Regular Paper

Multi-agent Reinforcement Learning Based Approach for Periodic-review Joint Replenishment Problem under Practical Cost Structures

Hiroshi Suetsugu[1,a)] Yoshiaki Narusue[1,b)] Hiroyuki Morikawa[1,c)]

Received: August 28, 2020, Revised: October 20, 2020,
Accepted: December 2, 2020

Abstract: A periodic-review joint replenishment problem is considered. In literature, can-order and modified periodic-review policies have been proposed, and either of them cannot always outperform the other depending on the demand characteristics. In addition, whereas numerous types of joint-replenishment cost structures exist in practical settings, most studies have assumed the fixed joint-replenishment costs, and for the periodic-review system, no study has been conducted to incorporate the practical cost structures into the existing policies. In this study, a multi-agent reinforcement learning-based solution for a joint replenishment problem is proposed, which can be used for problems with several demand settings, and be applied for various cost structures with minor modification. Our numerical experiments demonstrate that the performance of our proposed agent equals or surpasses that of the existing policies, which are can-order, and modified periodic policies.

Keywords: joint replenishment problem, multi-product inventory, multi-agent reinforcement learning, credit assignment, joint action selection

1. Introduction

A joint replenishment problem (JRP) [6] under stochastic demands in a periodic-review system, where joint-replenishment costs are shared among products, and replenishment opportunity comes at a regular time intervals, is considered. The recent increase in e-commerce has led smaller companies to participate in international transportation using trucks or container ships. In such a situation, the practical consideration of JRP under stochastic demands is required due to the high demand deviation and the high ratio of the shared cost to the total supply chain cost.

While most studies in JRPs have considered the deterministic demands, those that considered the stochastic demands are limited [6]. The Markov decision processes (MDP) have been used for formulating the problem regarding stochastic demands. The MDP can only be solved when the number of products is less than four because the action spaces grow exponentially with the number of products considering its combinatorial nature [17], [18]. Thus, several approximated class of policies have been proposed for a large number of products.

However, some studies have reported the pros and cons of each class of policy depending on the demand characteristics, e.g., demand deviation or correlation among products. Specifically, for the well-known coordinated policy, called can-order, or (s, c, S) policy, its performance would be worse compared to the optimal solution for two-product case, when the demand correlation among products exists [7]. Another policy, called the modified periodic (MP) policy, has been reported to achieve good performance compared to the can-order policy [20]. However, some studies [11] specified that the MP policy could not outperform the can-order policy for problems with high demand deviation.

In addition, although most studies have assumed fixed replenishment cost, many types of cost structures exist in practice, e.g., capacity constraint or stepwise cost for container or truck shipment, nonlinear cost for warehouse costs, etc. Studies [12], [14], [15] incorporated the warehouse or truck capacity constraint into the existing policies. However, to the best of our knowledge, such attempts have been only limited to the continuous-review inventory system while we focus our attention on the periodic-review inventory system.

Previously, a reinforcement learning based agent has been proposed, called the branching deep-Q network with reward allocation [21]. The previous approach is uniquely characterized by shared representation of state-action value function followed by the product-independent branches with credit assignment mechanism, encouraging the cooperative behavior among agents while enabling the linear growth of the total number of network outputs regarding the number of products. However, it could not outperform the existing approximated policies when the number of products became large.

In this study, we proposed a multi-agent reinforcement learning-based solution for a JRP that does not assume a specific class of policies, unlike existing approach including can-order

[1] Graduate School of Engineering, The University of Tokyo, Bunkyo, Tokyo 113–0033, Japan
a) hsuetsugu@sglab.co.jp
b) narusue@mlab.t.u-tokyo.ac.jp
c) mori@mlab.t.u-tokyo.ac.jp

and modified periodic policies. Our solution can be used for problems with several demand settings, in terms of the number of products, demand deviation, and demand correlation among products, and be applied for various cost structures with minor modification, including the capacitated transportion, stepwise-transportation costs, and nonlinear holding costs.

We used the reinforcement learning (RL), that requires no knowledge on the state transition probability and reward function, which makes our solution applicable to various cost structures. However, naively formulating as a single agent RL would suffer from the large action spaces as stated above. Thus, we use the multi-agent RL, where each agent comprises each product. The distinguishing requirement for the solution of JRP is *cooperation* among products, and for our agent to be *cooperative*, instead of taking action independently among agents, we introduced a central joint-action selection unit that decides the joint-action of all agents simultaneously, with each agent learning the state-action value function that is conditioned by the other agents' actions. By focusing on the credit assignment strategy, which decides how to allocate the joint costs among products, we used a heuristic joint-action search procedure which efficiently searches the joint-action for larger total state-action value among all agents.

We found that the performance of our proposed agent equaled the better one of the benchmark policies irrespective of the demand characteristics with fixed replenishment cost by conducting several numerical experiments. In addition, our proposed agent can easily incorporate several cost structures and outperformed the benchmark policies for cases with transportation capacity constraint or stepwise-transportation cost.

The remainder of this paper is as follows: Section 2 reviews the literature on both periodic-review JRP and multi-agent RL. Sections 3 and 4 presents our proposed solution and results of the numerical experiment, respectively, and Section 5 provides the conclusions of the paper.

2. Related Works

The literature review of periodic-review JRP with stochastic demands is first considered in this section. Then, we proceed to the survey of RL in multi-agent setting.

2.1 Periodic-review Joint Replenishment Problem under Stochastic Demand

Two types of inventory review systems are continuous-review and periodic-review system. In the continuous-review inventory system, the inventory level is continuously monitored and an order can be made when needed, while in the periodic-review inventory system, the inventory level can only be monitored, or an order can be put at a specific period. In practice, although the level of inventory can be continuously monitored due to the prevalence of information technology in the logistics industry, replenishment opportunities are often limited to a specific period either daily or weekly due to the shipment operation constraints.

For a periodic-review JRP, several approximated class of policies have been proposed in the literature. A well-known can-order policy was first proposed for the continuous-review inventory system by Balintfy [3], which places a joint order whenever the inventory position of a product reaches its must-order level, and other products will be included in the order if their inventory positions are at or below the can-order level. Then, the policy has been extended to be used for a periodic-review system [11]. Another class of joint replenishment policy for periodic review system is a periodic policy or (T, **S**) policy proposed by Atkins and Iyogun [2], which places order every T periods of time. This policy has been improved to modified periodic policy or MP policy [24].

However, the superiority of the performance between the can-order and MP policies depends on the demand deviation and the existence of demand correlation. Studies [7], [14], [15] have reported the weakness of the can-order policy when demand correlation exists. MP policy was reported to be a good choice over can-order policy [20], whereas the study [11] stated that can-order policy outperformed MP policy for problems with high-demand deviation.

Also, most studies have assumed the fixed ordering cost as a joint replenishment cost, i.e., the fixed cost is incurred when the replenishment take place irrespective of the volume of replenishment quantities. However, practically, many variations exist based on the joint replenishment cost. Specifically, when goods are delivered via container ships or trucks, and the transportation cost depends on the number of vehicles, the transportation cost has a stepwise function regarding the number of required vehicles. When the company has its own warehouse and they can use leased additional warehouse if their inventory level exceeds their warehouse capacity, the inventory holding cost becomes nonlinear function. Studies [12], [14], [15] incorporated the warehouse or truck capacity constraint into the existing policies. However, to the best of our knowledge, such attempts have only been limited to the continuous-review inventory system, and required modification to incorporate such additional cost structure into the existing policies is demanding.

2.2 Multi-agent Reinforcement Learning

RL is an approach to obtain a solution for a MDP without any knowledge about the state transition probability and reward function. RL maximizes the long-term discounted reward per action. Q-learning is based on estimating the expected total discounted future rewards of each state-action pair under the policy π. The state-action value function, or Q function is expressed as follows:

$$Q_\pi(s_t, a_t) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-t} r_T | \pi\right], \quad (1)$$

where s_t, a_t, r_t, and γ denote the states, action, reward, and discount factor, respectively. The Q function can be computed recursively using dynamic programming as follows:

$$Q^\pi(s, a) = \mathbb{E}_{s'}\left[r + \gamma \mathbb{E}_{a' \sim \pi(s')}\left[Q^\pi(s', a')\right] | s, a, \pi\right]. \quad (2)$$

Let the optimal $Q^*(s, a)$ denote $\max_\pi Q^\pi(s, a)$. Then, the optimal Q function satisfies the Bellman equation: $Q^*(s, a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s', a') | s, a\right]$. Q-learning is an off-policy TD control algorithm, and the one-step Q-learning is defined by:

$$Q(s_t, a_t) = (1 - \alpha_t) Q(s_t, a_t) + \alpha_t \left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a)\right), \quad (3)$$

where α denotes the learning rate.

