International Journal of Smart Computing and Artificial Intelligence
International Institute of Applied Informatics
2023, Vol. 7, No. 2, IJSCAI789

Improving Multi-Agent Reinforcement Learning for Beer Game by Reward Design Based on Payment Mechanism

Masaaki Hori [*], Toshihiro Matsui [*]

Abstract

Supply chain management aims to maximize profits among supply chain partners by managing the flow of information and products. Multiagent reinforcement learning in artificial intelligence research fields has been applied to supply chain management. The beer game is an example problem in supply chain management and has also been studied as a cooperation problem in multiagent systems. In the previous study, a solution method SRDQN that is based on deep reinforcement learning and reward shaping has been applied to the beer game. By introducing a single reinforcement learning agent with SRDQN as a participant in the beer game, the cost of beer inventory was reduced. However, the previous study has not addressed the case of multiagent reinforcement learning due to the difficulties in cooperation among agents. To address the multiagent cases, we apply a reward shaping technique RDPM based on mechanism design to SRDQN and improve cooperative policies in multiagent reinforcement learning. Furthermore, we propose two reward design methods with modifications to the state value function designs in RDPM to address various consumer demands for beers in the supply chain. And then we empirically evaluate the effectiveness of the proposed approaches.

Keywords: Multi-Agent Reinforcement Learning, Beer Game, Reward Shaping, Supply Chain Management, VCG Mechanism

1 Introduction

Supply chain management generally optimizes a system consisting of multiple participants in production and distribution process that cooperate or compete with each other. It manages the flow of products, raw materials, and associated information in order to improve the profitability of the entire system. In today's highly uncertain and complex society, there is a need for methods that can flexibly respond to environmental changes and demands, and achieve desirable benefits. In addition, the diversification of consumer demand and the shortening of product cycles in the supply chain have made it necessary for companies to cope with increased storage costs due to inventory holding and opportunity loss due to inventory shortages. Therefore, inventory management has become an important business challenge.

[*] Nagoya Institute of Technology, Aichi, Japan

Multiagent systems consisting of multiple agents that make decisions autonomously have been studied as a research area of artificial intelligence. Agents obtain information from environment and try to cooperatively/competitively solve complex large-scale problems of the entire system by relatively simple perceptual actions. Since it is difficult to design the agents' behavior in advance, multiagent reinforcement learning (MARL) has been studied [3, 16], where each agent aims to adapt to its environment by trial and error, depending on the surrounding situation.

We address the improvement of an existing solution method for the beer game that is an example problem of supply chain management [13]. The beer game deals with inventory management in the supply chain, and consists of a serial supply chain network in which four agents are located at the retailer, warehouse, distributor and manufacturer [5]. The agents must make independent replenishment decisions with limited information, and the objective of the beer game is to minimize the inventory cost of the entire system. The system can be modeled as a coordination problem on a decentralized multiagent system.

In the study by Afshin et al. [13], Shaped Reward DQN (SRDQN), which is based on the Deep Q-Network (DQN) [9] has been proposed for solving the beer game. SRDQN employs reward shaping, which considers the interaction among multiple agents. At the end of each game, the history of states, actions and rewards of agents are shared, and reward shaping is applied to the rewards of the agent performing reinforcement learning. This improves the quality of learned policies and the overall inventory cost of the system. In the previous work, one of four agents in a beer game performs SRDQN, and the other agents act according to predefined rules. This involves the attempt to stabilize the entire system, including the unlearning agents. However, there is room to improve the solution quality and stability of the conventional method. Moreover, a system in which all four agents perform reinforcement learning has not yet been considered.

Matsunami et al. [8] have proposed Reward Design for MARL based on the Payment Mechanism (RDPM), which is a reward design method to improve cooperative policies in MARL. RDPM applies the concept of payment used in mechanism design. It causes agents to avoid selfish policies and learn effective policies to increase social surplus.

We apply RDPM to the SRDQN framework to improve the accuracy and stability of learning in the conventional method. In particular, this reduces the cost value of the system in which all four agents perform reinforcement learning. In the study of RDPM [8], the technique is applied to the immediate rewards of each agent in MARL. However, in the beer game, RDPM cannot be applied to rewards during each play of the game because information sharing among players is prohibited during each play. We investigate the case of application of RDPM to reward shaping in SRDQN for the beer game. Furthermore, we propose two reward design methods with modifications to the state value function designs in RDPM to address various consumer demands in the supply chain. We evaluate the effectiveness of the proposed method through experiments.

Our study shows how MARL can be applied to inventory optimization in a fundamental game of supply chain management. We expect that our approach can be applied to other supply chain management problems in which supply chain participants behave unpredictably or selfishly. This study is an extension of our previous work [11]. We improve the description by adding new reward design methods and experimental results.

The rest of the paper is organized as follows. The next section presents the preliminaries of this study, including a description of the beer game, SRDQN, and RDPM, with reference to previous studies. Then we present how RDPM is applied to SRDQN and the four reward design methods for comparison in Section 3. We experimentally investigate the proposed

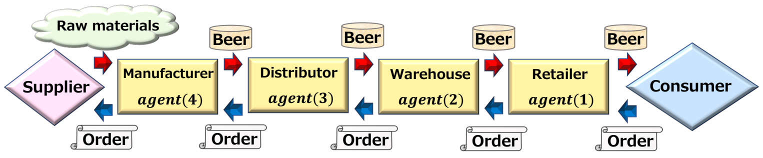

Figure 1: Beer Game

approach in Section 4 and conclude our study in Section 5.

2 Preliminaries

In this section, we present the background of our research, including multiagent reinforcement learning, the beer game, the conventional method SRDQN, and the reward design method RDPM based on the Vickrey-Clarke-Groves (VCG) Mechanism.

2.1 Multi-Agent Reinforcement Learning (MARL)

Reinforcement learning is a machine learning technique for agents interacting with their environment. The agent explores its environment and empirically acquires a policy consisting of a sequence of optimal actions. Reinforcement learning is based on an agent's set of states S, a set of actions A and a reward function $R : S \times A \to r$. At each time step t, the agent observes the current state $s_t \in S$ of the system, selects an action $a_t \in A$, obtains a reward r_t, which then causes the system to transition to state s_{t+1}. The goal of reinforcement learning is to determine the policy $\pi : S \to A$ that maximizes the expected discounted sum of the reward r_t. In multiagent reinforcement learning (MARL), reinforcement learning is performed by multiple agents that share an environment. Agents consider their interactions and aim to obtain a policy that improves their own or the entire system's benefit.

2.2 Beer Game

The beer game (Figure 1) is performed on a serial supply chain network consisting of four agents arranged in the following order: retailer, warehouse, distributor, and manufacturer. Each agent must make inventory replenishment decisions based on the limited information available from its neighboring agents. This game is widely used as an example to illustrate the bullwhip effect, in which order variability increases as one moves upstream in the supply chain, and the importance of communication and coordination in supply chains [6].

In this system, the manufacturer is at the upstream end and the retailer is at the downstream end. The retailer agent supplies beer in response to stochastic demand from consumers. The manufacturer agent can receive an unlimited amount of raw materials from suppliers. For each agent's inventory, the shortage cost and the excess holding cost are evaluated. During the play of the beer game, the demand for beer occurs from the consumer at the beginning of each time. In each period of the game, each agent chooses the quantity q of discrete orders to be submitted to its upstream agent or suppliers to minimize the long-term cost of the entire system. The entire system's inventory cost is given by the following equation.

$$\sum_{t=1}^{T}\sum_{i=1}^{4} c_h^i \times \max(0, IL_t^i) + c_p^i \times \max(0, -IL_t^i) \tag{1}$$

In Equation (1), i is the index of the agent, t is the index of the period, T is the time horizon of the game, c_h^i and c_p^i are the coefficients of agent i's holding cost and shortage cost, and IL_t^i is agent i's inventory level at period t. The inventory level is an integer value representing the amount of beer in stock, or a negative number if beer is in short supply.

