NEURON AS AN AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing multi-agent reinforcement learning (MARL) communication methods have relied on a trusted third party (TTP) to distribute incentive to agents, leaving them inapplicable in peer-to-peer environments. This paper proposes incentive distribution using *Neuron as an Agent* (NaaA) in MARL without a TTP with two key ideas: (i) inter-agent reward distribution and (ii) auction theory. Auction theory is introduced because inter-agent reward distribution is insufficient for optimization. Agents in NaaA maximize their profits (the difference between reward and cost) and, as a theoretical result, the auction mechanism is shown to have agents autonomously evaluate counterfactual returns as the values of other agents. NaaA enables representation trades in peer-to?peer environments, ultimately regarding unit in neural networks as agents. Finally, numerical experiments (a single–agent environment from Open AI gym and a multi-agent environment from ViZDoom) confirm that NaaA framework optimization leads to better performance in RL.

1 INTRODUCTION

Now that intelligence with reinforcement learning has been shown to be superior to humans (Tesauro, 1995; Mnih et al., 2015; Silver et al., 2016), reinforcement learning is expected to expand into industry applications such as stock trading, autonomous cars, smart grids, and IoT. In future industrial applications, numerous companies will own agents used to improve their revenues. Such a situation can be considered as each agent independently solving the problems of a partially observed Markov decision process (POMDP).

These company agents are designed to independently maximize their own rewards; however, if such agents could exchange information, the overall revenue of stakeholders would increase. As each agent has limited visibility of the environment, information exchanges between agents would be helpful in solving their independent tasks. Thus, this paper aimed to actualize a society where stakeholders with conflicting interests willingly trade information.

This paper investigated the communication situation in multi-agent reinforcement learning (MARL) using several existing methods such as R/DIAL (Foerster et al., 2016) and CommNet (Sukhbaatar et al., 2016). CommNet is a state-of-the-art MARL method that considers communication between agents and features learning among agents with backpropagation.

When considering conditions in MARL such that different stakeholders have created different agents that communicate with each other, an incentive distribution design (e.g., monetary payment) and a framework without a *trusted third party* (TTP) are required. TTP is a neutral administrator that performs reward distributions for all participants, and is supposed implicitly by most existing literature regarding MARL (Agogino & Tumer, 2006; Foerster et al., 2016; Sukhbaatar et al., 2016). Although there is a requirement for TTP neutrality towards all participants, several peer-to-peer trade configurations, such as inter-industry and -country trades, cannot place TTP. If reward distribution is performed by an untrusted third party, the rewards for partial participants may be undesirably altered.

To the best of our knowledge, no existing literature discusses reward distributions in the configuration described above. Because CommNet assumes an environment that distributes a uniform reward to all the agents, if the distributed reward is in limited supply (such as money), it causes the *Tragedy of the Commons* (Lloyd, 1833), where the reward of contributing agents will be reduced due to the participation of free riders. Although there are several MARL methods for distributing rewards ac-

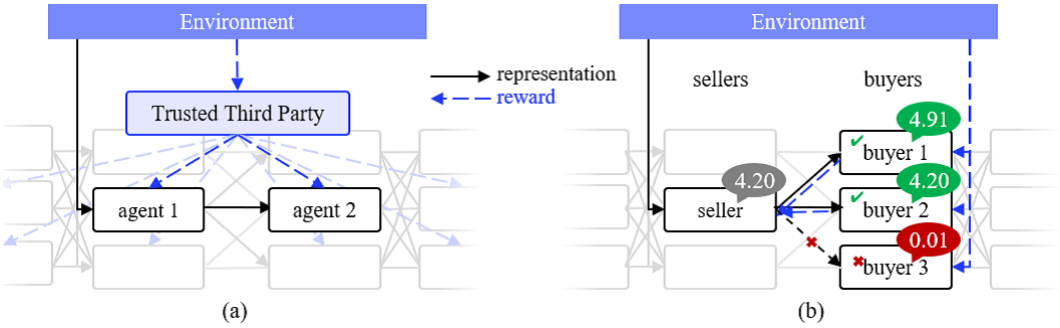

Figure 1: A schematic illustration of reward distribution models in MARL. **(a)** Centralized reward distribution model (Agogino & Tumer, 2006; Sukhbaatar et al., 2016; Foerster et al., 2016; 2017). They should suppose TTP to distribute the optimal reward to the agents. **(b)** Inter-agent reward distribution model (our model). Some agents receive reward from the environment directly, and redistribute to other agents. The idea to determine the optimal reward without TTP is playing auction game among the agents.

cording to agents' contribution such as QUICR (Agogino & Tumer, 2006) and COMA (Sukhbaatar et al., 2016), they suppose the existence of TTP and hence cannot be applied to the situation investigated here.

