# Single-site Perishable Inventory Management under Uncertainties: A Deep Reinforcement Learning Approach (Technical Report)

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

## 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

$$g(x) = \min_{j \in S}\{c_o(j) \cdot x + c_b(j) \cdot I(x)\}, \quad x \geq 0. \tag{1}$$

Then, the costs for time period $i$ would be,

$$\begin{aligned} C(i) &= C_h(i) + C_o(i) + C_d(i) \\ &\geq (c_h + c_d) \cdot (o(i) - d(i)) + g(o(i)) \\ &= (c_h + c_d) \cdot o(i) + g(o(i)) - (c_h + c_d) \cdot d(i). \end{aligned} \tag{2}$$

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

$$C^*(i) = g(d(i)). \tag{3}$$

Thus, the competitive ratio can be computed by

$$\begin{aligned} cr(i) &= \max_{x \in \mathcal{X}} \frac{obj(A(x))}{obj(A^*(x))} \\ &\geq \max_{d(i) \geq 0} \frac{(c_h + c_d) \cdot o(i) - (c_h + c_d) \cdot d(i) + g(o(i))}{g(d(i))} \\ &= \lim_{x \to 0^+} \frac{(c_h + c_d) \cdot o(i) - (c_h + c_d) \cdot x + g(o(i))}{g(x)} \\ &= \frac{(c_h + c_d) \cdot o(i) + f(o(i))}{f(0)} \end{aligned} \tag{4}$$

**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,

$$\begin{aligned} C(i) &= C_o(i) + C_s(i) \\ &\geq g(o(i)) + c_s \cdot (d(i) - o(i)) \\ &= \min_{j \in S}\{(c_o(j) - c_s) \cdot o(i) + c_b(j) \cdot I(o(i))\} + c_s \cdot d(i), \end{aligned} \tag{5}$$

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

$$C^*(i) = g(d(i)). \tag{6}$$

Thus, the competitive ratio can be computed by

$$\begin{aligned} cr(i) &= \max_{x \in \mathcal{X}} \frac{obj(A(x))}{obj(A^*(x))} \\ &\geq \max_{d(i) \geq 0} \frac{g(o(i)) + c_s \cdot (d(i) - o(i))}{g(d(i))} \\ &= \lim_{x \to +\infty} \frac{g(o(i)) + c_s \cdot (x - o(i))}{g(x)} \\ &= \frac{c_s}{\min_{j \in S}\{c_o(j)\}} \end{aligned} \tag{7}$$


- *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*

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

$$cr(i) \geq \max\left(\frac{(c_h + c_d)\cdot o(i) + f(o(i))}{f(0)}, \frac{c_s}{\min\limits_{j\in S}\{c_o(j)\}}\right) \quad (8)$$

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

$$\begin{aligned} cr(A) &= \max_{x\in\mathcal{X}} \frac{obj(A(x))}{obj(A^*(x))} \\ &= \max_{\boldsymbol{d}\geq\mathbf{0}} \frac{\sum_{i=1}^{T} C(i)}{\sum_{i=1}^{T} C^*(i)} \\ &= \frac{\sum_{i=1}^{T} cr(i)\cdot C^*(i)}{\sum_{i=1}^{T} C^*(i)} \\ &\geq \min_i cr(i). \end{aligned} \quad (9)$$

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 [1]). *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 [1] 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

$$w^{(i)} = c_s \cdot d(i). \quad (10)$$

**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

$$w^{(i)} = c_o(j_{k+1}) \cdot d(i) + c_b(j_{k+1}). \quad (11)$$

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

$$Q_k = a(i') \leq V + d(i') - v(i'). \tag{12}$$

**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,

$$w^{(i)} = (c_o(j_k) + (i - i') \cdot c_h) \cdot d(i). \tag{13}$$

**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,

$$\begin{aligned} w_1^{(i)} &= (c_o(j_k) + (i - i') \cdot c_h) \cdot (Q_k - q_k) \\ w_2^{(i)} &= c_o(j_{k+1}) \cdot d'(i) + c_b(j_{k+1}). \end{aligned} \tag{14}$$

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. 12) 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 [2] 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
## PSEUDO-CODES OF RL4LS ALGORITHM

The pseudo-codes of RL4LS is presented in Algorithm 1.