The game can be modeled as a coordination problem on a decentralized multiagent system. We focus on the previous study [13] that aims to reduce the cost of the system by introducing a player agent that performs reinforcement learning. In that setting, one agent is the reinforcement-learning agent, and the other agents behave according to a prescribed algorithm.

In the standard rules of the beer game, agents cannot communicate in any way during a play. Only at the end of a game, all agents can share local inventory statistics and cost information with each other, and thus the agents that are capable of learning can improve their policies. However, during a play, each agent makes decisions based only on partial information about the environment. The objective of the game is to minimize the total cost of the entire system by cooperating with other agents under the above conditions. As in the previous study [13], we evaluate the results of playing a beer game by simulation.

2.3 Shaped Reward DQN (SRDQN)

In the study by Afshin et al. [13], Shaped Reward DQN (SRDQN), which is based on the deep reinforcement learning method DQN [9], is proposed for solving the beer game. SRDQN employs a reward shaping technique to consider the interactions among multiple agents. At the end of each game, the history of the states, actions and rewards of the agents in the play is shared, and reward shaping is applied to the rewards of the agents that perform reinforcement learning. This improves the accuracy and stability of learning and further improves the inventory cost of the entire system. In the previous study, a single agent performs reinforcement learning with the goal of reducing the entire system cost. The other agents act is based on prescribed rules of behavior, such as the Sterman formula [15]. The system of reinforcement learning in the previous study is represented by $\langle S, A, R\rangle$. S denotes the set of states s, A denotes the set of actions a, and R denotes the reward function. The details of the reinforcement learning are described below.

State: The state is the history of the last m steps of play. The state of agent i at time t is represented as follows.

$$s_t^i = [(\max(0, IL_j^i), \max(0, -IL_j^i), OO_j^i, AO_j^i, AS_j^i)]_{j=t-m+1}^{t} \tag{2}$$

In Equation (2), OO_t^i is the amount of on-order beers at agent i, in other words, the amount that agent $i+1$ has ordered but not yet received. AO_t^i is the amount of order received from agent $i-1$, and AS_t^i is the amount of beer that has arrived from agent $i+1$. a_t^i is the action taken by agent i, and IL_t^i is the inventory level. The inventory level represents the amount of beer in stock at each facility. If an agent is not able to fulfill an order from a downstream agent, the shortage is evaluated as a negative value. An overstock is evaluated as a positive value. AO_t^1 represents the demand from consumers and AS_t^4 represents the amount of raw materials from external suppliers.

Action: Although the beer game rules allow an agent to order any amount of beer in $[0, \infty]$ during each period of the game, some restrictions are placed on the amount that agents can order in a simulation of actual play, as shown in Equation (3). Action a_t^i of agent i at time t is the order quantity chosen by agent i, and it is represented by using constant a_u as follows.

$$a_t^i = AO_t^i + x, \ (x : \text{-}\mathrm{a}_u \leq x \leq \mathrm{a}_u) \tag{3}$$

Reward function: Using the coefficient of holding cost c_h^i and the coefficient of shortage cost c_p^i given to each agent, the reward is shown as follows.

$$r_t^i = -\{\mathrm{c}_p^i \times \max(0, -IL_{t+1}^i) + \mathrm{c}_h^i \times \max(0, IL_{t+1}^i)\} \tag{4}$$

The reward is negative so that the agent gets low reward when the inventory cost is high. The maximum reward at each time in the beer game is zero.

DNN: The Deep Neural Network (DNN) acts as an approximator of the Q function, outputting Q values for any pair of state s and action a. The DNN is trained iteratively using random mini-batches taken from the experience replay memory until all episodes are completed. At each time t, observation data $e_t^i = (s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ is added to the experience replay memory of each agent using SRDQN.

Feedback scheme: At the end of each episode of the beer game, a reinforcement learning agent using SRDQN updates the observed reward in the last T time steps in its experience replay memory by Equation (5) so that the agent considers the information of the reward for the entire system.

$$r_t^i = r_t^i + \frac{\beta^i}{3}\left(\frac{1}{T}\sum_{i=1}^{4}\sum_{t=1}^{T} r_t^i - \frac{1}{T}\sum_{t=1}^{T} r_t^i\right), \ (\forall t \in \{1, ..., T\}) \tag{5}$$

In Equation (5), β^i is a coefficient of agent i to balance the information between the original reward and the reward of the entire system.

2.4 Reward Design based on Payment Mechanism (RDPM) for MARL

In the study by Matsunami et al. [8], RDPM that is a reward design method for MARL based on the Vickrey-Clarke-Groves (VCG) mechanism [14] has been proposed. Mechanism design is a branch of microeconomics and game theory that deals with the problem of how to efficiently bring together multiple selfish agents and design a mechanism that maximizes social surplus. Unlike the common difference reward [1], which takes into account the degree of contribution of the agent, RDPM uses nuisance payments of the agent. To compute the payment, a state value function representing the social utility is designed in advance. The payment to agent i is based on the difference between two values determined by the state value function. One is the sum of the values determined by the other agents when agent i is in the environment. The other value is the sum of the values determined by the other agents when agent i is not present in the environment. As with the VCG payment, the difference between the two is subtracted from the reward earned by the agent being evaluated. The payment of agent i represents the extent to which the presence of agent i affects others. By incorporating these payments into the reward design, agents avoid selfish policies and learn effective policies to increase social surplus. The payment rule is defined by the Equations (6) and (7).

$$p^i = \sum_{j \neq i} v^j(S^{-i}) - \sum_{j \neq i} v^j(S) \tag{6}$$

$$= V^{-i}(S^{-i}) - V^{-i}(S) \tag{7}$$

In Equations (6) and (7), state value function $V(S)$ is introduced to evaluate the nuisance in state S, and it is defined in advance. S^{-i} denotes the situation excluding the influence of agent i. $V^i(S)$ represents the case with the contribution of agent i, and $V^{-i}(S)$ is the case without the contribution of agent i. In order to deal with the problem of individual profit and overall profit not necessarily coinciding, that study define payment amount p^i as a part of agent i's reward as shown in Equation (8).

$$r_i = \alpha R(s^i) - p^i \tag{8}$$

In Equation (8), $R(s^i)$ is the local reward function determined for agent i only. It is treated as corresponding to the value function of each individual agent. In addition, α is a coefficient that determines the weight between the local reward and the payment amount, and it is used to adjust the balance between the local reward and the social value.

In the previous study, RDPM yielded higher average rewards than a similar reward shaping method based on difference rewards. The difference appears particularly clear as the problem setup becomes more complex.

3 Application of RDPM to SRDQN

In the previous study [13], one of four agents in a beer game employs SRDQN and the other agents act according to the predefined rules. In other words, the case in which an attempt is made to stabilize the entire system including the unlearning agents has been considered. However, there is room to improve the accuracy and stability of the learning results of the proposed method. In addition, a system in which all four agents perform reinforcement learning has not been considered. In this study, we apply more effective reward shaping [8] to improve the stability of learning and the cost of the system.

In the study by Matsunami et al. [8], RDPM is applied to each agent's immediate reward in multiagent reinforcement learning, while the experience replay memory can be employed with DQN. However, in the beer game, RDPM cannot be applied directly to rewards during game play because information sharing among players is prohibited during game play. Therefore, we investigate a method to apply RDPM to the feedback scheme in SRDQN. This method is denoted as SRDQN+RDPM. By applying RDPM, we improve the cooperative policies when using MARL.