The proposed method, *Neuron as an Agent* (NaaA), extends CommNet to actualize incentive distributions in MARL without TTP based on two key ideas: (i) inter-agent reward distribution and (ii) auction theory. Auction theory was introduced because inter-agent reward distributions were insufficient for optimization. Agents in NaaA maximize *profit*, the difference between their received rewards and the costs which they redistribute to other agents. If the framework is naively optimized, a trivial solution is obtained where agents reduce their costs to zero to maximize profits. Then, NaaA employs the auction theory in game design to prevent costs from dropping below their necessary level. As a theoretical result, we show that agents autonomously evaluate the *counterfactual return* as values of other agents. The counterfactual return is equal to the discounted cumulative sum of counterfactual reward (Agogino & Tumer, 2006) distributed by QUICR and COMA. NaaA actualizes reward distributions, making it more of a pareto improvement than inter-agent reward distribution method.

NaaA enables representation trades in peer-to-peer environments and, ultimately, regards neural network units as agents. As NaaA is capable of regarding units as agents without losing generality, this setting was utilized in the current study. The concept of the proposed method is illustrated in Figure 1.

An environment extending ViZDoom (Kempka et al., 2016), a POMDP environment, to MARL was used for the experiment. Two agents, a cameraman sending information and a main player defeating enemies with a gun, were placed in the environment. Results confirmed that the cameraman learned cooperative actions for sending information from dead angles (behind the main player) and outperformed CommNet in score.

Interestingly, NaaA can apply to single- and multi-agent settings, since it learns optimal topology between the units. *Adaptive DropConnect* (ADC), which combines DropConnect (Wan et al., 2013) (randomly masking topology) with an adaptive algorithm (which has a higher probability of pruning connections with lower counterfactual returns) was proposed as a further application for NaaA. Experimental classification and reinforcement learning task results showed ADC outperformed DropConnect.

The remainder of this paper is organized as follows. First, the two key ideas are described: inter-agent reward distribution and auction theory. After related works are introduced, the experimental results are shown in ViZDoom. Subsequently, ADC and its experimental results are shown to demonstrate a further application. Finally, a conclusion ends the paper.

2 PROBLEM DEFINITION

Suppose there are N agents interacting to an environment. Our goal is to maximize the discounted cumulative reward which the system will obtain from the external environment.

$$G = \sum_{i=1}^{N} \left[\sum_{t=0}^{T} \gamma^t R_{it}^{\mathrm{ex}} \right], \tag{1}$$

where R_{it}^{ex} is an external reward which i-th agent obtains at t, and $\gamma \in [0, 1]$ is the discount rate and T is the terminal time.

3 PROPOSED METHOD

3.1 INTER-AGENT REWARD DISTRIBUTION

Some agents get reward from the environment, and distribute it to other agents as incentive to give precise information. The reward is limited so that if an agent distribute ρ reward, the agents' reward subtracted by ρ such as currency. For the reason, other agents for an agent itself can be interpreted as another environment which give a reward $-\rho$ instead of an observation x. We name the another environment as *internal environment* and name the original environment as an *external environment*.

Similarly to CommNet (Sukhbaatar et al., 2016), we assume that the communication protocol between the agent is a continuous quantity such as a vector, and the content can be trained by backpropagation. Hence, we can interpret the multi-agent communication as a huge neural network. Therefore, we can interpret a unit as an agent without loss of generality. This framework can single-agent reinforcement learning as well as multi-agent reinforcement learning.