**Algorithm 1:** RL4LS algorithm

**Input:** Demands $d(i)$ and lead time $l(i, j)$ for all $i$ and $j$, where they arrive in an online fashion.
**Output:** Plan $o(i, j)$ for all $i$ and $j$, total costs $C$.

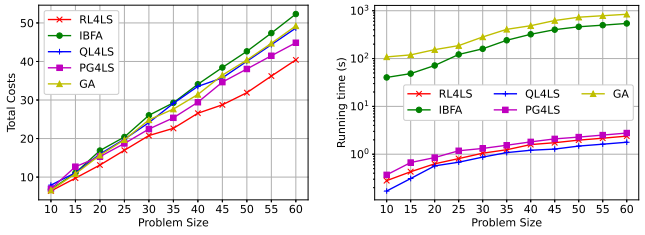
$C \leftarrow 0$;
**foreach** $i \in [1, T]$ **do**
  $s_i \leftarrow$ observe the inventory status as Eq. (11);
  Take an action $a_i = [j^*, q^*]$ which satisfies Eq. (15);
  Output $o(i, j^*) = q^*$ and $o(i, j) = 0$ for $j \neq j^*$;
  $C(i) \leftarrow$ total costs in the time period $i$;
  $C \leftarrow C + C(i)$;
**return** $C$;

## APPENDIX D
## ADDITIONAL EXPERIMENTAL SETUP

**Model Training.** The real dataset consists of the history data from 2015 to 2020. We use the first five years for training and the data of the last year for testing. The networks $Q(s, a; \omega)$ and $\mu(s; \theta)$ are both composed of a hidden layer and an output layer. In both hidden layers, we use the tanh function as the activation function with 64 neurons. In the output layers, we use linear function and sigmoid function with $|S|$ neurons for $Q$ and $\mu$, respectively. For training, we set the discount factor $\gamma$ to 1, and use the Adam optimiser with constant learning rate $10^{-4}$. We adopt $\epsilon$-greedy process and set $\epsilon = 0.1$. 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.

**Hyperparameter Setting.** 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 $|M| \times |S|$.

## APPENDIX E
## ADDITIONAL EXPERIMENTAL RESULTS

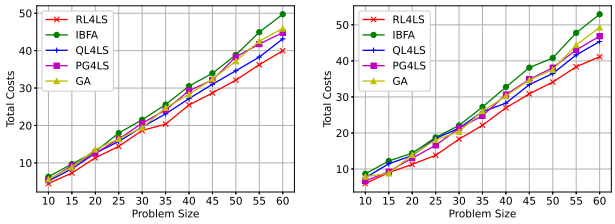



(a) Total costs ($\times 10^7$) (b) Running time

Fig. 1. Results on synthetic dataset.

**(5) Results on Synthetic Dataset**. The results of the synthetic datasets are present in Figure 1. Since the demands and lead time generated in [3] 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

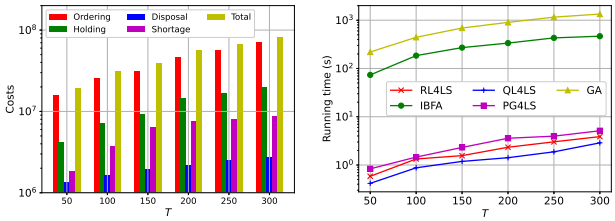
other algorithms under most cases. GA has the second best results when the problem size is small (i.e., problem size $< 20 \times 20$) since GA could generate a large search space due to the chromosome structure, and it is easy to extract a good solution. However, when the problem size increases, it 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.




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

Fig. 2. Results on zero lead time and constant demand.

**(6) 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 2(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 2(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. 3. Results on varying the maximum time period $T$.

**(7) 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 $10 \times 10$. Among all algorithms, RL4LS achieves the best performance, i.e., the least cost. Figure 3(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 3(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.

## REFERENCES


[1] 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.

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

[3] 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.