In addition to SRDQN+RDPM, we define the state value function for a more realistic consumer demand. Actual consumer demand for beer varies gradually with periodic peaks throughout the year. Therefore, we define the state value function using the variation of cost values between time series, and denote this method as TSRDPM. The method that takes account of the variation of cost values between time series is denoted as SRDQN+TSRDPM.

The previous study [8] also presented a method based on difference reward [1] for comparison, which is slightly different from the SRDQN formulation. We show the application of these methods to the beer game. In the following part of this section, we first explain the difference between the problem in which RDPM was applied in the previous study [8] and

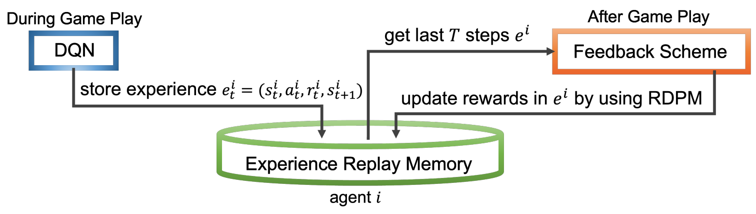

Figure 2: Applying RDPM to rewards in experience replay memory

that in the beer game, and we explain why RDPM is applied to the rewards in the experience replay memory of each reinforcement learning agent. Then, we define the state value function for determining the RDPM payment in the beer game. Furthermore, we define the state value function of SRDQN+TSRDPM for more realistic consumer demand. We then explain how the three methods are applied and compared to the conventional method SRDQN.

3.1 Updating rewards in experience replay memory by using RDPM

In the previous study [8], RDPM has been applied to the Highway Driving Problem Domain (HDD), which simplifies the problem of autonomous driving on highways [2], and the Predator-Prey Domain (PPD) [7]. In HDD, learning agents driving autonomously on a highway attempt to cooperate. A driving agent's personal utility is to drive as fast as possible, while its social utility is to avoid dangerous driving by allowing each agent to match its speed with that of other agents. PPD consists of four predator agents and a single prey agent. Predators learn and cooperate to hunt a single prey that performs a predefined policy. In these two problems, the learning agent can obtain rewards by taking into account the information of other agents in the environment.

In the beer game, however, each agent cannot share information with other agents during a play, and the agent's immediate reward cannot reflect information other than its own. Figure 2 shows the method that apply RDPM to SRDQN. During game play, each reinforcement learning agent is only aware of its individual utility, making it difficult for the entire system to cooperate. Then, at the end of the game, all agents can share their local inventory statistics and cost information with each other. In the feedback scheme of SRDQN performed at this step, the history of actions and rewards of the agents in play are shared, and reward shaping is applied to the rewards in the experience replay memory of the agents performing reinforcement learning. We apply RDPM to reward shaping in the feedback scheme to improve the cooperative policies of reinforcement learning agents.

3.2 Definition of state value function for payment

We use the cost value of Equation (9), which inverts the positive and negative values of Equation (4) representing the agent's reward in the beer game.

$$cost_t^i = c_p^i \times \max(0, -IL_{t+1}^i) + c_h^i \times \max(0, IL_{t+1}^i) \tag{9}$$

Using $cost_t^i$, we define the state value function of agent i for calculating the payment in RDPM as shown in Equations (10) and (11). Here, $v_t^i(S)$ in Equation (11) is defined as the difference in the cost value between the agent having the largest cost value among all agents participating in the beer game and agent i. The strategy of this formulation resembles the study of Matsunami et al. [8]. Then, the state value function is formulated.

$$V(S) = \sum_{i \in N} v_t^i(S) \tag{10}$$

$$v_t^i(S) = \max_{k \in N}(cost_t^k - cost_t^i) \tag{11}$$

Here, N denotes the number of all agents in the system. By the definition of the state value function given in Equations (10) and (11), the payment increases when one agent takes an action that produces a larger cost value than the other agents. Therefore, this agent is expected to cooperate with other agents to decrease the cost value of the entire system.

3.3 Another state value function for continuous demand fluctuations

In addition to the state value function in Equation (11), we propose another state value function aimed to deal with the case of relatively continuous demand fluctuations in the supply chain. In a realistic consumer demand for beer, it may be relatively cyclical or a little noise may be placed on a relatively continuous fluctuation.

Therefore, we define the state value function that considers the variation of cost values in a time series as shown in Equation (12). Instead of the cost value of each agent in time t in Equation (11), the difference between the average cost value in the last τ times and the cost in time t is used in Equation (12).

$$v_t^i(S) = \max_{k \in N}((cost_t^k - \frac{1}{\tau}\sum_{l=1}^{\tau} cost_{t-l}^k) - (cost_t^i - \frac{1}{\tau}\sum_{l=1}^{\tau} cost_{t-l}^i)) \tag{12}$$

In Equation (12), for agents i and $k \in N$, we first compute the difference between the cost value at time t and the average of the cost values for the last τ times. Next, we further compute the difference between the previously computed i's difference and k's one. This difference is computed for each pair of agents i and $k \in N$ in the supply chain, and the maximum difference value is defined as i's state value $v_t^i(S)$ at time t. The payment decreases when one of agents takes an action that produces a lower cost value at time t than the average cost value in the last τ times, and the difference value is lower than that of other agents. Therefore, agent i is expected to cooperate with other agents to decrease the cost value of the entire system with the cyclical consumers' demand for beers by considering the improvement in cost value from the last τ times.

We denote this RDPM method that considers the cost value between time series as Time Series RDPM (TSRDPM). Then, the method to apply TSRDPM to the feedback scheme in SRDQN is denoted as SRDQN+TSRDPM.

3.4 Four reward design methods for comparison

In the previous study [8], to evaluate the effectiveness of the RDPM, global reward (GR) and difference reward (DR) [1], which are the basic reward design methods used in MARL, are used for comparison. GR is a method that gives the same global reward to all agents. In a baseline approach of MARL, GR defines a desired state as a whole and then agents aim to obtain a policy that maximizes GR through learning. However, learning that only considers of GR may not be sufficient for learning interactions. DR is a method to evaluate the contribution of agent i by considering the reward of this agent as the difference between the utility of the entire system and the utility when the agent i is excluded. DR is represented as Equation (13).

$$r^i = R(S) - R(S^{-i}) \tag{13}$$

The difference between Equation (7) and Equation (13) is that Equation (7) evaluates the degree of nuisance of agent i, while Equation (13) evaluates the contribution of agent i.

We denote the method that applies DR to the SRDQN feedback scheme as SRDQN+DR. To make a comparison with SRDQN+RDPM, we treat Equation (13) as a payment and express the reward update equation used in SRDQN+DR as shown in Equation (14).

$$r^i = R(s^i) + \gamma(V(S) - V^{-i}(S^{-i})) \tag{14}$$

Here, γ is a coefficient for adjusting the weight of the payment. Local reward $R(s^i)$ in the beer game is represented as shown in Equation (15) as well as in Equation (4).

$$R(s^i) = -\{\mathrm{c}_p^i \times \max(0, -IL_{t+1}^i) + \mathrm{c}_h^i \times \max(0, IL_{t+1}^i)\} \tag{15}$$

In this study, we compare the reward shaping of SRDQN with the three other methods: SRDQN+DR and our two proposed methods SRDQN+RDPM/TSRDPM. The update equations in the feedback scheme for the methods are defined as Equations (16), (17), (18) and (19).