A typical artificial neural network is a directed graph $\mathfrak{G} = (\mathcal{V}, \mathcal{E})$ among the units. $\mathcal{V} = \{v_1, \ldots, v_N\}$ is a set of the units. $\mathcal{E} \subset \mathcal{V}^2$ is a set of edges representing connections between two units. If $(v_i, v_j) \in \mathcal{E}$, then connection $v_i \to v_j$ holds, indicating that v_j observes activation of v_i. We denote activation of the unit v_i at time t as $x_{it} \in \mathbb{R}$. Additionally, we designate a set of units which unit i connects to as $N_i^{\mathrm{out}} = \{j | (v_i, v_j) \in \mathcal{E}\}$ and a set of units which unit i is connected from as $N_i^{\mathrm{in}} = \{j | (v_j, v_i) \in \mathcal{E}\}$. We denote $N_i = N_i^{\mathrm{in}} \cup N_i^{\mathrm{out}}$.

NaaA interprets v_i as an agent. Therefore, $\mathfrak{G}$ is a multi-agent system. An environment for v_i comprises an environment that the multi-agent system itself touches and a set of the unit to which v_i directly connects: $\{v_i \in \mathcal{V} | i \in N_i\}$. We distinguish both environments by naming the former as an external environment, and by naming the latter as an internal environment. v_i will receive rewards from both environments. We add the following assumption for characterizing v_i.

- N1: (Selfishness) The utility which an v_i wants to maximize is its own return (cumulative discounted reward): $G_{it} = \sum_{k=0}^{T} \gamma^k R_{i,t+k}$.
- N2: (Conservation) The summation of internal reward over $\mathcal{V}$ equals to 0. Hence, the summation of a reward by which $\mathcal{V}$ will receive both an internal and external environment R_{it} are equivalent to reward R_t^{ex}, which the entire multi-agent system receives from the external environment: $\sum_{i=1}^{n} R_{it} = \sum_{i=1}^{n} R_{it}^{\mathrm{ex}}$.
- N3: (Trade) The v_i receives internal reward ρ_{jit} from $v_j \in \mathcal{V}$ in exchange of activation signal x_i before transferring the signal to the unit. At the same time, ρ_{jit} is subtracted from the reward of v_j.
- N4: (NOOP) v_i has NOOP (no operation), for which the return is $\delta > 0$ as an action. With NOOP, the unit inputs nothing and outputs nothing.

The social welfare function (total utility of the agents) G^{all} is equivalent to the objective function G. That is,

$$G^{\mathrm{all}} = \sum_{i=1}^{n} G_{it} = \sum_{i=1}^{n} \left[\sum_{k=0}^{T} \gamma^t R_{it} \right] = \sum_{k=0}^{T} \left[\gamma^t \sum_{i=1}^{n} R_{it} \right]. \tag{2}$$

From N2, $G^{\mathrm{all}} = G$ holds.

3.1.1 DISCOUNTED CUMULATIVE PROFIT MAXIMIZATION

We denote the external reward by which unit v_i receives at time step t as R_{it}^{ex}, where $\sum_{i=1}^{n} R_{it}^{\mathrm{ex}} = R_t^{\mathrm{ex}}$ holds. From N3, reward R_{it}, which v_i receives at t can be written as the following.

$$R_{it} = R_{it}^{\mathrm{ex}} + \sum_{j \in N_i^{\mathrm{out}}} \rho_{jit} - \sum_{j \in N_i^{\mathrm{in}}} \rho_{ijt}. \tag{3}$$

The equation is divided into positive terms and a negative term, we name the former as revenue, and the latter as cost, and denote them respectively as $r_{it} = R_{it}^{\mathrm{ex}} + \sum_{j \in N_i^{\mathrm{out}}} \rho_{jit}$, $c_{it} = \sum_{j \in N_i^{\mathrm{in}}} \rho_{ijt}$. We name R_{it} as profit.

v_i maximizes the cumulative discounted profit G_{it} represented as

$$G_{it} = \sum_{k=0}^{T} \gamma^k R_{i,t+k} = \sum_{k=0}^{T} \gamma^k (r_{i,t+k} - c_{i,t+k}) = r_t - c_t + \gamma G_{i,t+1}. \tag{4}$$

G_{it} is unobserved unless the time is reached at the end of the episodes. Because prediction based on the current value is needed to select the optimal actions, we approximate G_{it} with value function $V_i^{\pi_i}(s_{it}) = \mathbb{E}_{\pi_i}\left[G_{it} \mid s_{it}\right]$ where $s_{it} \in \mathcal{S}_{\mathrm{O}}$. In this case, the following equation holds.

$$V_i^{\pi_i}(s_{it}) = r_{it} - c_{it} + \gamma V_i^{\pi_i}(s_{i,t+1}), \tag{5}$$

Therefore, we need only consider maximization of revenue, the value function, and cost minimization. $R_{it} > 0$, i.e., $r_{it} > c_{it}$ indicates that the unit gives the additional value to the obtained data. The unit acts NOOP because $V_i^{\pi_i}(s_{it}) \leq 0 < \delta$ if $R_{it} \leq 0$ for all t because of N4.