For a large state space, Q-learning with function approximation has been proposed. In the function approximated Q-learning using the neural net, the following loss function needs to be minimized in the training process:

$$L(\theta) = \mathbb{E}[(y - Q(s, a; \theta))^2], \tag{4}$$

where

$$y = r + \gamma \max_{a'} Q(s', a'; \theta'), \tag{5}$$

and θ is the parameter of the neural network. A target network and experience replay have been proposed to cope with the problems related to non-stationarity and correlation in the sequence of observations [16].

Independent Q-learning(IQL) [22] has been proposed for multi-agent setting, where each agent treats other agents as a part of the environment. When there are multiple learners in the environment, one needs to set the credit assignment policy with which the obtained reward is allocated to each agent. According to the study [5], the simplest solution is to split the team reward equally among each of the learners, called global reward. Generally, the global reward allocation has been reported not to scale well to increasingly difficult problems as the learners do not have sufficient feedback tailored to their own specific actions. Another way is to assess each agent's performance based solely on its individual behavior, which is called local reward. In this case, this provides no incentive for an agent to help other agents. Local reward induces faster learning rates, but not necessarily to better results than global reward [5].

Several studies have shown using multi-agent RL for inherently a single agent problem by handling the large action space. Tavakoli [23] proposed the action branching architectures, a kind of deep Q-network (DQN) treating each actuator independently in robot control. It has the n dimensional action branches following the shared state representation in the neural network Q-function approximation, while enabling the linear growth of the total number of network outputs with increasing the dimensionality of action. By incorporating the reward allocation mechanism into this branching architecture, the branching deep Q-network with reward allocation for JRP has been proposed [21]. The proposed agent could learn the coordinated replenishment policy without any knowledge of other products' decision due to the existence of the shared representation in the state-action value function. However, as the number of products increases, the performance of the agent became worse compared to the existing approximated policies.

In the above-mentioned study [21], two types of credit assignments have been examined; one is the global reward allocation, with which the joint-replenishment cost is allocated equally to each agent, and the other is the local reward allocation, with which the joint-replenishment cost is allocated depending on the order quantities of each product. Depending on the credit assignment, the different characteristics of the agent's learnt behavior have been reported.

The agent with global reward could put an order simultaneously at the specific timing, called *coordinate-ordering* behavior, but could not learn to simultaneously stop putting an order at a specific timing, called *coordinated-canceling* behavior. It could not learn *coordinated-canceling* because, for an agent who places an order at a certain timing, the allocated joint cost is equal irrespective of the other agent's action, and does not have large incentive to stop an order even if any other agent would not put an order. Conversely, the agent with local reward allocation could learn the *coordinated-cancelling*, while could not learn the *coordinated-ordering*. It could not learn *coordinated-ordering* because an agent without order does not incur any joint cost allocation, and has no incentive of putting an order simultaneously with other products.

Therefore, the application of multi-agent RL for JRP, where each agent decides on its action independently, was found to be difficult for a larger number of products. In addition, the perfect credit assignment strategy for JRP was not identified in literature. With this in mind, a solution that explicitly takes into account the other agents' action while being able to handle the large joint-action space is required.

3. Method

3.1 Problem Setting

We consider a multi-product inventory system between one supplier and one retailer in a periodic-review system under stochastic stationary demands to minimize the total retailer cost, which includes the holding, penalty, and transportation costs. We have used the following notations:

i: Item number, $i = 1, \ldots, N$,
t: Period, $t = 1, \ldots, T$,
LT: Lead time from supplier to retailer, (in weeks),
l_i: Lot size of item i, (in palette),
$d_{i,t}$: Demand for item i during period t, (in palette),
$x_{i,t}$: Order quantity for item i made at time t, (in palette),
$Out_{i,t}$: Shipment of item i from retailer during period t, (in palette),
$In_{i,t}$: Replenishment for item i from supplier during period t, (in palette),
$I_{i,t}$: Inventory of item i at the start of time t, (in palette),
$I^p_{i,t}$: Inventory position of item i at the start of time t, (in palette),
$u_{i,t}$: Unsatisfied demand of item i during period t, (in palette),

We permitted the lost sales. Replenishment at time t can be used from time $t + 1$. In this study, we do not consider the supplier stock-out or any supply delay. Thus, the relationship among inventory, replenishment, shipment, demand, and unsatisfied demand can be formulated as follows. Here, inventory position means the on-hand inventory plus orders that have been ordered but have yet been received.

$$In_{i,t+LT_i} = x_{i,t}, \tag{6}$$
$$Out_{i,t} = \min(d_{i,t}, I_{i,t}), \tag{7}$$
$$I_{i,t+1} = I_{i,t} - Out_{i,t} + In_{i,t}, \tag{8}$$
$$I^p_{i,t+1} = I^p_{i,t} - Out_{i,t} + x_{i,t}, \tag{9}$$
$$u_{i,t} = d_{i,t} - Out_{i,t}. \tag{10}$$

The order quantity unit size, called the lot size, should be an integer in palette. Demand is specified in decimals because a customer's order to the retailer would be stated in pieces rather than in palettes. In this study, we let *LT* (which is the time required from order to delivery) to be 3 weeks.

3.1.1 Cost Structures

Let C^{trans}, C^{hold}, and C^{pel} denote transportation, holding, and penalty costs, respectively, and U^{trans}, U^{hold}, and U^{pel} are the corresponding unit costs. The penalty cost is defined as follows:

$$C^{\text{pel}}_{i,t} = U^{\text{pel}} \times u_{i,t}. \tag{11}$$

We defined the several cost structures. One is the normal cost setting, often assumed in the literature, where fixed joint-replenishment cost is incurred regardless of the amount of orders, and holding cost is proportionate to the inventory amount:

$$C^{\text{trans}}_{t} = U^{\text{trans}} \left(\text{if} \sum_{i} x_{i,t} > 0 \right), \tag{12}$$

$$C^{\text{hold}}_{i,t} = U^{\text{hold}} \times I_{i,t}. \tag{13}$$

The second type is the capacitated-replenishment cost, where a certain capacity of transportation ($\text{CAP}_{\text{trans}}$) is considered ($\sum_i x_{i,t} \leq \text{CAP}_{\text{trans}}$), assuming the transportation using a truck or a container ship.

The third type is the stepwise-replenishment cost, where transportation cost depends on the number of vehicles required, also assuming the transportation by a truck or a container ship;

$$C^{\text{trans}}_{t} = U^{\text{trans}} \times \left\lceil \frac{\sum_i x_{i,t}}{\text{CAP}_{\text{trans}}} \right\rceil. \tag{14}$$

The fourth type is the nonlinear holding cost, in which the fixed holding cost is incurred when the inventory level is equal or below the certain warehouse capacity (CAP_{wh}), and additional cost is incurred for the level of surplus inventory as a fee for short-term leasing warehouse;

$$C^{\text{hold}}_{t} = U^{\text{hold}_{\text{f}}} + U^{\text{hold}_{\text{v}}} \times \max\left(0, \sum_{i} I_{i,t} - \text{CAP}_{\text{wh}}\right), \tag{15}$$

where $U^{\text{hold}_{\text{f}}} = U^{\text{hold}} \times \text{CAP}_{\text{wh}}$ and $U^{\text{hold}_{\text{v}}} > U^{\text{hold}}$.

Table 1 summarizes the cost setting above, and **Fig. 1** shows the visual presentation of the cost function of the transportation and holding costs.

3.2 Single-agent Formulation

First, we formulate our problem as a single agent problem. In our problem setting, if we consider the inventory level and inventory position at time *t* as state, transportation, holding, and penalty costs depend only on the state and orders placed at time *t*. At every time step, meaning the beginning of every week, the agent obtains information about the on-hand inventory and inventory position, i.e., $s_t = [(I_{i,t}, I^{p}_{i,t})]^{N}_{i=1}$, makes a decision $a_t = (a_0, a_1, \ldots, a_n) \in \mathcal{A}$ based on s_t, and receives an immediate reward r_{t+1}, that is, the sum of the transportation, holding, and penalty costs incurred at time *t*.

Thus, the single agent Q-learning can be formulated as Eq. (3). However, the Q-learning (or function-approximated Q-learning, e.g., DQN [16]) cannot converge due to the increasing action spaces when the number of products becomes large.

Table 1 Cost scenario.

Cost scenario	Transportation cost	Holding cost
1) Base	Fixed	Linear
2) Capacitated (trans)	Fixed (with CAP constraint)	Linear
3) Stepwise (trans)	Stepwise	Linear
4) Nonlinear (hold)	Fixed	Nonlinear

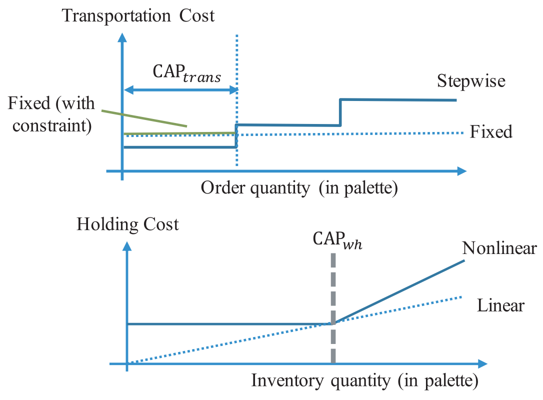

Fig. 1 Transportation and holding cost.