SRDQN:

$$r_t^i = r_t^i + \frac{\beta^i}{3} \cdot \frac{1}{T}\left(\sum_{i=1}^{4}\sum_{t=1}^{T} r_t^i - \sum_{t=1}^{T} r_t^i\right), \ (\forall t \in \{1, ..., T\}) \tag{16}$$

SRDQN+DR:

$$r_t^i = r_t^i + \gamma\left(\sum_{j \in N} \max_{k \in N}(cost_t^k - cost_t^j) - \sum_{j \neq i} \max_{k \neq i}(cost_t^k - cost_t^j)\right), \ (\forall t \in \{1, ..., T\}) \tag{17}$$

SRDQN+RDPM:

$$r_t^i = r_t^i - \gamma\left(\sum_{j \neq i} \max_{k \neq i}(cost_t^k - cost_t^j) - \sum_{j \neq i} \max_{k \in N}(cost_t^k - cost_t^j)\right), \ (\forall t \in \{1, ..., T\}) \tag{18}$$

SRDQN+TSRDPM:

$$\begin{aligned} r_t^i = & r_t^i - \gamma\Bigg(\sum_{j \neq i} \max_{k \neq i}\left((cost_t^k - \frac{1}{\tau}\sum_{l=1}^{\tau} cost_{t-l}^k) - (cost_t^j - \frac{1}{\tau}\sum_{l=1}^{\tau} cost_{t-l}^j)\right) \\ & - \sum_{j \neq i} \max_{k \in N}\left((cost_t^k - \frac{1}{\tau}\sum_{l=1}^{\tau} cost_{t-l}^k) - (cost_t^j - \frac{1}{\tau}\sum_{l=1}^{\tau} cost_{t-l}^j)\right)\Bigg), \ (\forall t \in \{1, ..., T\}) \end{aligned} \tag{19}$$

4 Evaluation

4.1 Settings of experiment

The effectiveness of the proposed method in the beer game was evaluated through experiments. In the simulation of the beer game, we used the same experimental environment as in the previous study [13]. We evaluated the cost values shown in Equation (1) as follows. After playing 100 times with the training data, the game was played 50 times with the test data, and the average cost of the 50 test results was recorded as the evaluation value for the

Table 1: Simulation parameter settings

Number of episodes	40,000	State space size (m)	10
Steps per episodes	100	Constant for action (a_u)	50
Learning rate	2.5×10^{-3}	Constant for SRDQN (β)	10
Gamma	0.99	Shortage cost coefficients (c_p)	[1, 1, 1, 1]
Initial ε value	0.9	Holding cost coefficients (c_h)	[1, 1, 1, 1]
End ε value	0.1	Transportation lead times (l_{tr})	[2, 2, 2, 4]
Experience replay memory size	500,000	Information lead times (l_{in})	[2, 2, 2, 0]
Batch size	128		

number of training sessions at each time point. This procedure was repeated up to the maximum number of training sessions to evaluate the evolution of the average cost per game as the reinforcement learning agent's training progressed. We implemented an experimental environment including the proposed method based on a simulation of the beer game [12]. We evaluated a system including only one reinforcement learning agent and a system with four players performing reinforcement learning. We compared the conventional method SRDQN, a compared method SRDQN+DR, the proposed methods SRDQN+RDPM and SRDQN+TSRDPM. A system consisting only of agents using the Sterman formula [15] was used as a baseline. The other co-players in the system with only one player performing reinforcement learning used the Sterman formula to determine the amount of beer to order each period, as in the previous study [13]. The Sterman formula models the behavior of real human players, and an agent's order is represented by Equation (20).

$$q_t^i = \max\{0, AO_{t+1}^{i-1} - 0.5(IL_t^i - \mathrm{a}^i) - 0.2(OO_t^i - \mathrm{b}^i)\} \tag{20}$$

Here, a^i and b^i are parameters corresponding to the inventory level and the amount of beer currently on order. For all agents $(i = 1,2,3,4)$, $\mathrm{a}^i = \mu_d, \mathrm{b}^i = \mu_d(l_i^{f_i} + l_i^{t_r})$ with the average demand from consumers μ_d. $l_i^{f_i}$ and $l_i^{t_r}$ are the lead times for order and product flows.

Agents performing reinforcement learning use DQN [9] as the learning algorithm. The agent learns each Q-network based on its own state and behavior. The input to the Q network is the value of each state vector. The neural network has three hidden layers, with each hidden layer having 180, 130, and 61 outputs. Each output was activated by a ReLU function [10], and the optimizer was Adam [4]. The simulation of the beer game was performed using the parameters shown in Table 1. The parameters of the experiments were selected based on the settings of the previous study, and the parameters that gave typical results were selected by preliminary experiments.

In the evaluation of SRDQN+TSRDPM, the cases $\tau = 1$ and $\tau = 5$ were tested. Here, the parameter τ is a constant that determines the degree of use of information about past cost values in determining the payment of TSRDPM. In the captions of the following results figures of SRDQN+TSRDPM, the case $\tau = 1$ was denoted SRDQN+TSRDPM-τ1, and the case $\tau = 5$ was denoted SRDQN+TSRDPM-τ5.

We tested two types of demand fluctuations as shown in Figures 3 and 9. During the play of the beer game, the demand for beer occurs from the consumer at the beginning of each time. The first is the case of random demand fluctuations (Figure 3). In the first case, we assumed a supply chain in which the amount of demand from consumers changes rapidly over a short period of time, and evaluated problems with only one reinforcement learning agent and problems with all four agents using reinforcement learning. The second

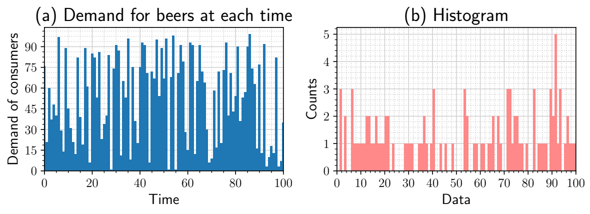

Figure 3: Demand of consumers in case of random demand fluctuations

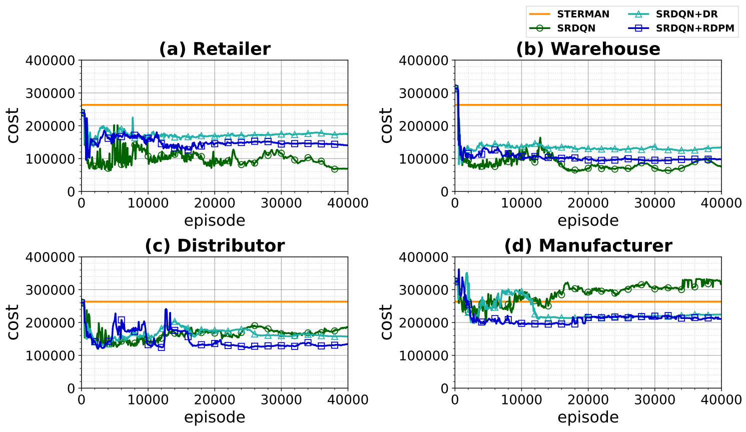

Figure 4: Total cost (single RL agent)

is the case of relatively continuous demand fluctuations (Figure 9). In the second case, we assumed a more realistic supply chain in which the amount of demand from consumers changes slowly with periodic peaks, and evaluated only problems with all agents using reinforcement learning.

4.2 Case of random demand fluctuations

In this section, the amount of consumers' demand was represented by a discrete uniform distribution taking integer values of $[0, 100]$. Figure 3 shows the example which represents fluctuation of demand for beers during a play of the beer game. In this case, a system consisting only of agents using the Sterman formula [15] was used as a baseline.