3.1.2 SOCIAL DILEMMA

STUB

3.2 AUCTION THEORY

We introduce mechanism design because, unlike several existing studies (Sukhbaatar et al., 2016), NaaA assumes that all agents are not cooperative but selfish. If we naively optimize the optimization problem of NaaA, then we obtain the trivial solution that the internal rewards will converge to 0, and that all the units except of the output units become NOOP. This phenomena occurs regardless the network topology $\mathfrak{G}$ as any nodes have no incentive to send payment ρ_{ijt} to other units. Therefore, the multi-agent system should select the action with no information. It is equivalent to taking an action randomly. For that reason, the external reward R_t^{ex} shrinks markedly.

3.2.1 ENVY-FREE AUCTION

To maximize the overall reward, our objective function, we borrow the idea from the digital goods auction. The auction theory belongs to mechanism design. It is intended to unveil the true price of goods. Digital goods auction is one mechanism from auction theory. It is target to copyable goods without cost, such as digital books and music.

Although several variations of digital goods auctions exist, we use an envy-free auction (Guruswami et al., 2005) because it requires a simple assumption: the same goods have one price simultaneously. In NaaA, it can be represented as the following assumption:

N5: (Law of one price) If $\rho_{j_1,i,t}, \rho_{j_2,i,t} > 0$, then $\rho_{j_1,i,t} = \rho_{j_2,i,t}$.

The assumption above indicates that ρ_{jit} takes either 0 or a positive value depending on i at a same timing t. Therefore, we name the positive side v_i's *price*, and denote as q_{it}.

We present the envy-free auction process at the left of Figure 2. It shows the negotiation process between one unit in sending activation and a group of units that buy the activation. The negotiation performed per time step in RL. We name the unit in sending activation as a seller, and units in buying activation as buyers. First, the buyer bids the unit in bidding price b_{jit} (**1**). Next, the seller decides the optimal price $\hat{q}_{it}$, and performs allocation (**2**). Payment occurs if b_{ijt} exceeds q_{jt}. In

this case, $\rho_{jit} = H(b_{jit} - q_{it})q_{it}$ holds where $H(\cdot)$ is a step function. Besides, we define $g_{jit} = H(b_{jit} - q_{it})$ and name it *allocation*. After allocation, the buyers perform payment as $\rho_{jit} = g_{jit}\hat{q}_{it}$ (**3**). Eventually, seller earns The seller only sends activation x_i to the allocated buyers (**4**). A buyer which cannot receive the activation approximates x_i with $\mathbb{E}_\pi[x_i]$.

3.2.2 Theoretical Analysis

In the following, we discuss revenue, cost, and value functions based on Eq:(5).

Revenue: The revenue of a unit is given as

$$r_{it} = \sum_{j \in N_i^{\text{out}}} g_{jit} q_{it} + R_i^{\text{ex}} = q_i d_{it} + R_i^{\text{ex}}, \tag{6}$$

where $d_{it} = \sum_{j \in N_i^{\text{out}}} g_{jit}$ is a count of units for which the bidding price for q_{it} is greater than or equal to q_{it}, designated as demand. q_{it} maximizing the equation is designated as the optimal price. It is denoted as $\hat{q}_{it}$. Because R_i^{ex} is independent of q_t, the optimal price $\hat{q}_{it}$ is given as

$$\hat{q}_{it} = \underset{q \in [0,\infty)}{\operatorname{argmax}}\, q d_{it}(q). \tag{7}$$

We present the curve of r_{it} on the right side of Figure 2.

Cost: The cost is an internal reward that the unit should pay to other units. It is represented as shown below.

$$c_{it} = \sum_{j \in N^{\text{in}}} g_{ijt} q_{jt} = \mathbf{g}_{it}^{\mathrm{T}} \mathbf{q}_t, \tag{8}$$

where $\mathbf{g}_{it} = (g_{i1t}, \ldots, g_{iNt})^{\mathrm{T}}$ and $\mathbf{q}_t = (q_{1t}, \ldots, q_{Nt})^{\mathrm{T}}$. Although c_{it} itself is minimized when $b_{ijt} = 0$, this represents a tradeoff with the following value function.

