Periodic-Review Joint Replenishment Policy using Multi-Agent Reinforcement Learning

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

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 there are many types of joint-replenishment cost structures 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 heuristic approach. The purpose of this paper is to propose a multi-agent reinforcement learning-based solution for a joint replenishment problem, 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 is equal or greater than that of the existing heuristic policies, that 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) 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. Recent increase in e-commerce have made the smaller companies participate in the international transportation using trucks or container ships, where the practical consideration of JRP under stochastic demands is needed. It is because the deviation of demand as well as the ratio of the shared cost over the total supply chain cost are typically high.

There have been many studies in JRPs, and most studies have considered the deterministic demands, wheres the studies on stochastic demands have been limited[2]. With stochastic demands, the Markov decision processes (MDP) have been employed for the formulation of the problem. When the number of products is less than four, the MDP can be only solved because the action spaces grow exponentially with the number of products because of its combinatorial nature [11], [12].

Thus, several heuristic algorithms have been proposed for a large number of products. However, there are some studies that have reported the pros and cons of each class of policy depending on the demand characteristics, e.g., demand variation or correlation among products. For example, for the well-known coordinated policy, called can-order, or (s, c, S) policy, its performance would be worse as compared to the optimal solution for two product case, when the demand correlation among products exists [3]. Another policy, called modified periodic (MP) policy, has been reported to achieve good performance as compared to the can-order policy. However, some studies [5] pointed out that the MP policy could not outperform the can-order policy for problems with high demand variation.

In addition, most studies have assumed the fixed replenishment cost, whereas there are many types of cost structures in practice, e.g., capacity constraint or stepwise cost for container or truck shipment, non-linear cost for warehouse costs, etc. Some studies [6], [8], [9] 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.

Earlier, a reinforcement learning based agent has been proposed, called the branching deep-Q network with reward allocation [14]. The earlier approach has the distinguishing feature of the shared representation of state-action value function followed by the product-independent branches with credit assignment mechanism, with the aim of encouraging the cooperative behavior among agents, while enabling the linear growth of the total number of network outputs with respect to the number of products. However, that could not outperform the existing heuristic policy when the number of products became large.

With this in mind, the purpose of this paper is to propose a multi-agent reinforcement learning-based solution for a JRP that, unlike existing approach including can-order and modified periodic policies, does not assume specific class of policies. Our solution can be used for problems with several demand settings, in terms of the number of products, demand variation, and the demand correlation among products, and be applied for various

[1] Graduate School of Engineering, The University of Tokyo, 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

cost structures with minor modification, including the capacitated transportion, stepwise transportation costs, and non-linear holding costs.

We employ the reinforcement learning (RL), which does not require the knowledge on the state transition probability and reward function, with the aim of making our solution applicable for 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 consists of each product. The distinguishing requirement for the solution of JRP is *cooperation* among products, and in order for our agent to be *cooperative*, instead of taking an action independently among agents, we introduce the central joint-action selection unit that decides the joint-action of all the agents simultaneously, with each agent learning the state-action value function which is conditioned by the other agents' actions. In the central joint-action selection unit, we employ the heuristic joint-action search procedure which efficiently searches the joint-action for larger total state-action value among all the agents. Our heuristic procedure works only when coupled with the proper credit assignment strategy, with which joint costs among products are allocated to each agent.

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

2. Method

2.1 Problem setting

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

i : Item number, $i = 1, ..., N$,
t : Period, $t = 1, ..., T$,
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 take supplier stock-out or any supply delay into consideration. 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.

$$In_{i,t+LT_i} = x_{i,t}, \tag{1}$$
$$Out_{i,t} = \min(d_{i,t}, I_{i,t}), \tag{2}$$
$$I_{i,t+1} = I_{i,t} - Out_{i,t} + In_{i,t}, \tag{3}$$
$$I^p_{i,t+1} = I^p_{i,t} - Out_{i,t} + x_{i,t}, \tag{4}$$
$$u_{i,t} = d_{i,t} - Out_{i,t}. \tag{5}$$