3.3 Multi-agent Formulation

To handle the exponentially increasing action space with respect to the number of products, we used the multi-agent approach. The simplest way to formulate as a multi-agent problem is the IQL, where each agent treats other agents' action as part of the environment, that is;

$$\begin{aligned} Q_i(s_{i,t}, a_{i,t}) &= (1 - \alpha_t) Q_i(s_{i,t}, a_{i,t}) \\ &\quad + \alpha_t \left(r_{i,t+1} + \gamma \max_{a_i} Q(s_{i,t+1}, a_i) \right), \end{aligned} \tag{16}$$

where only the difference from the Eq. (3) is the existence of subscript *i*, and the $r_{i,t+1}$ is the allocated reward for product *i* according to a certain credit assignment strategy. However, IQL often fails to converge as changes in the policy of one agent will affect that of the others, and vice versa. This problem becomes more serious when the function approximated Q-learning using neural nets is used, where experience replay plays a key role [8].

The previous work [21] was an extended version of IQL that has shared representation with the aim of encouraging the agents to exhibit a coordinated behavior. Although it could learn the coordinated ordering behavior, the performance was not found to be better than the existing coordinated approximated policies, such as can-order and periodic review policies. Thus, we started incorporating the central control unit that explicitly considers all the agents' actions.

In **Fig. 2**, our proposed solution is presented. Each agent comprises one product, and the agent for product *i* has its own state-action value function with parameter θ_i. What differs most from the earlier work [21] is that the information of the other agents' actions was treated as part of a state in the state-action value function. Thus, the state-action value function of each agent is conditioned by the actions of the other agents rather than selecting an action independently. Moreover, we have a central joint-action selection unit, where the joint-action is decided based on the Q-value function of all the agents. Algorithm 1 describes the pseudo-code of our proposed solution. The explanation of the joint-action selection, credit assignment, and the ϵ-greedy explo-

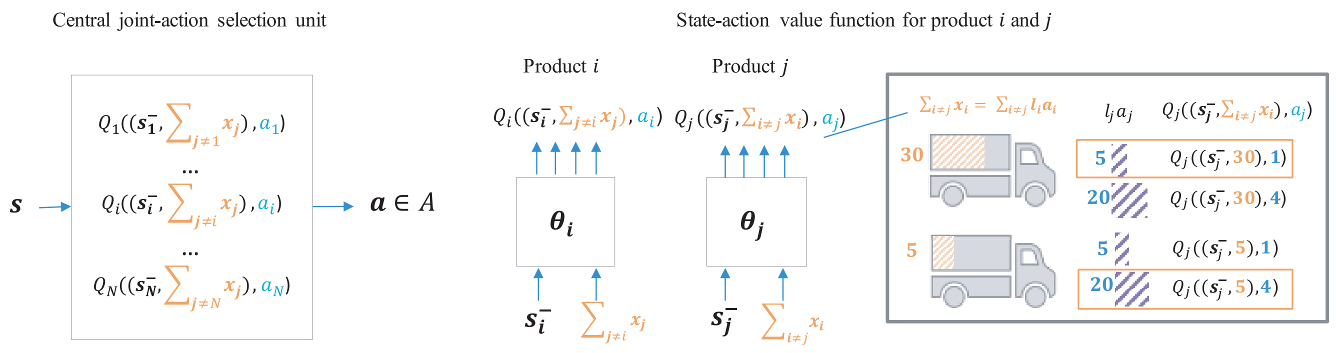
Central joint-action selection unit
State-action value function for product i and j
Product i
Product j
s
$Q_1((s_1^-, \sum_{j\neq 1} x_j), a_1)$
$Q_i((s_i^-, \sum_{j\neq i} x_j), a_i)$
$Q_N((s_N^-, \sum_{j\neq N} x_j), a_N)$
$a \in A$
$Q_i((s_i^-, \sum_{j\neq i} x_j), a_i)$
$Q_j((s_j^-, \sum_{i\neq j} x_i), a_j)$
θ_i
θ_j
s_i^-
$\sum_{j\neq i} x_j$
s_j^-
$\sum_{i\neq j} x_i$
$\sum_{i\neq j} x_i = \sum_{i\neq j} l_i a_i$
$l_j a_j$
$Q_j((s_j^-, \sum_{i\neq j} x_i), a_j)$
30
5
5
20
5
20
$Q_j((s_j^-, 30), 1)$
$Q_j((s_j^-, 30), 4)$
$Q_j((s_j^-, 5), 1)$
$Q_j((s_j^-, 5), 4)$

Fig. 2 each agent's state-action value function conditioned by sum of other agent's order quantities.

Algorithm 1 Learning procedure

```
Require: $K_e$, $K_s$, $\mathrm{Freq}_{target}$
  Initialize:
    $Q_i, \hat{Q}_i$ for all $i$
  for episode = $1, \cdots, K_e$ do
    Initialize:
      $s_1, a_1$
    for $t = 1, \cdots, K_s$ do
      if random $\in [0, 1] < \epsilon$ then:                     ▷ $\epsilon$-greedy exploration
        $a_t$ ← select action at random
        $s_t$ ← update states based on the randomly selected actions
      $r_{t+1}$ ← simulate inventory movement based on (states, actions)
      $[r_{i,t+1}]_{i=1}^N$ ← allocate reward $r_{t+1}$                 ▷ credit assignment
      $s_{t+1}, a_t$ ← action selection in central unit   ▷ joint action selection
      Memory $\mathcal{D}_i$ ← $(s_{i,t}, a_{i,t}, s_{i,t+1}, r_{i,t+1})$ for all $i$
      update parameters of $Q_i$ for all $i$
    if episode $= 0 \pmod{\mathrm{Freq}_{target}}$ then:
      $\hat{Q}_i$ ← $Q_i$ for all $i$
```

ration strategy are provided in the following section.

When the holding cost is proportionate to the inventory quantity (in the cost scenarios 1, 2, and 3), a state of each agent i aside from the other agent's action is only an inventory and inventory position of product i. Thus, we can define the state by $s_{i,t} = (s_{i,t}^-, \sum_{j\neq i} x_{j,t})$, where $s_{i,t}^- = (I_{i,t}, I_{i,t}^p)$ and $x_{j,t} = l_j a_{j,t}$ is the order quantity of the agent j when selecting an action a_j. For the nonlinear holding cost (in the cost scenario 4), the inventory level of the other products should be considered. In this case, the sum of the inventory of all the products $\sum_i I_{i,t}$ is added to the aforementioned state.

At every time step, each agent in state $s_{i,t}$ takes an action $a_{i,t}$ (that is decided by the central joint-action selection unit), receives the immediate allocated reward $r_{i,t+1}$ (through credit assignment), and moves to the next state $s_{i,t+1}$. Because an infinite action space is impractical, and taking a large number of orders compared with the demand is unrealistic from the point of view of a supply chain, we limited the possible order quantity for product i to $X_i = \{l_i a_i \mid a_i \in \mathcal{A}_i = \{0, 1, 2, 3, 4, 5\}\}$ where $\mathcal{A}_i$ denotes the action space for product i. In this setting, the joint-action space of all the products would be 6^n. The state-action value of each agent with function approximation is updated according to the following loss:

$$L_i = \mathbb{E}_{(s_i, a_i, r_i, s_i') \sim \mathcal{D}_i} \left[L_\delta \left(y_i, Q_i \left(s_i, a_i \right) \right) \right], \tag{17}$$

where

$$y_i = r_i + \gamma Q_i^- \left(s_i', \arg\max_{a_i} Q_i(s_i', a_i) \right), \tag{18}$$

L_δ denotes the Huber loss function, and Q^- denotes the target network.

3.3.1 Credit Assignment

Since an environment has several agents, a credit assignment should be decided, which satisfies $\sum_i r_{i,t}$ equals the negative sum of the transportation, holding, and penalty costs at time t. For the transportation cost, we used the global reward allocation, where the transportation cost C_t^{trans} is equally allocated to each product, and the holding cost can be calculated independently per product (Eq. (13)) for the linear holding cost scenario (cost scenarios 1, 2, and 3), that is:

$$r_{i,t+1} = -(C_t^{\mathrm{trans}}/N + C_{i,t}^{\mathrm{hold}} + C_{i,t}^{\mathrm{pel}}). \tag{19}$$

This credit assignment strategy is tightly coupled with the joint-action heuristics which we will explain later.