Value Function: The activation x_{it} depends on input from the units in N_i^{in} affecting the bidding price from units in N_i^{out}. If we minimize b_{ijt} and let $b_{ijt} = 0$, then the purchase of activation fails, and the reward the unit can obtain from the units to which the unit connects becomes lower in the future.

We consider effects for value functions in the cases when a unit succeeds in purchasing v_j or not. We approximate the value function as a linear function of $\mathbf{g}_{it}$:

$$V_i^{\pi_i}(s_{i,t+1}) \approx \mathbf{o}_{it}^{\mathrm{T}} \mathbf{g}_{it} + V_{i,t+1}^0, \tag{9}$$

where $\mathbf{o}_{it}$ is a parameter implemented as difference between two returns of v_i whether we observe x_i or not. As $\mathbf{o}_{it}$ is equivalent to the cumulative discount value of counterfactual reward (Agogino & Tumer, 2006), we name it *counterfactual return*. V_{it}^0 is a constant independent of $\mathbf{g}_{it}$ and we name it *blind value function* as it is equivalent to value function when v_i takes action without any observation $x_1, \ldots, x_N$.

Therefore, the optimization problem is presented below.

$$\max_{\mathbf{a}} Q_i(s_{it}, \mathbf{a}) = \max_q q d_{it}(q) - \min_{\mathbf{b}} \mathbb{E}_{\hat{\mathbf{q}}_t}\left[\mathbf{g}_{it}(\mathbf{b})^{\mathrm{T}}(\hat{\mathbf{q}}_t - \gamma \mathbf{o}_{it})\right] + \text{const.}, \tag{10}$$

where $\mathbf{a} = (\mathbf{b}, q)$. Note that $\mathbf{g}_{it} = H(\mathbf{b} - \mathbf{q}_t)$. We take the expectation $\mathbb{E}_{\hat{\mathbf{q}}_t}[\cdot]$ because the asked price $\hat{\mathbf{q}}_t$ is unknown for v_i, except for $\hat{q}_{it}$, and $g_{iit} = 0$.

Then, what is bidding price b_{it} to maximize return? The following theorem holds.

Theorem 3.1. *(Truthfulness) the optimal bidding price for maximizing return is* $\hat{\mathbf{b}}_{it} = \gamma \mathbf{o}_{it}$.

See the Appendix for the proof.

That is, the unit should only consider its counterfactual return (!). If $\gamma = 0$, the case is equivalent to a case without auction. Hence, the bidding value raises if each unit consider long-time reward. Consequently, in the mechanism of NaaA, the unit obeys as if performing valuation to the other units, and declares the value truthfully.

Then, the following corollary holds:

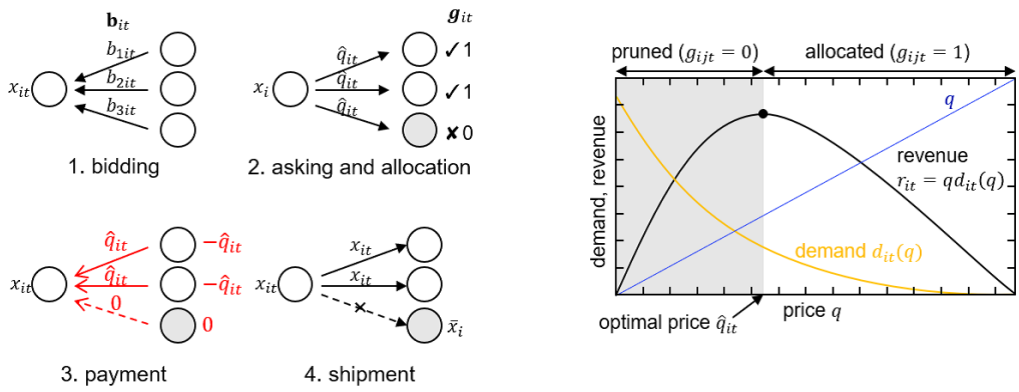

Figure 2: **Left**: The process of trade in an envy-free auction. **Right**: A price determination curve for a unit. Revenue of a unit is a product of monotonically decreasing demand and price. The price maximizing the revenue is the optimal price.