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

2.1.1 Cost structures

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

$$C^{\text{pel}}_{i,t} = U^{\text{pel}} \times u_{i,t}. \tag{6}$$

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;

$$C^{\text{trans}}_t = U^{\text{trans}} \ (\text{if} \sum_i x_{i,t} > 0), \tag{7}$$
$$C^{\text{hold}}_{i,t} = U^{\text{hold}} \times I_{i,t}. \tag{8}$$

The second type is the capacitated replenishment cost, where a certain capacity of transportation ($\text{CAP}_{\text{trans}}$) is taken into consideration ($\sum_i x_{i,t} \leq \text{CAP}_{\text{trans}}$), 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;

$$C^{\text{trans}}_t = U^{\text{trans}} \times \lceil \frac{\sum_i x_{i,t}}{\text{CAP}_{\text{trans}}} \rceil. \tag{9}$$

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

$$C^{\text{hold}}_t = U^{\text{hold}_{\text{f}}} + U^{\text{hold}_{\text{v}}} \times max(0, \sum_i I_{i,t} - \text{CAP}_{\text{wh}}), \tag{10}$$

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

Tab. 1 summarizes the cost setting above.

Table 1 Cost scenario

Cost scenario	Transportation cost	Inventory cost
1) Base	Fixed	Linear
2) Capacitated (trans)	Fixed (with CAP constraint)	Linear
3) Stepwise (trans)	Stepwise	Linear
4) Non-linear (hold)	Fixed	Non-linear

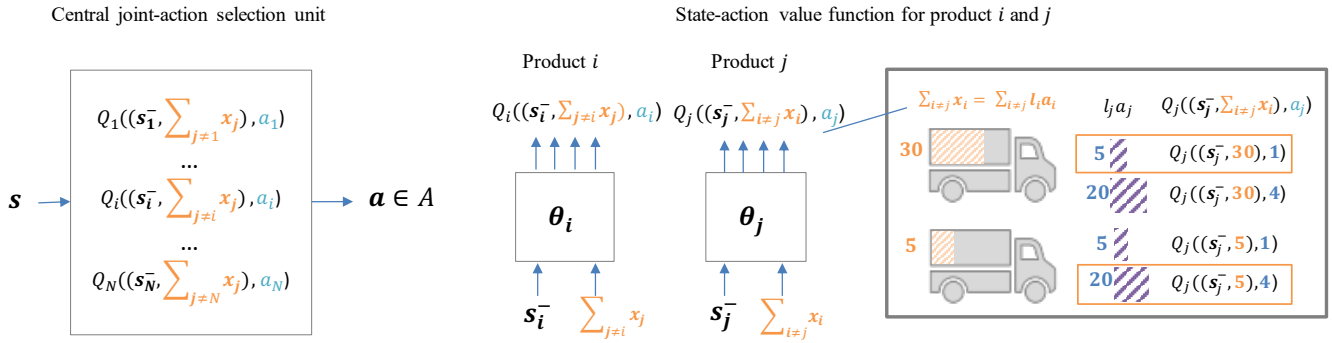

Fig. 1 each agent's state-action value function conditioned by sum of other agent's order quantities

2.2 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, that is, $s_t = [(I_{i,t}, I^p_{i,t})]^N_{i=1}$, makes a decision $a_t = (a_0, a_1, ...a_n) \in \mathcal{A}$ based on s_t, and receives a 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 :

$$Q(s_t, a_t) = (1-\alpha_t)\, Q(s_t, a_t) + \alpha_t \left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right), \quad (11)$$

However, the Q-learning above (or function-approximated Q-learning, e.g., deep Q-Network [10]) cannot converge due to the increasing action spaces when the number of products becomes large.