When the holding cost was also a joint-cost among products in cost scenario 4 (Eq. (14)), we used the local and global reward allocations for the variable and the fixed part of the holding cost, respectively. Thus, allocated reward for each agent is defined as follows:

$$r_{i,t+1} = -\left(C_t^{\mathrm{trans}}/N + U^{\mathrm{hold_f}}/N + C_t^{\mathrm{hold_f}} \times \frac{I_{i,t}}{\sum_i I_{i,t}} + C_{i,t}^{\mathrm{pel}} \right), \tag{20}$$

where $C_t^{\mathrm{hold_f}} = C_t^{\mathrm{hold}} - U^{\mathrm{hold_f}}$.

3.3.2 Joint ϵ-greedy Exploration Strategy

We employed the joint ϵ-greedy exploration strategy, in which all the agents jointly select an action randomly with probability ϵ to explore the *joint* replenishment opportunities. Moreover, each agent takes a random action independently with the probability ϵ/N.

3.3.3 Joint-action Selection

Since our goal is to derive the policy that obtains the maximum total expected discounted reward, we need to take the following joint-action:

$$a = (a_0, a_1, \ldots, a_n) = \arg\max_{a \in A} \sum_k Q_k \left(\left(s_k^-, \sum_{j\neq k} x_j \right), a_k \right). \tag{21}$$

Here, we omitted the time index t for simplicity.

We, however, still face a combinatorial action selection problem. In the learning process, we need to solve this optimization problem for each step. Thus, the efficient heuristics for the Eq. (21) is needed.

Algorithm 2 Joint-action selection

```
Require: $[s_i]_{i=1}^N$, $[Q_i]_{i=1}^N$, $K$
  function EVALUATE($\boldsymbol{a}$)
      $x_i \leftarrow l_i a_i$ for all $i$
      $q_{this} \leftarrow \sum_i Q_i(s_i, \sum_{j \neq i}(x_j), a_i)$
      if ($q_{this} > q_{best}$) then
          $q_{best} \leftarrow q_{this}$
          $\boldsymbol{a}_{best} \leftarrow \boldsymbol{a}$
      return $q_{this}$
  function SEARCH($\boldsymbol{x}$, sequential:bool, start:integer=1)
      for $i = 1, \cdots, N$ (starting from start) do
          $a_i \leftarrow \arg\max Q_i(s_i, \sum_{j \neq i}(x_j))$
          if sequential then:
              $x_i \leftarrow l_i a_i$
      $\boldsymbol{x} \leftarrow \{l_i a_i\}_{i=1}^N$
      return $\boldsymbol{a}, \boldsymbol{x}$
  Initialize:
      $\boldsymbol{a}_{best} \leftarrow (0, 0, \cdots, 0)$
      $q_{best} \leftarrow$ evaluate($\boldsymbol{a}_{best}$)
  for $k = 1, \cdots, K$ do
      $\boldsymbol{a}, \boldsymbol{x} \leftarrow$ search($\boldsymbol{x}$, sequential=False)          ▹ batch search
      $q_{best}^k \leftarrow$ evaluate($\boldsymbol{a}$)
      for $i = 1, \cdots, N$ do
          $\boldsymbol{a}, \boldsymbol{x}^{seq} \leftarrow$ search($\boldsymbol{x}$, sequential=True, start=$i$)   ▹ sequential search
          $q^{seq} \leftarrow$ evaluate($\boldsymbol{a}$)
          if ($q^{sec} > q_{best}^k$) then:
              $q_{best}^k \leftarrow q^{sec}$
              $\boldsymbol{x} \leftarrow \boldsymbol{x}^{seq}$
  return $\boldsymbol{a}_{best}$
```

3.3.4 Joint-action Selection Heuristics

With the availability of the state-action value function of each agent, each agent decision can be modified by treating the other agents' actions as fixed. By incrementally modifying joint-action tuple, we can efficiently search the action space that would achieve higher expected discounted reward. Our algorithm is shown in Algorithm 2 and illustration is provided in **Fig. 3**.

As described in Fig. 3, we started our search from the initial solution. In each iteration, each agent updates its decision by assuming the other agent's action is the current solution. If the current solution is $(a_0, \ldots, a_i, \ldots, a_N)$, the sum of all the other agents' replenishment quantity for agent i is $\sum_{j \neq i} l_j a_j$. Thus, agent i can update its action given the state $\boldsymbol{s}_i^-$ such that Q_i is maximized, i.e., $a_i \leftarrow \arg\max Q_i((\boldsymbol{s}_i^-, \sum_{j \neq i} l_j a_j), a)$.

Here, we use the characteristics of the learned behavior depending on the credit assignment. As stated in Section 2, the agent with the global reward allocation failed to learn the *coordinated-canceling* behavior. However, if we start our search from the joint-action, where any product does not put an order, that is, $\boldsymbol{a} = (0, 0, \ldots, 0)$, the *coordinated-canceling* opportunity is examined every time.

If we use the local reward allocation, where a joint-transportation cost is allocated in proportion to the order quantities, the incremental process will not work because for an agent given $\sum x_{j \neq i} = 0$, all the joint-transportation cost will be allocated to this specific product, and this cost allocation is too expensive for an agent to put an order. This is the reason why the agent with the local credit assignment failed to learn the *coordinated-ordering* in the earlier study [21].

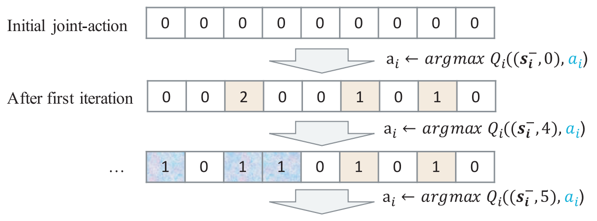
Initial joint-action
0 0 0 0 0 0 0 0 0
$a_i \leftarrow argmax\ Q_i((\boldsymbol{s}_i^-, 0), a_i)$
After first iteration
0 0 2 0 0 1 0 1 0
$a_i \leftarrow argmax\ Q_i((\boldsymbol{s}_i^-, 4), a_i)$
…
1 0 1 1 0 1 0 1 0
$a_i \leftarrow argmax\ Q_i((\boldsymbol{s}_i^-, 5), a_i)$

Fig. 3 Image of our heuristic search procedure.

However, with the global reward allocation, given that the other agents do not put an order at all, the allocated cost for a specific agent is $1/N$. Thus, an agent whose inventory level is low would put an order according to the state-action value function. Once more than one agent decides to put an order in our heuristic algorithm, for agents whose inventory level is not so low, given the updated other agents actions, they may place an order because joint-transportation cost would be allocated regardless of their own actions, and *coordinated-ordering* behavior can emerge.

In our algorithm, we update each agent's action both in batch and sequential manner. In batch manner, each agent updates its decision based on the other agents' action at the beginning of the each iteration. In the sequential manner, each agent's action is sequentially updated, and each agent decides on its action based on the updated actions of the other agents. Since the order of updating its action influences the results of the action selection in the sequential manner, in every iteration, we attempt to use all the agents as a sequential search starting point, while randomly selecting agents for the second and subsequent orders.

For the capacitated cost scenario, where there is transportation capacity constraint, only the joint-actions whose sum of replenishment quantity is equal or below the transportation capacity $\mathrm{CAP_{trans}}$ is allowed in our search procedure.

4. Experiments

4.1 Experimental Setting

We conducted several numerical experiments to answer the following questions:

1) Can the proposed agent achieve the performance equal to or greater than the existing approximated policies under various demand characteristics in the normal cost setting?

2) Can the proposed agent achieve the performance equal to or greater than the existing approximated policies in a situation where non-fixed joint costs exist?

To validate these questions, we conducted experiments by varying 1) number of products, 2) cost scenario, and 3) demand characteristics (demand deviation and correlation among products). In literature, the most frequently examined number of products in JRP ranges from 2 to 12. Thus, we examined three settings: 2, 5, and 10. With regard to the demand characteristics, we altered two parameters; c_v and ρ, which defines the demand deviation and correlation of demand among products, respectively. The demands are generated following the multi-variate normal distribution ($d_t = [d_{i,t}]_{i=1}^N \sim N(\mu, \Sigma)$). The variance for product i is defined as $c_v \times \mu_i$ and the covariance matrix is defined as $\Sigma_{i,j} = \sigma_i \times \sigma_j \times \rho^{|i-j|}$. Per-step average demand μ, lot size of each

Table 2 Demand and lot size settings in each experiment.