Algorithm 1 NaaA: inter-agent reward distribution with envy-free auction

for $t = 1$ **to** T **do**
  Compute a bidding price for every edge: **for** $(v_j, v_i) \in \mathcal{E}$ **do** $b_{ijt} \leftarrow Q^{\pi_i}(\mathbf{s}_{it}, \mathbf{e}_j) - Q^{\pi_i}(\mathbf{s}_{it}, \mathbf{0})$
  Compute an asking price for every node: **for** $v_i \in \mathcal{V}$ **do** $\hat{q}_{it} \leftarrow \operatorname{argmax}_{q \in [0,\infty)} q d_{it}(q)$.
  for $(v_i, v_j) \in \mathcal{E}$ **do**
    Compute allocation: $g_{jit} \leftarrow H(b_{jit} - \hat{q}_{it})$
    Compute the price the agent should pay: $\rho_{jit} \leftarrow g_{jit}\hat{q}_{it}$
  end for
  Make a payment: **for** $v_i \in \mathcal{V}$ **do** $R_{it} \leftarrow \sum_{j \in N_i^{\text{out}}} \rho_{jit} - \sum_{j \in N_i^{\text{in}}} \rho_{ijt}$,
  Make a shipment: **for** $v_i \in \mathcal{V}$ **do** $\tilde{x}_{ijt} = g_{ijt} x_{ijt} + (1 - g_{ijt})\bar{x}_{ijt}$
  for $v_i \in \mathcal{V}$ **do**
    Observe external state $\mathbf{s}_{it}^{ex}$
    $\mathbf{s}_{it} \leftarrow (\mathbf{s}_{it}^{ex}, \tilde{\mathbf{x}}_{it}, \boldsymbol{\theta}_i)$, where $\tilde{\mathbf{x}}_{it} = (\tilde{x}_{i1t}, \ldots, \tilde{x}_{int})^{\mathrm{T}}$ and $\boldsymbol{\theta}_i$ is v_i's parameter.
    Sample action $a_{it}^{ex} \sim \pi_i^{ex}(\mathbf{s}_{it})$
    Receive external reward $R_{it} \leftarrow R_{it} + R_{it}^{ex}(a_{it}^{ex})$
    Update Q^{π_i} under the manner of Q-learning by calculating the time difference (TD)-error
  end for
end for

Corollary 3.1. *The Nash equilibrium of an envy-free auction* $(\mathbf{b}_{it}, q_{it})$ *is* $(\mathbf{o}_{it}, \operatorname{argmax}_q q d_{it}(q))$.

The remaining problem is how to predict $\mathbf{o}_t$. Although several method can be applied to this problem, we use Q-learning to predict $\mathbf{o}_t$. As $\mathbf{o}_{it}$ is difference of two Qs, we approximate each of Q. Other RL such as SARSA and A3C can be employed. We parametrize the state with a vector $\mathbf{s}_t$ which contains input and weight. ϵ-greedy policy with Q-learning typically suppose that discrete actions So, as an action, we employ allocation g_{ijt} instead of $\mathbf{b}_{it}$ and q_{it}.

Algorithm The overall algorithm is shown in Algorithm 1.

3.3 Neuron as an Agent

Stub

4 RELATED WORKS

Existing multi-agent reinforcement learning (MARL) communication methods have relied on a trusted third party (TTP) to distribute incentive to agents, leaving them inapplicable in peer-to-peer environments. R/DIAL (Foerster et al., 2016) is a communication method for deep reinforcement learning, which train the optimal communication among the agent with Q-learning. It focuses on that paradigm of centralized planning. CommNet (Sukhbaatar et al., 2016), which exploits the characteristics of a unit that is agnostic to the topology of other units, employs backpropagation to train multi-agent communication. Instead of reward $R(a_t)$ of an agent i for actions at t a_t, QUICR-learning (Agogino & Tumer, 2006) maximizes counterfactual reward $R(a_t) - R(a_t - a_{it})$, the difference in the case of the agent i takes an action a_{it} (a_t) and not $(a_t - a_{it})$. COMA (Foerster et al., 2017) also maximizes counterfactual rewards in an actor–critic setting. CommNet, QUICR and COMA have a centralized environment for distributing rewards through a TTP. Hence, these methods above cannot be applied into peer-to-peer setting. In contrast, NaaA does not rely on a TTP, and hence, each agent calculates its reward.