4.2.1 Results for case of single reinforcement learning agent

Figure 4 shows the evolution of cost values during the learning process for the system with a single reinforcement learning agent, and the other co-players use the Sterman formula. We show where the reinforcement learning agent is plased. In Figure 4, we show the learning results for SRDQN+DR and SRDQN+RDPM with $\gamma = 1.0$. For this system, the result of one instance is shown because there was little difference in the experimental results from instance to instance. When the reinforcement learning agent was placed in a retailer or a warehouse, the cost value was lower for SRDQN ((a) and (b) in Figure 4). Since the retailer agent and the warehouse agent are in the positions to be affected by erratic demand

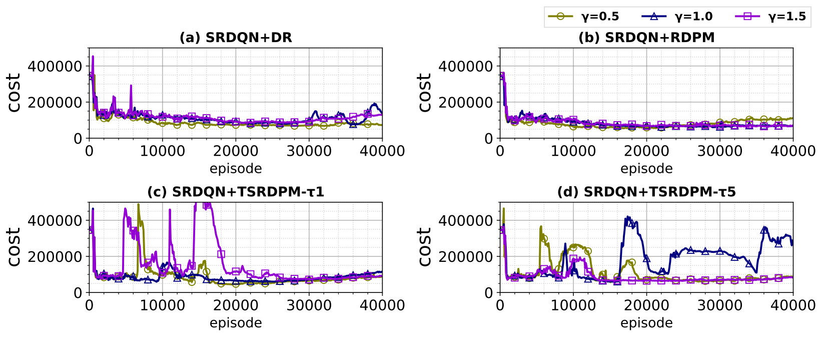

Figure 5: Total cost (four RL agents with $\gamma = 0.5, 1.0, 1.5$)

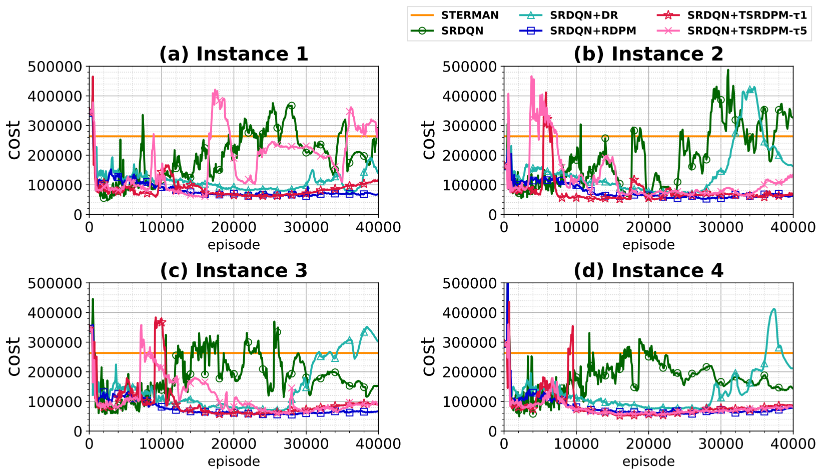

Figure 6: Total cost (four RL agents)

from consumers, it is expected that the cost value could be significantly improved if the learning process were successful. When the reinforcement learning agents were placed at distributors or manufacturers, SRDQN+RDPM had the lowest cost values, but not by much ((c) and (d) in Figure 4). The more upstream the reinforcement learning agent was placed, the smaller the contribution of the reinforcement learning agent became, making it more difficult to improve the cost value. Except for the case where the agent using SRDQN was placed at the manufacturer, the cost values were better than those of the system consisting only of agents using the Sterman formula. For all of the methods, the system with only one agent performing reinforcement learning had a higher stability of learning. The cost values of SRDQN+DR and SRDQN+RDPM did not differ significantly, although SRDQN+RDPM showed relatively lower cost values for any facility with a reinforcement learning agent.

4.2.2 *Results for case of four agents using reinforcement learning*

Figure 5 shows the evolution of the cost values during the learning process with each values of γ. To evaluate the impact of the parameter γ on the learning process of SRDQN+DR, SRDQN+RDPM and SRDQN+TSRDPM, three types of $\gamma \in \{0.5, 1.0, 1.5\}$ were tested. In

Table 2: Statistical information of total cost value (6 instances)

RL method	Min	Max	Average	Variance
SRDQN	$\mathbf{5.099 \times 10^4}$	4.965×10^5	1.628×10^5	5.316×10^9
SRDQN+DR	7.200×10^4	4.520×10^5	1.348×10^5	6.090×10^9
SRDQN+RDPM	5.519×10^4	$\mathbf{4.329 \times 10^5}$	$\mathbf{8.212 \times 10^4}$	$\mathbf{2.369 \times 10^9}$
SRDQN+TSRDPM-τ1	5.276×10^4	4.628×10^5	8.647×10^4	3.414×10^9
SRDQN+TSRDPM-τ5	5.923×10^4	5.046×10^5	1.318×10^5	7.740×10^9

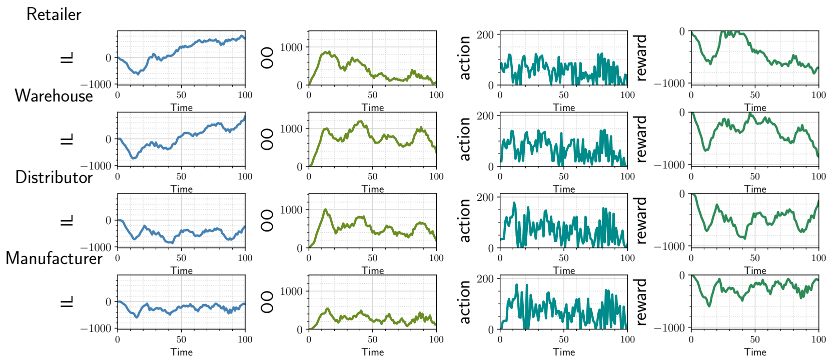

Figure 7: IL, OO, action and reward of all agents (SRDQN)

each method, the parameter γ is used as a coefficient for adjusting the weight of payment. SRDQN+DR and SRDQN+RDPM did not show significant differences in learning stability when γ was varied ((a) and (b) in Figure 5). SRDQN+TSRDPM-τ1 showed worse learning stability in the early stages with γ =1.5 ((c) in Figure 5). SRDQN+TSRDPM-τ5 showed weaker learning stability with γ =1.0 ((d) in Figure 5).

Figure 6 shows the evolution of the cost values during the learning process for the system consisting only of reinforcement learning agents. We present four typical experimental results, since there were relatively large differences in the results. In Figure 6, we show the learning results for SRDQN+DR, SRDQN+RDPM, SRDQN+TSRDPM-τ1 and SRDQN+TSRDPM-τ5 with $\gamma = 1.0$. For all instances, SRDQN+RDPM had the lowest cost value and the highest learning stability. However, SRDQN+TSRDPM-τ1 also had high learning stability and the second lowest cost value following SRDQN+RDPM. The results for different values of τ in SRDQN+TSRDPM show that using a smaller τ resulted in more stable learning and smaller cost values. In this case, we concluded that using information from the exceedingly distant past would result in poor performance in the learning process. When SRDQN was used, the stability of learning became worse. The learning results were also unstable in some cases of using SRDQN+DR. Since SRDQN+DR is based on a reward design similar to that of SRDQN, this can be considered a common weakness. Compared to the results for the system with a single reinforcement learning agent, SRDQN+RDPM was able to significantly improve the cost value over those of the other methods. The increase in the number of agents using SRDQN+RDPM can be attributed to the improvement in cooperative policy.