Cost scenario	Number of Products	μ	Lot size	CAP_{trans}	CAP_{wh}
1	2	[2, 2]	[4, 4]	-	-
2		[2, 2]	[4, 4]	20	-
3		[15, 15]	[10, 10]	20	-
4		[2, 2]	[4, 4]	-	20
1	5	[0.3, 0.4, 0.5, 0.5, 0.7]	[1, 1, 1, 1, 2]	-	-
2		[0.3, 0.4, 0.5, 0.5, 0.7]	[1, 1, 1, 1, 2]	20	-
3		[3, 4, 5, 5, 7]	[5, 5, 5, 5, 5]	20	-
4		[0.3, 0.4, 0.5, 0.5, 0.7]	[1, 1, 1, 1, 2]	-	20
1	10	[.3, .4, .5, .5, .7, .9, 1., 1., 1.2, 1.2]	[1, 1, 1, 1, 2, 2, 3, 3, 3, 3]	-	-
2		[.3, .4, .5, .5, .7, .9, 1., 1., 1.2, 1.2]	[1, 1, 1, 1, 2, 2, 3, 3, 3, 3]	20	-
3		[1.5, 2, 2.5, 2.5, 3.5, 4.5, 5, 5, 6, 6]	[3, 3, 3, 3, 5, 5, 5, 7, 10, 10]	20	-
4		[.3, .4, .5, .5, .7, .9, 1., 1., 1.2, 1.2]	[1, 1, 1, 1, 2, 2, 3, 3, 3, 3]	-	20

product, and transportation and warehouse capacity settings are presented in **Table 2**. Note that for scenario 3, where stepwise transportation is assumed, we set larger demands so that per-step total demands is larger than the transportation capacity $\mathrm{CAP}_{\mathrm{trans}}$.

To summarize, our experiments comprise the following factors;
1) number of products $\in (2, 5, 10)$,
2) cost scenario $\in (1, 2, 3, 4)$,
3) demand characteristics: $c_v \in (0.4, 0.6), \rho \in (-0.5, 0, 0.5)$

4.2 Benchmark Methodology

As a benchmark policy, we selected a can-order and MP policy, because they are well-known joint-replenishment policy under periodic-review inventory system.

A can-order policy has three parameters for each product: s, c, and S represent the must-order, can-order, and order-up-to level, respectively, whereas a MP policy has one global parameter to represent the frequency of ordering timing, and each product has two parameters: s and S represent must-order and order-up-to level, respectively.

We derived the parameters of each policy using the genetic algorithm (GA). GA is a part of evolutionary computing. GA starts with a set of solutions (represented by chromosomes) called population. The solutions from one population are obtained and employed to form a new population that is better than the old one. The solutions that will be used to form new solutions (offspring) are selected according to their fitness.

For each solution, i.e., parameter set of the benchmark policies, we conducted 12 simulations, and the fitness function was the average total cost for those 12 simulations. Crossover and mutation are two basic operators of GA. We employed a two-point crossover. After performing a crossover, mutation occurs. The number of population, crossover probability, mutation probability, and number of generation are $50 \times N$, 0.5, 0.2, and 100, respectively. Each s, c (for the can-order policy), and S are real-valued, and the global parameter for the MP policy is an integer. We employed the library deap [9] for the implementation with binary coding.

To make the comparison fair, we implemented additional logic for cost scenario 2 and 3 to incorporate the transportation capacity and loading ratio. Similar to the study [4], we implemented a procedure that accounts for the availability of "free" remaining capacity of transportation vehicles that have been partially filled with other items, or remove the products if the sum of order quan-

Algorithm 3 Loading ratio adjustment procedure

```
Require: $\mathrm{CAP}_{\mathrm{trans}}, \boldsymbol{x}, \boldsymbol{I}^p, \boldsymbol{s}_i^1, \boldsymbol{s}_i^2$
  $n \leftarrow \lceil \frac{\sum_i x_{i,t}}{\mathrm{CAP}_{\mathrm{trans}}} \rceil$
  $mode \leftarrow$ decrease or increase depending on loading ratio
  if mode = decrease then                                  ▹ emptying one vehicle
      while $\sum_i x_i > (n-1) \times \mathrm{CAP}_{\mathrm{trans}}$ do
          $j \leftarrow \arg\min_{i \in j | I_j^p < s_j^1} (|S_i - (I_i^p + x_i - l_i)|$
          $x_j \leftarrow x_j - l_j$
  else                                      ▹ filling the "free" remaining capacity
      while $\sum_i x_i < n \times \mathrm{CAP}_{\mathrm{trans}}$ do
          j $\leftarrow \arg\min_{i \in j | I_j^p > s_j^2} (|S_i - (I_i^p + x_i + l_i)|$
          $x_j \leftarrow x_j + l_j$
  return $\boldsymbol{x}$
```

tities exceed its capacity or loading ratio of the vehicle is low, where loading ratio is defined by $\sum_i x_{i,t} / (\lceil \frac{\sum_i x_{i,t}}{\mathrm{CAP}_{\mathrm{trans}}} \rceil \times \mathrm{CAP}_{\mathrm{trans}})$. Algorithm 3 represents the pseudo code of this loading ratio adjustment procedure for cost scenario 3. The procedure for scenario 2 is similar; however, only one vehicle is allowed. Either of can-order and MP policy has the parameter S_i which represents the order-up-to level for product i, and our algorithm chooses the product for adjustment whose adjusted inventory position $I_i^p + \hat{x}_i$ is the nearest to the order-up-to level, where $\hat{x}_i$ represents the adjusted order quantity for product i. Whether to decrease or increase is decided based on the loading-ratio. We select the item based on the above-mentioned criteria among products whose inventory position is equal or less than s^1 for decreasing mode, or greater than s^2 for increasing mode. Here, (s^1, s^2) is (s, c) for a can-order and (s, s) for MP policy.

For the capacitated transportation scenario, the application of the additional heuristic procedure is mandatory to satisfy the constraint regarding the transportation capacity. Conversely, adjustment of the loading ratio is not a must for a stepwise transportation cost scenario.

We confirmed that by adding this procedure, the performance of the benchmark policies increased for the stepwise transportation cost scenario. Section 4.3.3 provides further investigation regarding this loading ratio adjustment procedure.

4.3 Experimental Results

4.3.1 Performance Evaluation

Table 3 shows the experimental results, representing the average and standard deviation of the total cost, and **Table 4** shows the statistics of the sampled time-series movement, i.e., the mean in-

Table 3 Experimental results.

Cost scenario	Number of Products	(c_v, ρ)	can-order policy	MP policy	MARL(ours)	% improved
1) Base	2	(0.2,0)	95.0 ± 0.9	99.2 ± 0.8	95.3 ± 0.7	−0.4%
		(0.6,0)	116.5 ± 2.1	125.3 ± 2.4	115.3 ± 2.3	1.0%
	5	(0.2,0)	76.6 ± 0.6	76.6 ± 0.5	73.6 ± 1.2	3.9%
		(0.6,0)	87.8 ± 0.7	89.1 ± 1.2	87.0 ± 0.8	0.9%
	10	(0.2,0)	173.8 ± 1.7	169.0 ± 1.5	171.6 ± 0.2	−1.6%
		(0.6,0)	210.7 ± 2.7	209.4 ± 2.0	210.5 ± 1.7	−0.5%
2) Capacitated Transportation	2	(0.2,0)	98.4 ± 1.3	102.5 ± 2.5	97.1 ± 0.6	1.4%
		(0.6,0)	119.0 ± 2.0	129.4 ± 5.7	118.1 ± 1.2	0.7%
	5	(0.2,0)	78.1 ± 4.7	77.0 ± 0.3	72.9 ± 0.7	5.3%
		(0.6,0)	87.5 ± 0.7	89.0 ± 1.3	87.1 ± 0.5	0.5%
	10	(0.2,0)	192.5 ± 8.3	189.3 ± 0.6	178.5 ± 1.2	5.7%
		(0.6,0)	229.3 ± 6.0	225.5 ± 1.3	219.1 ± 1.3	2.9%
3) Stepwise Transportation	2	(0.2,0)	507.5 ± 7.1	503.3 ± 4.3	506.5 ± 2.4	−0.6%
		(0.6,0)	728.8 ± 16.2	718.6 ± 7.6	715.8 ± 12.8	0.4%
	5	(0.2,0)	460.0 ± 4.6	479.2 ± 25.6	444.0 ± 3.6	3.5%
		(0.6,0)	611.1 ± 8.5	618.3 ± 12.7	607.7 ± 4.4	0.5%
	10	(0.2,0)	918.6 ± 23.4	893.6 ± 3.7	751.7 ± 3.1	15.9%
		(0.6,0)	1126.4 ± 28	1108.9 ± 24.9	1014.2 ± 12.8	8.5%
4) Nonlinear Holding	2	(0.2,0)	90.2 ± 1.6	91.8 ± 0.4	90.1 ± 0.5	0.1%
		(0.6,0)	103.5 ± 0.8	110.6 ± 2.0	104.0 ± 1.2	−0.5%
	5	(0.2,0)	75.0 ± 0.4	74.3 ± 0.3	75.4 ± 0.3	−1.5%
		(0.6,0)	81.5 ± 0.5	83.0 ± 1.3	82.2 ± 0.8	−0.9%
	10	(0.2,0)	152.5 ± 2.8	148.6 ± 1.3	149.6 ± 2.4	−0.6%
		(0.6,0)	191.1 ± 2.5	188.4 ± 2.1	190.1 ± 1.1	−0.9%

We conducted 6 times learning with different initialization both for benchmark and our proposed agent. In each run, results were derived by calculating the averaged total cost over 12 simulations with different seeds in demand generation. The value in (±·) represents the standard deviation over 6 runs. % improved represents the gap in performance of our proposed agent and better performance of the benchmark policies.