2.3 Multi-agent formulation

In order to handle the exponentially increasing action space with respect to the number of products, we employed the multi-agent approach. The simplest way to formulate as a multi-agent problem is the independent Q-learning (IQL)[15], where each agent treats other agents' action as part of the environment, that is;

$$Q_i(s_{i,t}, a_{i,t}) = (1-\alpha_t)\, Q_i(s_{i,t}, a_{i,t}) + \alpha_t \left(r_{i,t+1} + \gamma \max_{a_i} Q(s_{i,t+1}, a_i) \right), \quad (12)$$

where only difference from the eq. 11 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 fail 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 [4].

The earlier work [14] was extended version of IQL that has shared representation with the aim of encouraging the coordinated behavior among agents. Although it could learn the coordinated ordering behavior, the performance was not better when compared to the existing coordinated heuristic policies like can-order and periodic review policies. Thus, we started attempting to incorporate the central control unit that explicitly considers all the agents' actions.

Fig. 1 represents our proposed solution. Each agent consists of 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 [14] is that we treat the information of other agents' actions as a part of a state in the state-action value function. Thus, the state-action value function of each agent is conditioned by the other agents' action rather than selecting an action independently. We also have a central joint-action selection unit, where the joint-action is decided based on the Q-value function of all the agents.

When the inventory cost is proportionate to the inventory quantity (in the cost scenario 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 $s^-_{i,t} = (I_{i,t}, I^p_{i,t})$ and $x_{j,t} = l_j a_{j,t}$ is the order quantity of the agent j when selecting an action a_j. In case of non-linear inventory cost (in the cost scenario 4), other products inventory level should be taken into consideration. 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 $s_{i,t+1}$. The explanation of the joint-action selection, credit assignment, and the ϵ-greedy exploration strategy are provided from the following section. Since an infinite action space is not practical, and taking a large number of orders as compared with the demand is unrealistic from a supply chain point of view, 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, joint-action space of all the products would be 6^n. Each agent's state-action value with function approximation is updated according to the following loss:

$$L_i = \mathbb{E}_{(s_i, a_i, r_i, s'_i) \sim \mathcal{D}_i} \left[L_\delta \left(y_i, Q_i(s_i, a_i) \right) \right], \quad (13)$$

where

$$y_i = r_i + \gamma Q^-_i(s'_i, argmax_{a_i} Q_i(s'_i, a_i)), \quad (14)$$

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

2.3.1 Credit assignment

Since there are several agents in an environment, a credit assignment should be decided that satisfies $\sum_i r_{i,t}$ equals the nega-

tive sum of the transportation, holding, and penalty cost at time t. As for the transportation cost, we employed the global reward allocation, where transportation cost C_t^{trans} is equally allocated to each product, and holding cost can be calculated independently per product (Eq. 8) for the linear inventory cost scenario (cost scenario 1, 2, and 3), that is;

$$r_{i,t+1} = -(C_t^{\text{trans}}/N + C_{i,t}^{\text{hold}} + C_{i,t}^{\text{pel}}). \quad (15)$$

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

When the inventory cost is also a joint-cost among products in cost scenario 4 (Eq. 9), we employed the local and global reward allocation for the variable and the fixed part of the holding cost, respectively. Thus, allocated reward for each agent is defined as follows;

$$r_{i,t+1} = -(C_t^{\text{trans}}/N + U^{\text{hold}_{\text{f}}}/N + C_t^{\text{hold}_{\text{f}}} \times \frac{I_{i,t}}{\sum_i I_{i,t}} + C_{i,t}^{\text{pel}}), \quad (16)$$

where $C_t^{\text{hold}_{\text{f}}} = C_t^{\text{hold}} - U^{\text{hold}_{\text{f}}}$.

2.3.2 Joint ϵ-greedy exploration strategy

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

2.3.3 Joint-action selection

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

$$\boldsymbol{a} = (a_0, a_1, ...a_n) = argmax_{\boldsymbol{a}\in A} \sum_i Q_i((s_i^-, \sum_{j\neq i} l_j a_j), a_i). \quad (17)$$

We, however, still face a combinatorial action selection problem. In the learning process, we need to solve this optimization problem per each step. Thus, the efficient heuristics for the Eq. 17 is needed.