Table 2 shows statistical information of total cost value for the system consisting only of reinforcement learning agents in the case of random demand fluctuations. The values in Table 2 represent the average values of the various statistics for the cost values of six in-

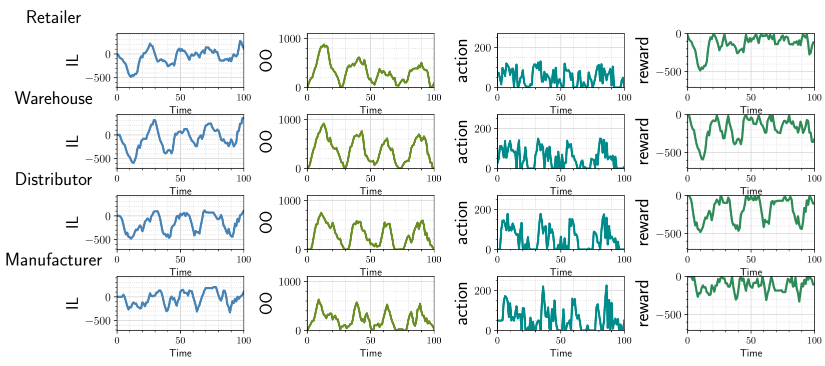

Figure 8: IL, OO, action and reward of all agents (SRDQN+RDPM)

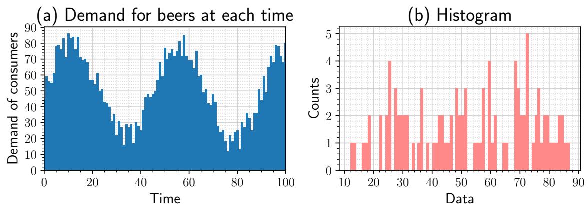

Figure 9: Demand of consumers in case of relatively continuous demand fluctuations

stances. The value of γ for each reinforcement learning method is the same as the value used in the experiments in Figure 6. In terms of average value and variance, SRDQN+RDPM was able to minimize the cost the most, while SRDQN+TSRDPM-τ1 was able to minimize it the second most. SRDQN+RDPM had the lowest average value and variance, indicating that it contributed to the stability of the entire system with a relatively low cost value.

Figures 7 and 8 show examples of the results of playing a beer game with SRDQN or SRDQN+RDPM after training with 40,000 episodes in Instance 3. We show the details of Inventory Level (IL), Open order (OO), action and reward for each agent. Here, IL and OO are as shown in Section 2.3, IL is the inventory level that represents the amount of beers in stock or lack at each facility, and OO is the amount of on-order beers at each agent. First, note that in the initial example we used, the system is particularly difficult to stabilize due to the extreme situation where consumer orders always follow a uniform distribution. When SRDQN+RDPM was used, a time-series relationship can be read between the amount of orders placed by each player (Figure 8). However, in the case of SRDQN, some agents behaved selfishly and did not cooperate well with each other (Figure 7).

4.3 Case of relatively continuous demand fluctuations

In this experiment, the amount of consumers' demand was represented by a periodic function. Figure 9 show the example which represents demand for beers during a play of the beer game. In contrast to the case of random demand fluctuations (Figure 3), there are no large fluctuations over a short period of time, and peaks occur periodically in the case of relatively continuous demand fluctuations (Figure 9). In this case, we also evaluate the system consisting only of reinforcement learning agents.

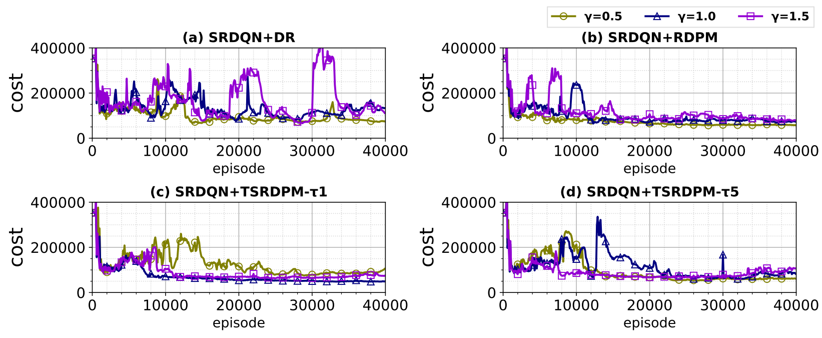

Figure 10: Total cost (four RL agents with $\gamma = 0.5, 1.0, 1.5$)

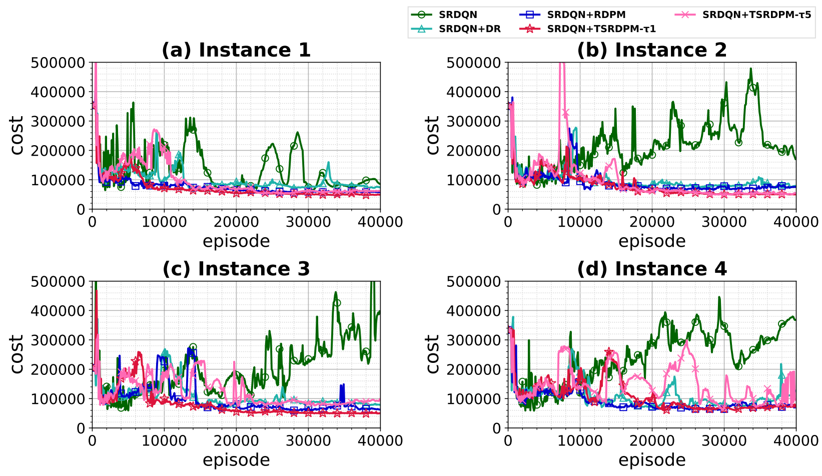

Figure 11: Total cost (four RL agents)

Figure 10 shows the evolution of the cost values during the learning process with each values of γ. To evaluate the impact of the parameter γ on the learning process of SRDQN+DR, SRDQN+RDPM and SRDQN+TSRDPM, three types of $\gamma \in \{0.5, 1.0, 1.5\}$ were tested. In each method, the parameter γ is used as a coefficient for adjusting the weight of payment. When the value of γ was varied, SRDQN+DR differed in terms of learning stability, with the most stable and lowest cost with $\gamma = 0.5$ ((a) in Figure 10). The other methods did not show significant differences in learning stability when γ was varied. In terms of cost minimization, SRDQN+DR, SRDQN+RDPM, and SRDQN+TSRDPM-τ5 had the lowest cost with $\gamma = 0.5$, and SRDQN+TSRDPM-τ1 had the lowest cost with $\gamma = 1.0$.

Figure 11 shows the evolution of the cost values during the learning process. We present four typical experimental results, since there were relatively large differences in the results. In the experiments in Figure 11, each method used the value of γ when it achieved the lowest cost value in the experiments in Figure 10. Thus, Figure 11 shows the learning results for SRDQN+DR, SRDQN+RDPM and SRDQN+TSRDPM-τ5 with $\gamma = 0.5$, and for SRDQN+TSRDPM-τ1 with $\gamma = 1.0$. When SRDQN was used, the learning stability was worse in all instances, and the cost values were sometimes very large.

Table 3: Statistical information of total cost value (6 instances)

RL method	Min	Max	Average	Variance
SRDQN	6.386×10^4	5.680×10^5	2.033×10^5	7.555×10^9
SRDQN+DR	7.010×10^4	$\mathbf{4.366 \times 10^5}$	1.060×10^5	2.788×10^9
SRDQN+RDPM	6.188×10^4	4.513×10^5	9.241×10^4	$\mathbf{2.767 \times 10^9}$
SRDQN+TSRDPM-τ1	$\mathbf{5.172 \times 10^4}$	4.411×10^5	$\mathbf{8.577 \times 10^4}$	3.377×10^9
SRDQN+TSRDPM-τ5	6.562×10^4	5.085×10^5	1.352×10^5	5.875×10^9

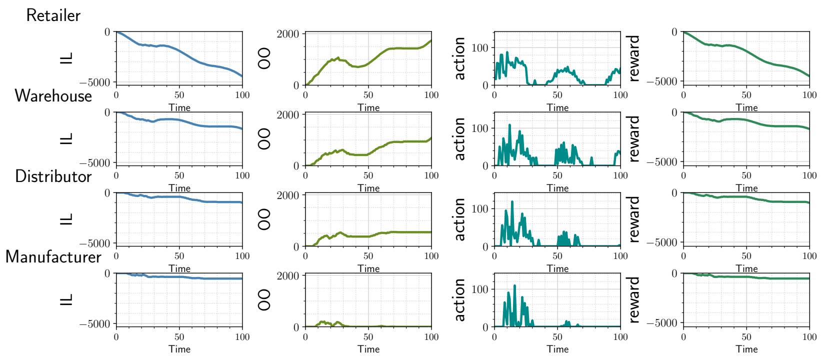

Figure 12: IL, OO, action and reward of all agents (SRDQN)

When SRDQN+DR or SRDQN+TSRDPM-τ5 was used, the learning stability was also worse in some cases, but more stable than when SRDQN was used. By contrast, when SRDQN+RDPM and SRDQN+TSRDPM-τ1 were used, learning stability was very high in both cases, but SRDQN+TSRDPM-τ1 achieved a smaller total cost value. The results for different values of τ in SRDQN+TSRDPM show that using a smaller τ resulted in more stable learning and a smaller cost value. In this case, we also have concluded that using information from the too distant past can reduce the performance in the learning process.