Table 4 Statistics of the sampled time-series movement of the learned policies.

			can-order policy			MP policy			MARL (ours)		
Cost scenario	Number of Products	(c_v, ρ)	Inv.	Load	Repl.	Inv.	Load	Repl.	Inv.	Load	Repl.
1) Base	2	(0.2,0)	16.5	-	23.1	16.8	-	23.6	14.6	-	21.1
		(0.6,0)	19.5	-	22.6	20.8	-	21.6	17.0	-	17.9
	5	(0.2,0)	14.4	-	20.2	14.6	-	20.5	10.7	-	13.7
		(0.6,0)	16.4	-	18.3	15.7	-	18.3	13.7	-	13.8
	10	(0.2,0)	35.6	-	39.8	33.8	-	37.9	23.2	-	18.8
		(0.6,0)	41.2	-	34.0	38.7	-	30.5	30.7	-	18.6
2) Capacitated Transportation	2	(0.2,0)	15.2	93%	18.6	13.7	85%	17.0	12.7	81%	16.2
		(0.6,0)	19.0	94%	18.8	20.1	77%	15.4	17.1	84%	16.7
	5	(0.2,0)	13.5	91%	18.2	13.3	83%	16.7	10.5	69%	13.8
		(0.6,0)	15.4	83%	16.6	15.0	81%	16.2	13.4	65%	13.1
	10	(0.2,0)	33.1	91%	18.4	26.0	77%	15.3	23.5	79%	16.7
		(0.6,0)	37.5	86%	17.6	32.5	77%	15.4	32.0	76%	16.3
3) Stepwise Transportation	2	(0.2,0)	59.0	97%	30.1	60.4	99%	29.9	54.5	98%	30.2
		(0.6,0)	92.2	93%	37.2	93.9	97%	38.3	68.1	97%	35.4
	5	(0.2,0)	51.2	91%	23.8	65.6	94%	51.6	48.8	86%	24.1
		(0.6,0)	81.1	85%	24.0	93.7	92%	49.8	74.0	84%	22.5
	10	(0.2,0)	213.3	90%	93.2	127.2	89%	114.8	85.2	82%	39.5
		(0.6,0)	163.1	87%	59.3	165.6	90%	105.3	110.8	83%	44.4
4) Nonlinear Holding	2	(0.2,0)	18.2	-	23.7	18.9	-	23.8	18.0	-	24.9
		(0.6,0)	21.7	-	22.0	23.1	-	23.6	20.3	-	23.0
	5	(0.2,0)	15.9	-	20.5	16.1	-	21.6	16.7	-	23.6
		(0.6,0)	18.3	-	20.0	18.2	-	19.6	18.1	-	19.5
	10	(0.2,0)	32.9	-	32.1	33.6	-	37.9	28.2	-	23.0
		(0.6,0)	41.1	-	31.2	39.8	-	31.8	34.8	-	22.4

Time-series statistics from the simulated results following the learned policies are presented. Since we conducted six times learning, we simulated one episode per each result and average of the six simulations was calculated. Inv., Load, and Repl. indicate the mean inventory volume, the mean loading ratio, and the mean replenishment quantity, respectively.

ventory level, mean loading ratio, and mean replenishment quantity, with which we can examine the characteristics of the learned policy behavior particularly focusing on how the learned policies handle the trade-off between transportation and holding costs.

Since the demand correlation among products did not have a significant effect on the performance gap in this study, only the results for $\rho = 0$ is presented. For reference, please see **Table 5**, which represents the result of the case with two products with non-zero demand correlation.

In terms of the comparison between the can-order and MP policies, when the number of products is small, we can see that the performance of the MP policy is worse, particularly for problems with high demand deviation, which is consistent with the literature. However, for problems with 10 products, the performance of the MP policy is equal to or higher than that of the can-order policy for all the cost scenarios.

As for the performance of our proposed agent, we evaluated our performance by comparing with the better one of the can-

Table 5 Experimental results with demand correlation for a two-product case.

Cost scenario	(c_v, ρ)	can-order policy	MP policy	MARL (ours)	% improved
1) Base	$(0.2, -0.5)$	95.8 ± 1.6	99.0 ± 1.0	95.5 ± 0.6	0.3%
	$(0.2, 0)$	95.0 ± 0.9	99.2 ± 0.8	95.3 ± 0.7	−0.4%
	$(0.2, 0.5)$	95.8 ± 1.9	98.7 ± 1.3	94.5 ± 0.6	1.4%
	$(0.6, -0.5)$	116.4 ± 0.9	123.7 ± 2.8	117.3 ± 2.0	−0.7%
	$(0.6, 0)$	116.5 ± 2.1	125.3 ± 2.4	115.3 ± 2.3	1.0%
	$(0.6, 0.5)$	115.2 ± 1.4	123.1 ± 1.7	114.0 ± 1.8	1.0%
2) Capacitated Transportation	$(0.2, -0.5)$	98.7 ± 1.3	105.2 ± 6.9	97.6 ± 0.4	1.1%
	$(0.2, 0)$	98.4 ± 1.3	102.5 ± 2.5	97.1 ± 0.6	1.4%
	$(0.2, 0.5)$	98.4 ± 0.9	101.4 ± 2.0	96.5 ± 0.8	1.9%
	$(0.6, -0.5)$	120.0 ± 2.0	125.3 ± 4.8	118.3 ± 1.3	1.5%
	$(0.6, 0)$	119.0 ± 2.0	129.4 ± 5.7	118.1 ± 1.2	0.7%
	$(0.6, 0.5)$	117.0 ± 2.2	131.9 ± 5.0	116.2 ± 1.1	0.6%
3) Stepwise Transportation	$(0.2, -0.5)$	516.6 ± 9.1	506.9 ± 5.5	506.5 ± 2.7	0.1%
	$(0.2, 0)$	507.5 ± 7.1	503.3 ± 4.3	506.5 ± 2.4	−0.6%
	$(0.2, 0.5)$	500.8 ± 3.2	500.3 ± 4.5	500.2 ± 2.2	0.0%
	$(0.6, -0.5)$	733.0 ± 5.5	718.8 ± 8.0	716.1 ± 12.2	0.4%
	$(0.6, 0)$	728.8 ± 16.2	718.6 ± 7.6	715.8 ± 12.8	0.4%
	$(0.6, 0.5)$	729.8 ± 14.8	715.4 ± 23.4	708.5 ± 3.6	1.0%
4) Nonlinear Holding	$(0.2, -0.5)$	90.1 ± 0.5	91.8 ± 1.1	90.0 ± 0.4	0.1%
	$(0.2, 0)$	90.2 ± 1.6	91.8 ± 0.4	90.1 ± 0.5	0.1%
	$(0.2, 0.5)$	89.3 ± 0.3	91.9 ± 0.7	90.0 ± 0.6	−0.8%
	$(0.6, -0.5)$	103.7 ± 1.1	109.3 ± 2.0	104.0 ± 0.9	−0.3%
	$(0.6, 0)$	103.5 ± 0.8	110.6 ± 2.0	104.0 ± 1.2	−0.5%
	$(0.6, 0.5)$	102.6 ± 1.8	110.2 ± 1.6	102.6 ± 1.2	0.1%

order and MP policies.

For the base cost scenario with fixed joint replenishment cost, the performance of our proposed agent was almost equal to that of the better one of the benchmark policies irrespective of the demand deviation and number of products. From Table 4, it can be seen that although the gap of the total cost is not so significant, the learned policy has different characteristics; the mean inventory level for the result of our proposed agent is lower than that of the benchmark policies, whereas the mean replenishment quantity of the benchmark policies is higher than ours, particularly when the number of products becomes large.

For the scenarios of capacitated and stepwise transportation, our proposed agent performed better, and the difference in performance increased as the number of products increased. This improvement in the performance indicates that the benchmark policies are not well suited for the cases with transportation capacity. To take into account the transportation capacity, we employed the additional heuristic presented in Section 4.2. However, this two-step process (GA parameter optimization and the loading ratio adjustment) could not always find a good solution (see further investigation regarding the two-step process in Section 4.3.3). From Table 4, similar to the observation for the base cost scenario, it can be seen that the inventory level of our result is lower than that of the can-order policy, whereas the loading ratio of the result for the can-order policy is higher than that of ours. This indicates that our proposed solution was able to strike a balance between the holding and transportation costs better than the benchmark two-step process.