While inter-agent reward distribution has not been considered in the context of communication, trading agents have been considered in other contexts. Trading agent competition (TACs), competitions for trading agent design, have been held in various locations regarding topics such as smart grids (Ketter et al., 2013), wholesale (Kuate et al., 2013), and supply chains (Pardoe & Stone, 2009), yielding innumerable trading algorithms such as Tesauro's bidding algorithm (Tesauro & Bredin, 2002) and TacTex'13 (Urieli & Stone, 2014). Since several competitions employed an auction as optimal price determination mechanism (Wellman et al., 2001; Stone et al., 2003), using auctions to determine optimal prices is now a natural approach. Unfortunately, these existing methods cannot be applied to the present situation. First, their agents did not communicate because the typical purpose of a TAC is to create market competition between agents in a zero-sum game. Secondly, the traded goods are not digital goods but instead goods in limited supply, such as power and tariffs. Hence, this is the first paper to introduce inter-agent reward distribution to MARL communications.

Auction theory is discussed in terms of mechanism design (Myerson, 1983), also known as inverse game theory. Second–price auctions (Vickrey, 1961) are auctions including a single product and several buyers. In this paper, a digital goods auction (Guruswami et al., 2005) was used as an auction with an infinite supply. Several methods extend digital goods auction to address collusion, including the consensus estimate (Goldberg & Hartline, 2003) and random sample auction (Goldberg et al., 2006), which can be used to improve our method.

This paper is also related to DropConnect in terms of controlling connections between units. Adaptive DropConnect (ADC), proposed in a later section of this paper as a further application, extends the DropConnect (Wan et al., 2013) regularization technique. The finalized idea of ADC (which uses a skew probability correlated to the absolute value of weights rather than dropping each connection between units by a constant probability) is closer to Adaptive DropOut (Ba & Frey, 2013), although their derivation differs. The adjective "adaptive" is added with respect to the method. Neural network optimizing using RL was investigated by Andrychowicz et al. (2016); however, their methods used a recurrent neural network (RNN) and are therefore difficult to implement, whereas the proposed method is RNN-free and forms as a layer. For these reasons, its implementation is simple and fast and it also has a wide area of applicability.

5 EXPERIMENT

We confirmed our reward distribution works as expected with a multi-agent setting in ViZDoom (Kempka et al., 2016), an emulator of Doom which have a map editor. In the setting, additional agents complement the main player. A player in ViZDoom environment should seek the enemy in the map, and then defeat the enemy.

5.1 SETUP

We used a scenario based on Defend the Center (DtC), provided by ViZDoom platform. In DtC, players are placed in the center of a field of circle. They attack enemies that come from the wall. The game has two players: a main player and a cameraman. Alhough the main player can attack

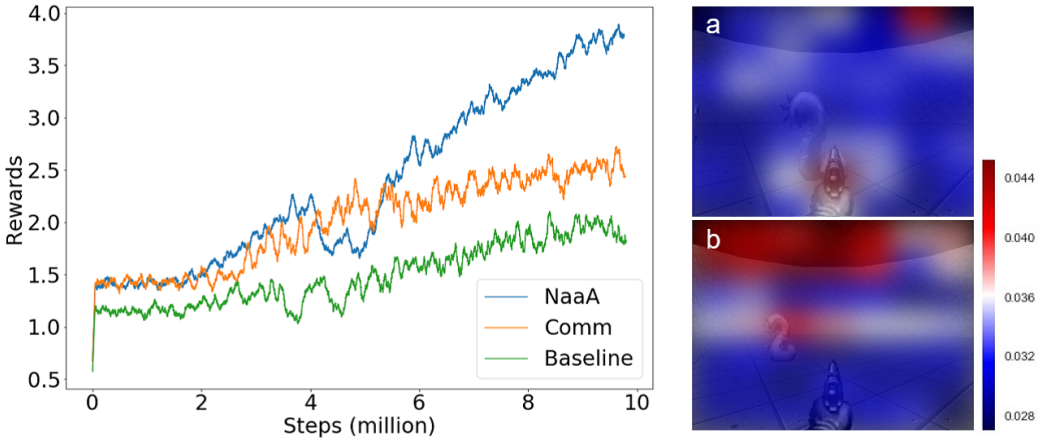

Figure 3: **Left:** Learning curve for the multi-agent task of VizDoom. Our method based on NaaA outperforms the other two methods: baseline and Comm DQN. **Right:** Visualizing reward from the main player to the cameramann shows us what is important information for the main player: (a) The cameraman sees the pistol. (b) The cameraman sees the point which enemy appear and come closer.