2.3.4 Joint-action selection heuristics

With the availability of the state-action value function of each agent, each agent decision can be modified 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. The illustration of the procedure is provided in Fig. 2.

Here, we utilize the characteristics of the learnt behavior depending on the credit assignment. As reported in the study [14], the agent with 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, $a = (0, 0, \cdots, 0)$, coordinated canceling opportunity is examined every time.

If we employed the local reward allocation, where a joint-transportation cost is allocated in proportion to the order quantities, incremental process should not work because for an agent given $\sum x_{j\neq i} = 0$, all the joint-transportation cost would be allocated to this specific product, and this cost allocation is too expensive for an agent to put an order, which is the reason why the agent with local credit assignment failed to learn the *coordinated ordering* in the earlier work [14].

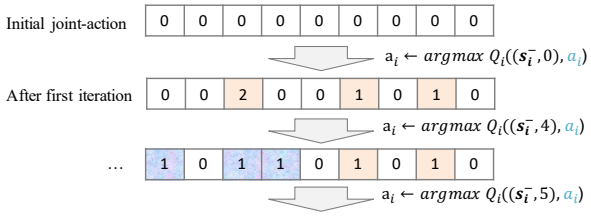

Fig. 2 image of our heuristic search procedure

On the other hand, with global reward allocation, even if given that the other agents do not put an order at all, 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.

For 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 $\text{CAP}_{\text{trans}}$ is allowed in our search procedure.

3. Experiments

3.1 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 heuristic policies under the various demand characteristics in the normal cost setting?

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

In order to validate these questions, we conducted experiments by varying 1) the number of products, 2) the cost scenario, and 3) the demand characteristics (demand variation and correlation among products). In literature, most frequent examined number of products in JRP are ranging from 2 to 12. Thus, we examined three settings; 2, 5, and 10. As for the demand characteristics, we altered two parameters; c_v and ρ, which defines the deviation of demand and correlation of demand among products, respectively. 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 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 $\text{CAP}_{\text{trans}}$.

To summarize, our experiments consist of the following factors;

1) number of products ∈ (2, 5, 10),

2) cost scenario ∈ (1, 2, 3, 4),

3) demand characteristics: $c_v \in (0.4, 0.6)$, $\rho \in (-0.5, 0, 0.5)$

3.2 Benchmark methodology

As a benchmark policy, we selected a can-order and MP policy, as they are well-known joint-replenishment policy under periodic-review inventory system. We derived the parameters of each policy using the genetic algorithm. The detail settings are provided in section 3.4.

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

In order 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 [1], 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 quantities exceed its capacity or loading ratio of the vehicle is low, where loading ratio is defined by $\sum_i x_{i,t}/(\lceil \frac{\sum_i x_{i,t}}{\mathrm{CAP_{trans}}} \rceil \times \mathrm{CAP_{trans}})$. Either of can-order and MP policy has the parameter S_i which represents the reorder 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 reorder 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 confirmed that by adding this procedures, the performance of the benchmark policies increased for both scenario 2 and 3.

3.3 Experiment results

Table. 3 shows the experiment results. Since the demand correlation among products did not make significant effect on the performance gap throughout our study, only the results with $\rho = 0$ is shown.

In terms of the comparison between the can-order and MP policy, when the number of products is small, we see that the performance of the MP policy is worse, particularly for problems with high demand deviation, which is consistent with literature. However, for problems with 10 products, the performance of the MP policy is equal or better 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-order and MP policies. For the base cost scenario with fixed joint replenishment cost, the performance of our proposed agent was equal to the better one of the benchmark policies irrespective of the demand deviation and number of products. For the scenarios of capacitated and stepwise transportation, our proposed agent performed better, and the difference in performance has widened as the number of products increased. For the non-linear holding cost scenario, the performance of ours was slightly lower than the better one of the benchmark policies, but the gap was not significant when taking the standard deviation over six runs into consideration.