Conversely, we also see that when the demand deviation is high ($c_v = 0.6$), the performance improvement against the benchmark policy is limited. This can be due to the high variance of the obtained reward in our RL approach. Another possible cause lies in the sequential search procedure in our joint-action selection heuristic. Because the second or later searched product is randomly selected in our sequential search in Algorithm 2, the selected joint-action may change, even if the input state and the action value function parameters are the same, i.e., the selected joint-action can be sub-optimal. The effect of random search in the sequential search procedure is considered to increase with high-demand deviation.

For the nonlinear holding cost scenario, the performance of our agent was slightly lower than that of the better one of the benchmark policies. However, the gap was insignificant when the standard deviation over six runs was considered. From the experimental results, we can deduce that the benchmark policies were able to handle the nonlinear holding cost without any additional procedure when the policy parameters were optimized using GA, and the necessity of devising the new solution for the nonlinear holding cost function is relatively low.

4.3.2 Evaluation of the Action Space Setting

In our experiments, we limited the cardinality of each agent's action space $|\mathcal{A}_i|$ to six. With an extremely small action space, the optimal solution may not be included. Conversely, with an extremely large action space, our algorithm may not converge or may need longer episodes to converge. Thus, we conducted additional experiments for the base cost scenario with $c_v = 0.2$ with increased action space to evaluate the effect on the performance with larger action space setting. It should be noted that the maximum selected action among $\{0, 1, 2, 3, 4, 5\}$, meaning the multiplier of the lot size, of all the products in our simulation with $|\mathcal{A}_i| = 6$ was 3, 4, and 2 for the base cost scenario with 2, 5, and 10 product cases, respectively. This indicates that $|\mathcal{A}_i| = 6$ for all i is sufficiently large.

From **Table 6**, it can be seen that the performance with $|\mathcal{A}_i| = 10$ and $|\mathcal{A}_i| = 20$ is almost equal to the case with $|\mathcal{A}_i| = 6$ for two-product and five-product cases. However, for ten-product case, slight drop in performance was observed. This indicates the necessity to adjust the number of learning episodes depending on the cardinality of each agent's action space and the number of products.

Table 6 Evaluation of the action space setting.

Number of Products	$\|\mathcal{A}_i\| = 6$	$\|\mathcal{A}_i\| = 10$	$\|\mathcal{A}_i\| = 20$
2	95.3	94.9	95.6
5	73.6	72.9	72.9
10	171.6	173.0	175.6

The average total costs for the base cost scenario with $c_v = 0.2$ are presented. Experiments with larger action space were conducted under the same condition with the case with $|\mathcal{A}_i| = 6$

Table 7 Evaluation of the effectiveness of benchmark methodology.

			Loading Adjst.	
policy	c_v	GA	w/o	w/
can-oder	0.2	w/o	493.2	448.7
		w/	468.3	460.0
	0.6	w/o	668.3	629.0
		w/	617.0	611.1
MP	0.2	w/o	536.6	508.0
		w/	489.3	479.2
	0.6	w/o	692.1	663.7
		w/	626.8	618.3

The average total costs for the five-product stepwise transportation scenario are presented. Loading adjst. indicates whether or not to apply the loading ratio adjustment procedure.

4.3.3 Evaluation of the Benchmark Methodology

For the capacitated transportation and stepwise transportation cost scenario, we employed the additional heuristic, described in Algorithm 3, to consider the transportation capacity. Our benchmark methodology consisted of two steps; 1) use GA to find the better parameters of the can-order and MP policies, and 2) apply the procedure of loading ratio adjustment. To individually evaluate the effectiveness of both steps, we conducted additional experiments with fixed parameters (without GA optimization) and/or without loading ratio adjustment for five-product stepwise transportation scenario.

With regard to the fixed parameters, the predetermined must-order level, s_i, of each item is set to the mean three-step demand plus 99.9% safety stock, which can be expressed as $3\mu_i + 3.1\sigma_i\sqrt{\mathrm{LT}}$. The predetermined can-order level, c_i, of each item is set to s_i plus one-step mean demand, and the order-up-to level, S_i, of each item is set to s_i plus the two-step mean demand. The global parameter indicating the frequency of the replenishment for the MP policy is set to one.

Table 7 presents the result. With regard to the effectiveness of the GA parameter optimization, we can see that the cost without GA is higher than that with GA, thus reflecting the improvement by applying the parameters derived by GA (row-wise comparison in Table 7 without the loading adjustment procedure). With regard to the loading ratio adjustment procedure, the total cost decreased for all the cases (column-wise comparison in Table 7) irrespective of the GA optimization.

However, it can also be seen that, for the can-order policy with $c_v = 0.2$, the cost of policy with GA and with loading adjustment procedure (460.0) is higher than that without GA and with loading adjustment procedure (448.7). This indicates that the two-step optimization could not converge to the optimal solution, and is considered to reflect the difficulty in extending the existing policies, including the can-order and MP policies, for non-base cost scenarios.

Table 8 Computation time comparison.

Number of Products	GA (can-order)	MARL (ours)
2	23 (23)	85
5	143 (57)	235
10	540 (108)	520

The computation time is expressed in minutes. To make a fair comparison, $(\cdot)$ for GA denotes the adjusted time, taking into account the different numbers of generations with respect to the number of products.

4.3.4 Computation Time

We also analyze the computation time both for our proposed RL-based approach and GA optimization approach. Since the actual computation time depends on the setting of the number of generations and the individuals of each population for GA, and the number of episodes for the RL-based approach, we focus on the analysis of the change in the computation time with respect to the number of products rather than the absolute computation time comparison.

Table 8 presents the average computation time. In our MARL approach, the same number of episodes is set, regardless of the number of products. Conversely, in our GA setting, we utilized the different numbers of population. Therefore, to make a fair comparison, a virtual computation time, dividing the actual time by $N/2$, is presented for GA.

The experiments were conducted on the same computer, and any GPU has not been used. It should be noted that the use of a GPU did not improve the computation time for our proposed solution. This is because the architecture of the neural network of the action value function is simple (see Section 4.4 for further information).

We can see that, for both the MARL and GA solutions, the computation time is almost linear with respect to the number of products when setting the same condition, i.e., the number of populations and generations for GA, and the number of episodes in MARL. In our implementation of MARL, any multi-core processing is not employed. However, the MARL solution can be easily sped up *via* parallel computing, since most of the MARL learning time is spent on the forwarding and backwarding of the neural network of each agent's action value function, and these process can be parallelized per each agent.

In our formulation, since the information of other products is summarized and fed into each agent's network as a scalar input, the network size of each agent would not change with respect to the number of products. Thus, if we have plenty of CPUs, the computation time would only increase sub-linearly with respect to the number of products.

It should be noted that speeding up the parameter optimization of GA is also possible *via* multi-processing in the evaluation of the each solution, and/or employing the gray coding [1], [10] instead of binary coding.

4.4 Experiment Detail

4.4.1 Cost Parameter Setting

We let the cost parameters U^{hold}, U^{pel}, and U^{trans} be 0.02, 1.0, and 1, respectively. These are set based on the typical logistic condition. With regard to the nonlinear holding cost scenario, the fixed holding cost is set to $U^{\mathrm{hold_f}} = 0.7 \times U^{\mathrm{hold_v}} \times \mathrm{CAP_{wh}}$, where

$U^{\mathrm{hold}_v} = U^{\mathrm{hold}}$.

4.4.2 Multi-agent Reinforcement Learning Setting

The network that represents state-action value function had three hidden layers with 64, 32, and 32 units. We used the Adam optimizer. A mini-batch size was 32 and a discount factor was 0.995. We used ReLu for all hidden layers and linear activation on the output layers. A target network was updated every 10 episodes.

One of the biggest challenges in multi-agent reinforcement learning is the non-stationary environment caused by the concurrent learning. To stabilize the learning process, we used the hysteretic learning [13], [19], where different learning rate is applied depending on whether the calculated loss is positive or negative. In our experiments, we used the 0.4x. learning rate for the negative loss. Also, the replay memory size with memory capacity was set to 10,000 so that the obsolete memory would not be used for learning.

5. Conclusion

Our numerical experiments demonstrate that the performance of our proposed agent is almost equal to that of the better one of the benchmark policies irrespective of the demand characteristics for the base cost scenario. In addition, our proposed agent outperformed the benchmark policies for cases with transportation capacity constraint or stepwise transportation cost.

Moreover, we show the limitation of extending the existing policies for the non-base cost scenario just by incorporating the additional heuristic procedure.

