introduce some new notations to solve the offline optimal LP-PMU problem.

Definition 2 (Order tuple). *An order tuple, denoted by (i, j, q), represents we order a quantity of q materials in period i from supplier j.*

Definition 3 (Partial solution). *A feasible partial solution for a horizon of t periods, denoted by $fp^{(t)} = \{(i_1, j_1, q_1), \cdots, (i_k, j_k, q_k)\}$, contains a list of ordering tuples. The sequence of these tuples are in ascending order of the receiving time, i.e., $i_1 + l(i_1, j_1) < i_2 + l(i_2, j_2) < \cdots < i_k + l(i_k, j_k)$. A feasible partial solution satisfies $v(t+1) = 0$.*

Now we develop the offline optimal algorithm. The algorithm considers the demands period by period. Assume that we have already had a feasible partial solution $fp^{(i-1)} = \{(i_1, j_1, q_1), \cdots, (i_k, j_k, q_k)\}$. Now we consider the demand $d(i)$ of period i. There are three possible cases.

Case 1. We do not meet these demands and they are having a shortage. Thus, the feasible partial solution $fp^{(i)}$ remains the same as $fp^{(i-1)}$, and it satisfies that $v(i+1) = 0$. The additional costs will be

$$w^{(i)} = c_s \cdot d(i). \tag{31}$$

Case 2. The demand $d(i)$ is met by an order other than those in $fp^{(i-1)}$. Specifically, we denote the order which meets $d(i)$ by $(i_{k+1}, j_{k+1}, d(i))$, where it should satisfy (1) ordering constraint $i_{k+1} \notin \{i_1, \cdots, i_k\}$ and (2) lead time constraint $i_{k+1} + l(i_{k+1}, j_{k+1}) = i$. Then the feasible partial solution after covering the period i will be $fp^{(i)} = \{(i_1, j_1, q_1), \cdots, (i_k, j_k, q_k), (i_{k+1}, j_{k+1}, d(i))\}$. Since the receiving materials just meet the demand, this feasible partial solution also satisfies $v(i+1) = 0$. Thus, the additional costs will be

$$w^{(i)} = c_o(j_{k+1}) \cdot d(i) + c_b(j_{k+1}). \tag{32}$$

Note that there might be several order tuples satisfying the above two constraints, and each tuple will generate a new feasible partial solution.

Case 3. The demand $d(i)$ is met by the last order (i_k, j_k, q_k) in $fp^{(i)}$. This case should first satisfy the lifetime constraint $i - i_k < L$. To further make $fp^{(i)}$ a feasible partial solution, we need to order additional $d(i)$ materials when we place the order in period i_k so that $v(i+1) = 0$. Therefore, the last order would be $(i_k, j_k, q_k + d(i))$. However, sometimes we cannot order $q_k + d(i)$ materials because of the inventory capacity V.

Consider the period $i' = i_k + l(i_k, j_k)$. The inventory level at the beginning of the period i' is $v(i')$, and the demand in that period is $d(i')$. Thus, the most order quantity in period i_k would be

$$Q_k = a(i') \leq V + d(i') - v(i'). \tag{33}$$

Case 3.1. $q_k + d(i) \leq Q_k$. Under this case, we can directly order $q_k + d(i)$ in period i_k from supplier j_k. Thus, the feasible partial solution after covering the period i will be $fp^{(i)} = \{(i_1, j_1, q_1), \cdots, (i_k, j_k, q_k + d(i))\}$. The additional costs would come from the ordering and holding these $d(i)$ materials,

$$w^{(i)} = (c_o(j_k) + (i - i') \cdot c_h) \cdot d(i). \tag{34}$$

Case 3.2. $q_k + d(i) > Q_k$. Under this case, we can only order Q_k in period i_k from supplier j_k. Then, there will remain $d'(i) = q_k + d(i) - Q_k$ materials in period i unmet. The remaining demand is met by another order, denoted by $(i_{k+1}, j_{k+1}, d'(i))$. Similar to the Case 2, this order should satisfy (1) ordering constraint $i_{k+1} \notin \{i_1, \cdots, i_k\}$ and (2) lead time constraint $i_{k+1} + l(i_{k+1}, j_{k+1}) = i$. And the feasible partial solution will be updated to $fp^{(i)} = \{(i_1, j_1, q_1), \cdots, (i_k, j_k, Q_k), (i_{k+1}, j_{k+1}, d'(i))\}$. And the additional costs will come from (1) the costs $w_1^{(i)}$ of ordering and holding of $(Q_k - q_k)$ materials and (2) the costs $w_2^{(i)}$ ordering of $d'(i)$ materials,

$$\begin{aligned} w_1^{(i)} &= (c_o(j_k) + (i - i') \cdot c_h) \cdot (Q_k - q_k) \\ w_2^{(i)} &= c_o(j_{k+1}) \cdot d'(i) + c_b(j_{k+1}). \end{aligned} \tag{35}$$

Thus, $w^{(i)} = w_1^{(i)} + w_2^{(i)}$.