3.4 Experiment detail

3.4.1 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 on the basis of the typical logistic condition. As for the non-linear holding cost scenario, fixed holding cost is set to $U^{\mathrm{hold_f}} = 0.7 \times U^{\mathrm{hold_v}} \times \mathrm{CAP_{wh}}$, where $U^{\mathrm{hold_v}} = U^{\mathrm{hold}}$.

3.4.2 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. In order to stabilize the learning process, we employed the hysteretic learning [7][13], 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. Third, the replay memory size with memory capacity was set to 10,000 so that the *obsolete* memory would not be used for learning.

3.4.3 Genetic algorithm setting

In our parameter estimation for benchmark policies, following parameters were used in genetic algorithm; The number of population, cross over probability, mutation probability, and number of generation are $50 \times N$, 0.5, 0.2, and 100, respectively.

4. Conclusion

As we see in the experiments, our numerical experiments demonstrate that the performance of our proposed agent is equal to the better one of the benchmark policies irrespective of the demand characteristics for base cost scenario. In addition, our proposed agent outperformed the benchmark policies for cases with transportation capacity constraint or stepwise transportation cost. 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 researches have not covered all the conditions. We hope our solution can be of help in the complicated practical conditions.

One of the limitations of our study is the versatility of cost structures. For future study, incorporating the 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.

References

[1] Ben-Khedher, N. and Yano, C. A.: The Multi-Item Joint Replenishment Problem with Transportation and Container Effects,

Table 2 Demand and lot size settings in each experiment

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

Table 3 Experiment results

Cost scenario	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%
	2	(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%
	5	(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%
	10	(0.6,0)	210.7 ± 2.7	209.4 ± 2	210.5 ± 1.7	-0.5%
2) CapacitatedTransportation	2	(0.2,0)	98.4 ± 1.3	102.5 ± 2.5	97.1 ± 0.6	1.4%
	2	(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%
	5	(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%
	10	(0.6,0)	229.3 ± 6.0	225.5 ± 1.3	219.1 ± 1.3	2.9%
3) StepWiseTransportation	2	(0.2,0)	507.5 ± 7.1	503.3 ± 4.3	506.5 ± 2.4	-0.6%
	2	(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%
	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%
	10	(0.6,0)	1126.4 ± 28	1108.9 ± 24.9	1014.2 ± 12.8	8.5%
4) Non-linearInventory	2	(0.2,0)	90.2 ± 1.6	91.8 ± 0.4	90.1 ± 0.5	0.1%
	2	(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%
	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%
	10	(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 $(\pm\cdot)$ represents the standard deviation over 6 runs. % improved represents the gap in performance of our proposed agent and the better one of the benchmark policies.

Transportation Science, Vol. 28, No. 1, pp. 37–54 (online), DOI: 10.1287/trsc.28.1.37 (1994).

[2] 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, No. 0 (online), DOI: 10.1590/0103-6513.222916 (2017).

[3] 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).

[4] 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, Proceedings of the 34th International Conference on Machine Learning (Precup, D. and Teh, Y. W., eds.), Proceedings of Machine Learning Research, Vol. 70, International Convention Centre, Sydney, Australia, PMLR, pp. 1146–1155 (online), available from ⟨http://proceedings.mlr.press/v70/foerster17b.html⟩ (2017).

[5] 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).

[6] 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).

[7] 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).

[8] 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).

[9] 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).

[10] 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).

[11] 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).

[12] 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).

[13] 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).

[14] Suetsugu, H., Narusue, Y. and Morikawa, H.: Branching Deep Q-Network Agent with Reward Allocation Mechanism for Joint Replenishment Policy, p. To Appear (2020).

[15] Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents, Proceedings of the tenth international conference on machine learning, pp. 330–337 (1993).