Since our approach does not rely on the knowledge on the state transition probability and reward functions, incorporating the new cost structure can be done just by modifying the cost setting in our simulator. There are many variants of cost settings in practice, and existing studies have not covered all the conditions. We hope our solution can be of help in complicated practical conditions.

However, for cases with high-demand deviation, the performance improvement was limited. For future study, this should be investigated from the perspective of the high variance of reward in the RL and the sequential search procedure in the joint-action selection heuristics.

One of the limitations of our study is the versatility of cost structures. For future study, incorporating other supply chain conditions, such as volume discount offered by a supplier, and the combination of several cost settings including those examined in our study, would be an interesting topic. Since our solution uses the heuristic search procedure, the validity of the heuristic procedure with different cost structure is worth investigating.

Our proposed solution has many parameters, which may not be acceptable for some companies at this moment, because the derived policy is not easily understandable for people. However, it is also true that in the logistics industry, huge loss is being caused by allowing people in operation to adjust the parameters by hand, but ending up leaving those parameters not updated. We believe that, in the near future, operations in inventory management will become fully automated, and complexity of replenishment policy like ours may not be a problem.

References

[1] Andre, J., Siarry, P. and Dognon, T.: An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization, *Advances in Engineering Software*, Vol.32, No.1, pp.49–60 (2001).

[2] Atkins, D.R. and Iyogun, P.O.: Periodic Versus "Can-Order" Policies for Coordinated Multi-Item Inventory Systems, *Management Science*, Vol.34, No.6, pp.791–796 (online), DOI: 10.1287/mnsc.34.6.791 (1988).

[3] Balintfy, J.L.: On a Basic Class of Multi-Item Inventory Problems, *Management Science*, Vol.10, No.2, pp.287–297 (online), DOI: 10.1287/mnsc.10.2.287 (1964).

[4] Ben-Khedher, N. and Yano, C.A.: The Multi-Item Joint Replenishment Problem with Transportation and Container Effects, *Transportation Science*, Vol.28, No.1, pp.37–54 (online), DOI: 10.1287/trsc.28.1.37 (1994).

[5] Buşoniu, L., BabuŠka, R. and Schutter, B.: A Comprehensive Survey of Multiagent Reinforcement Learning, *IEEE Trans. Systems Man and Cybernetics-Part C: Applications and Reviews*, Vol.38, No.2, pp.156–172 (online), DOI: 10.1109/TSMCC.2007.913919 (2008).

[6] dos Bastos, L., Mendes, M., de Nunes, D., Melo, A., Carneiro, M., do de Janeiro, B., Vargas, B. and do do Pará, B.: A systematic literature review on the joint replenishment problem solutions: 2006-2015, *Prod*, Vol.27 (online), DOI: 10.1590/0103-6513.222916 (2017).

[7] Feng, H., Wu, Q., Muthuraman, K. and Deshpande, V.: Replenishment Policies for Multi–Product Stochastic Inventory Systems with Correlated Demand and Joint–Replenishment Costs, Vol.24, No.4, pp.647–664 (online), DOI: 10.1111/poms.12290 (2015).

[8] Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P.H.S., Kohli, P. and Whiteson, S.: Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning, Precup, D. and Teh, Y.W. (Eds.), *Proc. Machine Learning Research*, Vol.70, pp.1146–1155 (2017) (online), available from ⟨http://proceedings.mlr.press/v70/foerster17b.html⟩.

[9] Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M. and Gagné, C.: DEAP: Evolutionary Algorithms Made Easy, *Journal of Machine Learning Research*, Vol.13, pp.2171–2175 (2012).

[10] Frank, G.: Pulse code communication, US Patent 2,632,058 (1953).

[11] Johansen, S. and Melchiors, P.: Can-order policy for the periodic-review joint replenishment problem, Vol.54, No.3, pp.283–290 (online), DOI: 10.1057/palgrave.jors.2601499 (2003).

[12] Kiesmüller, G.: Multi-item inventory control with full truckloads: A comparison of aggregate and individual order triggering, *European Journal of Operational Research*, Vol.200, No.1, pp.54–62 (online), DOI: 10.1016/j.ejor.2008.12.008 (2010).

[13] Matignon, L., Laurent, G.J. and Le Fort-Piat, N.: Hysteretic Q-learning: An algorithm for Decentralized Reinforcement Learning in Cooperative Multi-Agent Teams, *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.64–69 (2007).

[14] Minner, S. and Silver, E.A.: Multi-product batch replenishment strategies under stochastic demand and a joint capacity constraint, Vol.37, No.5, pp.469–479 (online), DOI: 10.1080/07408170590918254 (2005).

[15] Minner, S. and Silver, E.A.: Replenishment policies for multiple products with compound-Poisson demand that share a common warehouse, Vol.108, No.1-2, pp.388–398 (online), DOI: 10.1016/j.ijpe.2006.12.028 (2007).

[16] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. and Nature, V.J.: Human-level control through deep reinforcement learning, *Nature*, (online), DOI: 10.1038/nature14236 (2015).

[17] Ohno, K. and Ishigaki, T.: A multi-item continuous review inventory system with compound Poisson demands, *Math Method Oper Res*, Vol.53, No.1, pp.147–165 (online), DOI: 10.1007/s001860000101 (2001).

[18] Ohno, K., Ishigaki, T. and Yoshii, T.: A new algorithm for a multi-item periodic review inventory system, *Zeitschrift Für Operations Res*, Vol.39, No.3, pp.349–364 (online), DOI: 10.1007/bf01435462 (1994).

[19] Omidshafiei, S., Pazis, J., Amato, C., How, J.P. and Vian, J.: Deep decentralized multi-task multi-agent reinforcement learning under partial observability, *JMLR. org*, pp.2681–2690 (2017).

[20] Pyke, D., Peterson, R. and Silver, E.: Inventory Management and Production Planning and Scheduling (3rd Edition), *Journal of The Operational Research Society - J OPER RES SOC*, Vol.52, pp.845–845 (2001).

[21] Suetsugu, H., Narusue, Y. and Morikawa, H.: Branching Deep Q-Network Agent with Reward Allocation Mechanism for Joint Replenishment Policy, *IPSJ Trans. Mathematical Modeling and Its Applications*, Vol.13, No.2, pp.36–49 (2020).

[22] Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents, *Proc. 10th International Conference on Machine Learning*, pp.330–337 (1993).

[23] Tavakoli, A., Pardo, F. and Kormushev, P.: Action branching architectures for deep reinforcement learning, *32nd AAAI Conference on Artificial Intelligence*, pp.4131–4138 (2018).

[24] Viswanathan, S.: Note. Periodic Review (s, S) Policies for Joint Replenishment Inventory Systems, *Management Science*, Vol.43, No.10, pp.1447–1454 (online), DOI: 10.1287/mnsc.43.10.1447 (1997).

Hiroshi Suetsugu received his B.E., and M.E. degrees from the University of Tokyo in 2006 and 2008, respectively. His current research interests are sequential decision making, time series analysis, and peer-to-peer electricity trading.

Yoshiaki Narusue received his B.E., M.E., and Ph.D. degrees from the Graduate School of Information Science and Technology, the University of Tokyo, Tokyo, Japan, in 2012, 2014, and 2017, respectively. Currently, he is a research associate with the Department of Electrical Engineering and Information Systems, the University of Tokyo, Tokyo, Japan. He received the second-best student paper award at IEEE Radio and Wireless Symposium in 2013, Hiroshi Harashima academic encouragement award in 2013, the best paper award at IEEE Consumer Communications and Networking Conference in 2018, and ACM IMWUT distinguished paper award in 2020. His research interests include wireless power transfer, next-generation wireless communication systems, and the Internet of Things. He is a member of the IEEE, IEICE, and IPSJ.

Hiroyuki Morikawa received his B.E., M.E. and Dr. Eng. degrees in electrical engineering from the University of Tokyo, Tokyo, Japan, in 1987, 1989, and 1992, respectively. Since 1992, he has been in the University of Tokyo and is currently a full professor of School of Engineering at the University of Tokyo. From 2002 to 2006, he was a group leader of the NICT Mobile Networking Group. His research interests are in the areas of IoT/M2M/big data, sensor networks, wireless communications, and digital society design. He served as a technical program committee chair of many IEEE/ACM conferences and workshops, Vice President of IEICE, OECD Committee on Digital Economy Policy (CDEP) vice chair, Director of New Generation IoT/M2M Consortium, and he sits on numerous telecommunications advisory committees and frequently serves as a consultant to government and companies. He has received more than 70 awards including three IEICE best paper awards, IPSJ best paper award, JSCICR best paper award, Info-Communications Promotion Month Council President Prize, NTT DoCoMo Mobile Science Award, Rinzaburo Shida Award, and Radio Day Ministerial Commendation.