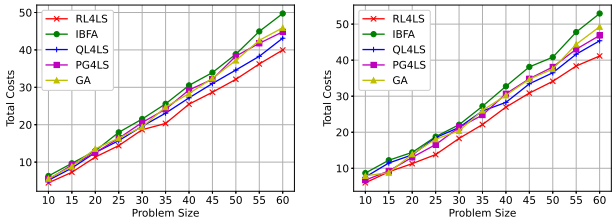
Note that for Case 3.1 and Case 3.2, there is a change on the inventory level at the beginning of the period i after we have the feasible partial solution $fp^{(i)}$. For Case 3.1, $v(i) = d(i)$ while for Case 3.2, $v(i) = Q_k - q_k = V + d(i') - v(i') - q_k$ at the beginning of period i, respectively. These values are useful when we calculate the $Q_{k'}$ (Eq. 33) for some other periods later.

Above three cases contains all possible block divisions of planning horizon. Thus, the offline optimal must lie in one of the feasible partial solutions of $fp^{(T)}$. Thus, we calculate the offline optimal by the following steps. Given an instance of the offline LP-PMU problem, where it includes T periods and the demands and lead time are known in advance, we construct a graph $G = (V, E)$. Each vertex represents a feasible partial solution $fp^{(i)}$ $(i \in [0, T])$, where $fp^{(0)}$ is the empty feasible partial solution. For each expansion from $fp^{(i)}$ to $fp^{(i+1)}$ using the above cases, there is a weighted direct arc $e \in E$ with its weight $w_e = w^{(i)}$ from $fp^{(i)}$ to

$fp^{(i+1)}$. In addition, there is a sink vertex t. For the vertices representing $fp^{(T)}$, they are all connected to a sink vertex t with a weighted direct edge from $fp^{(T)}$ to t with its weight equals 0. Thus, the offline LP-PMU problem can be solved by finding the shortest path from $fp^{(0)}$ to t.

Time Complexity. For each feasible partial solution $fp^{(i)}$, there are at most $2|S|+1$ expansions. Therefore, in the graph G, there are $O((2S+1)^T)$ vertices and $O((2|S|+1)^T)$ edges. Since the running time of the shortest path by Dijkstra [42] is $O(E + V \log V)$, the time complexity of running offline LP-PMU problem would be $O(T \cdot (2|S|+1)^T \log(2|S|+1))$.

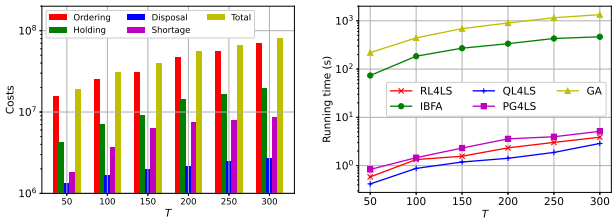
APPENDIX C
ADDITIONAL EXPERIMENTAL RESULTS



(a) Total costs ($\times 10^7$) on zero lead time (b) Total costs ($\times 10^7$) on constant demand

Fig. 6. Results on zero lead time and constant demand.

(5) Results on Zero Lead Time and Fixed Demand (Synthetic Datasets). In some previous work, there is no lead time (orders arrive immediately) or the demands are considered as deterministic. We also do such studies to test our algorithms. The results on zero lead time are presented in Figure 6(a). We can see that our algorithm also has the lowest total costs among all algorithms, and the value of the total costs in this setting is smaller than that of origin problem. This is because when the orders can arrive immediately, some unmet demands result from the shortage could be met by using the arriving materials, in which the shortage costs become smaller. As for the deterministic demands, the results are shown in Figure 6(b). Our algorithm also performs better than other algorithms at most time.



(a) Costs on varying T (RL4LS) (b) Running time on varying T

Fig. 7. Results on varying the maximum time period T.

(6) Results on varying the maximum time period T. We study the effects of the maximum time period T by varying its value from the range $\{50, 100, 150, 200, 250, 300\}$. We choose the problem size to be 10×10. Among all algorithms, RL4LS achieves the best performance, i.e., the least cost. Figure 7(a) decomposes the total costs of RL4LS into different categories. We can see that all costs increase as the maximum time period T increases. Among the costs, ordering cost takes up to 75%-85% of the total cost, which shows a similar trend as that on the real dataset. Figure 7(b) shows the results of the running time of different algorithms. Three RL-based algorithms run faster than the nature-inspired heuristics. Specifically, when T reaches 300, RL4LS runs in 3.8 seconds, showing its excellent efficiency.