Table 3 also shows statistical information of total cost value for the system consisting only of reinforcement learning agents in the case of relatively continuous demand fluctuations. The values in Table 3 represent the average values of the various statistics for the cost values of six instances. The value of γ for each reinforcement learning method is the same as the value used in the experiments in Figure 11. SRDQN+TSRDPM-τ1 had the lowest average value and minimum value. While SRDQN+RDPM was the best at minimizing the average total cost in the case of random consumer demand fluctuation, SRDQN+TSRDPM-τ1 had the best cost minimization performance in this case. Further improvement in the cost value of inventory would be expected by modifying the reward design in the feedback scheme depending on the fluctuation of consumer demand. In our experiments, the results for $\tau = 1$ were better. Therefore, it could be one direction to take into account further information on the derivative of the time variation.

Figures 12 and 13 also show examples of the results of playing a beer game with SRDQN or SRDQN+TSRDPM after training with 40,000 episodes in Instance 3. We also show the details of Inventory Level (IL), Open order (OO), action and reward for each agent. Here, IL and OO are as shown in Section 2.3, IL is the inventory level that represents the amount of beers in stock or lack at each facility, and OO is the amount of on-order beers at each agent. When SRDQN was used, the lack of inventory for downstream agents in the

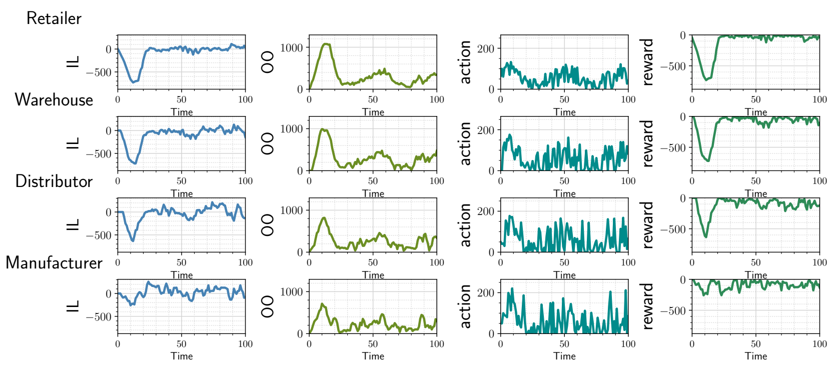

Figure 13: IL, OO, action and reward of all agents (SRDQN+TSRDPM-τ1)

supply chain was very large, and consequently the cost value also increased (Figure 12). In contrast, when SRDQN+TSRDPM-τ1 was used, a time-series relationship can be read between the actions, and rewards were stable at high values except in the early stages of the game (Figure 13).

4.4 Comparing game performance by number and placement of RL agents

We evaluated the statistical information of the test results in order to compare the case with only one reinforcement learning (RL) agent and the case with all agents are RL agent. To begin with, since our proposed method is for the coordination of multiple RL agents, SQDQN may be better for the case of only a single RL agent. However, when the proposed method was placed upstream (such as at the Distributor or Manufacturer) in the supply chain, the results were in some cases relatively better than SRDQN.

Tables 4 and 5 show statistical information of total cost value for the system with a single RL agent, and the other co-players use the Sterman formula. The values in Tables 4 and 5 represent the average values of the various statistics for the cost values of six instances. The Agent column in Tables 4 and 5 indicate at which facility a single RL agent is placed in the beer game, where R, W, D and M denote the Retailer, Warehouse, Distributor and Manufacturer, respectively. For the random demand fluctuations in Table 4, the lowest average cost was generally achieved when the SRDQN was used at the Retailer (R) position. This trend was consistent across all RL methods. In the case of relatively continuous demand fluctuations in Table 5, the lowest average cost was observed when the RL agent was at the Warehouse (W) position for the SRDQN. This trend was also consistent across all RL methods. For both random and relatively continuous demand fluctuations, placing the RL agent more downstream (such as at the Retailer or Warehouse) tended to result in lower average costs. This suggests that in the beer game, managing demand uncertainty closer to the end consumer (downstream) could be more beneficial in minimizing costs.

We evaluated the results in Tables 2 and 4, and in Tables 3 and 5, for comparing the case where there is only one RL agent and the case where all agents are RL agents. In Tables 2 and 4, which represent the case of random demand fluctuations, we observed that when the single RL agent was placed at the Retailer (R) position using the SRDQN, the system achieved the lowest average cost of 7.896×10^4 (Table 4). However, when all agents were RL agents and use the SRDQN+RDPM, there was only a minor increase in the average cost to 8.212×10^4 (Table 2), indicating that having all agents as RL agents might be a

Table 4: Statistical information on the total cost value when only one RL agent is used with relatively continuous demand fluctuations (6 instances)

RL method	Agent	Min	Max	Average	Variance
SRDQN	R	$\mathbf{5.178 \times 10^4}$	$\mathbf{2.017 \times 10^5}$	$\mathbf{7.896 \times 10^4}$	7.888×10^8
	W	$\mathbf{6.494 \times 10^4}$	2.862×10^5	$\mathbf{9.212 \times 10^4}$	8.110×10^8
	D	$\mathbf{1.212 \times 10^5}$	2.540×10^5	1.662×10^5	4.545×10^8
	M	2.042×10^5	3.543×10^5	2.769×10^5	7.241×10^8
SRDQN+DR	R	9.247×10^4	2.957×10^5	1.356×10^5	2.803×10^8
	W	1.232×10^5	$\mathbf{2.587 \times 10^5}$	1.709×10^5	3.522×10^8
	D	2.017×10^5	2.587×10^5	2.490×10^5	6.531×10^8
	M	2.017×10^5	3.650×10^5	2.490×10^5	6.531×10^8
SRDQN+RDPM	R	9.777×10^4	2.515×10^5	1.581×10^5	$\mathbf{2.165 \times 10^8}$
	W	9.421×10^4	2.925×10^5	1.069×10^5	4.539×10^8
	D	1.229×10^5	2.626×10^5	1.480×10^5	6.634×10^8
	M	1.969×10^5	3.662×10^5	2.199×10^5	6.177×10^8
SRDQN+TSRDPM-τ1	R	9.848×10^4	2.255×10^5	1.278×10^5	3.354×10^8
	W	9.308×10^4	3.096×10^5	1.065×10^5	3.870×10^8
	D	1.253×10^5	$\mathbf{2.211 \times 10^5}$	1.397×10^5	$\mathbf{1.179 \times 10^8}$
	M	$\mathbf{1.917 \times 10^5}$	3.365×10^5	$\mathbf{2.063 \times 10^5}$	3.733×10^8
SRDQN+TSRDPM-τ5	R	1.069×10^5	2.284×10^5	1.499×10^5	2.621×10^8
	W	1.031×10^5	2.758×10^5	1.168×10^5	$\mathbf{2.874 \times 10^8}$
	D	1.260×10^5	2.300×10^5	$\mathbf{1.383 \times 10^5}$	1.330×10^8
	M	1.929×10^5	$\mathbf{3.333 \times 10^5}$	2.071×10^5	$\mathbf{3.442 \times 10^8}$

Table 5: Statistical information on the total cost value when only one RL agent is used with relatively continuous demand fluctuations (6 instances)

RL method	Agent	Min	Max	Average	Variance
SRDQN	R	$\mathbf{5.700 \times 10^4}$	$\mathbf{2.490 \times 10^5}$	$\mathbf{1.130 \times 10^5}$	1.090×10^9
	W	$\mathbf{7.935 \times 10^4}$	$\mathbf{2.363 \times 10^5}$	$\mathbf{1.004 \times 10^5}$	4.007×10^8
	D	$\mathbf{1.526 \times 10^5}$	2.864×10^5	1.941×10^5	$\mathbf{4.655 \times 10^8}$
	M	2.947×10^5	$\mathbf{4.635 \times 10^5}$	3.965×10^5	1.037×10^9
SRDQN+DR	R	1.187×10^5	3.117×10^5	2.019×10^5	1.203×10^9
	W	1.118×10^5	2.702×10^5	1.748×10^5	1.047×10^9
	D	1.629×10^5	2.869×10^5	2.285×10^5	5.960×10^8
	M	2.979×10^5	4.979×10^5	4.552×10^5	1.160×10^9
SRDQN+RDPM	R	1.082×10^5	2.863×10^5	1.885×10^5	5.014×10^8
	W	1.073×10^5	2.601×10^5	1.354×10^5	4.106×10^8
	D	1.624×10^5	$\mathbf{2.790 \times 10^5}$	2.218×10^5	7.457×10^8
	M	3.018×10^5	5.089×10^5	4.238×10^5	1.751×10^9
SRDQN+TSRDPM-τ1	R	1.047×10^5	3.260×10^5	1.874×10^5	3.254×10^8
	W	1.127×10^5	2.590×10^5	1.340×10^5	$\mathbf{3.004 \times 10^8}$
	D	1.590×10^5	2.921×10^5	$\mathbf{1.874 \times 10^5}$	5.386×10^8
	M	$\mathbf{2.749 \times 10^5}$	4.646×10^5	$\mathbf{2.932 \times 10^5}$	8.938×10^8
SRDQN+TSRDPM-τ5	R	1.033×10^5	2.657×10^5	1.793×10^5	$\mathbf{1.958 \times 10^8}$
	W	1.151×10^5	2.465×10^5	1.386×10^5	3.214×10^8
	D	1.609×10^5	2.968×10^5	1.993×10^5	8.321×10^8
	M	2.784×10^5	4.790×10^5	2.968×10^5	$\mathbf{8.876 \times 10^8}$

viable strategy. In Tables 3 and 5, which represent the case of relatively continuous demand fluctuations, the lowest average cost was achieved when all agents were RL agents using the

SRDQN+TSRDPM-τ1, yielding an average cost of 8.577×10^4 (Table 3). This implies that a complete RL agents setup can bring significant advantages in terms of cost minimization when dealing with relatively continuous demand fluctuations. In summary, the placement of RL agents in the beer game and the choice of whether all agents should be RL agents significantly influence the system's performance. A downstream position of the RL agent (like the Retailer) seemed generally favorable for reducing the average cost, particularly in the case of random demand fluctuations. However, the best placement of RL agents can change based on the nature of the demand fluctuations and the specific RL method used. This emphasizes the importance of adaptability in MARL setups. Furthermore, a system where all agents are RL agents can result in the lowest average costs, indicating the possibility of a MARL setting in the beer game.

5 Conclusion

We proposed new reward shaping techniques in SRDQN for the beer game to improve the cooperation among multiple agents and the resulting inventory costs. The issues with conventional method were that there was room for improvement in learning stability, and the application to MARL had not been considered. We applied the reward design method based on the VCG mechanism to SRDQN with the aim of improving cooperation in a system with multiple reinforcement learning agents. In addition, in order to deal with a variety of fluctuations in consumer demand, we presented different types of reward design depending on the properties of the demand. In the evaluation using a beer game with two types of consumer demand when MARL is used, SRDQN+RDPM was the best solution in the case of random demand, and SRDQN+TSRDPM-τ1 had the best cost minimization performance in the case of relatively continuous demand. Compared with the previous method SRDQN, our experimental evaluation revealed that the proposed method improved learning stability and reduced cost values in the play results for the systems consisting only of reinforcement learning agents. Furthermore, by modifying the reward design in response to the properties of demand fluctuations, we achieved more cost values reduction. Our study presented how MARL can be applied to inventory optimization in a fundamental game of supply chain management. We believe that our approach can be applied to other supply chain management problems in which supply chain participants behave unpredictably or selfishly. Future work will include integration with other learning efficiency methods and detailed analysis under more realistic supply chain structure and demand fluctuations.
Acknowledgement:

6 Acknowledgement

This work was supported in part by JSPS KAKENHI Grant Number JP22H03647.

References

[1] Adrian K. Agogino and Kagan Tumer. Analyzing and Visualizing Multiagent Rewards in Dynamic and Stochastic Domains. *Autonomous Agents and Multi-Agent Systems*, 17(2):320 – 338, 2008.

[2] Sushrut Bhalla, Sriram Ganapathi Subramanian, and Mark Crowley. Deep Multi Agent Reinforcement Learning for Autonomous Driving. In *Canadian Conference on Artificial Intelligence 2020: Advances in Artificial Intelligence*, pages 67–78, 2020.

[3] Sven Gronauer and Klaus Diepold. Multi-Agent Deep Reinforcement Learning: A Survey. *Artificial Intelligence Review*, 55(2):895 – 943, 2022.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, Conference Track Proceedings*, 2015.

[5] Hau Lee, V. Padmanabhan, and Seungjin Whang. Comments on "Information Distortion in a Supply Chain: The Bullwhip Effect". *Management Science*, 50:1887–1893, 2004.

[6] Hau Lee, V. Padmanabhan, and Seungjin Whang. Information Distortion in a Supply Chain: The Bullwhip Effect. *Management Science*, 43:546–558, 2004.

[7] Benda M., Jagannathan V., and Dodhiawalla R. On optimal cooperation of knowledge sources. *Technical Report BCS-G2010-28*, 1985.

[8] Natsuki Matsunami, Shun Okuhara, and Takayuki Ito. Reward Design for Multi-Agent Reinforcement Learning with a Penalty Based on the Payment Mechanism. *Transaction of the Japanese Society for Artificial Intelligence*, 36(5):AG21–H_1–11, 2021.

[9] Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[10] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, page 807 – 814, 2010.

[11] Authors' names omitted for review. Applying Reward Design Based on Payment Mechanism to Shaped-Reward DQN for Beer Game. In *2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 384–390, 2022.

[12] Afshin Oroojlooy et al. A Deep Q-Network for the Beer Game: Deep Reinforcement Learning for Inventory Optimization. https://github.com/OptMLGroup/DeepBeerInventory-RL.

[13] Afshin Oroojlooy jadid, Mohammadreza Nazari, Lawrence Snyder, and Martin Takáč. A Deep Q-Network for the Beer Game: Deep Reinforcement Learning for Inventory Optimization. *Manufacturing & Service Operations Management*, 24(1):285–304, 2021.

[14] Tim Roughgarden. Algorithmic Game Theory. *Communications of the ACM*, 53(7):78 – 86, 2010.

[15] John D. Sterman. Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Experiment. *Management Science*, 35:321–339, 1989.

[16] Oriol Vinyals et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350 – 354, 2019.