the enemy with bullets, the cameraman has no way to attack, and only scouts for the enemy. The action space for the main player is the combination of { attack, turn left, turn right }. Therefore, the total number of actions is $2^3 = 8$. The cameraman has two possible actions: { turn left, turn right }. Although the players can only change direction, they cannot move on the field. The enemy will die if have the attack (bullet) from the main player once, then player receives +1. As a default on an episode, the ammunition amount is 26. The main player will die if under attack from the enemy to the extent that health becomes 0, then the player receives -1. The cameraman will not die if attacked by the enemy. The episode will terminate when the maim player dies, or after 525 steps have elapsed.

5.2 Model

We compared three models: the proposed method and two comparison targets.

Baseline: DQN without communication. The main player learns standard DQN with the perspective that the player is viewing. Because the cameraman does not learn, the player continues to move randomly.

Comm: DQN with communication. The main player learns DQN with two perspectives: the player's own and the cameraman's. The communication vector is learned with a feed-forward neural network. The method is inspired by Commnet.

NaaA: The proposed method. The main player learns DQN with two perspectives: the player's own and the cameraman's. The transmission of reward and communication are performed using the proposed method.

5.3 Results

Training is performed in 10 million steps. Figure 3 Left presents that our model NaaA outperforms two methods. Improvement is achieved by Adaptive DropConnect. We confirmed that the cameraman sees the enemy through an episode. This can be interpreted as the cameraman reporting the enemy position. In addition to seeing the enemy, the cameraman sees the area behind of main player several times. This action enables the cameraman to observe attacks from the enemy while seizing a better relative position.

For further interpretation of the result, we present visualization of the revenue that the agent earned in Figure 3 Right as a heatmap. The background picture is a screen in Doom taken at the moment when the filter in CNN is mostly activated. Figure 4 shows an example of learnt sequence of actions by our method.

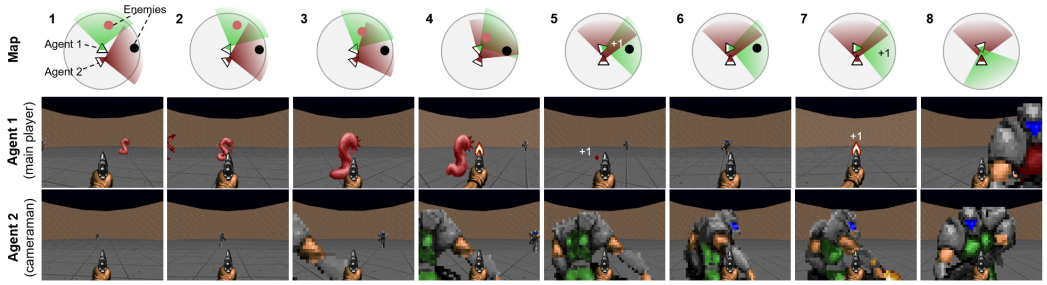

Figure 4: NaaA leads the agents to obtain cooperative relationship. First, the two agents are facing in different directions, and the cameraman sells its information to the main player (**1**). The main player who bought the information starts to turn right to find the enemy. The cameraman who sold the information starts to turn left to seek new information by finding the blind area of the main player (**2** and **3**). With turning, the main player attacks the first enemy which he already saw (**4** and **5**). After the main player finds out the enemy, he attacks the enemy, and obtain the reward (**6** and **7**). Until the next enemy appears, the agents watch their dead area each other (**8**).

6 Further Application

Actually, NaaA is useful not only for multi-agent RL, but also for training of the network. Typical training algorithms of a neural network such as those of RMSProp (Tieleman & Hinton, 2012) and Adam (Kingma & Ba, 2014) are based on a sequential algorithm such as stochastic gradient descent (SGD). Therefore, the problem can be interpreted as a problem to update the state (i.e., weight) to the goal, which is minimization of the expected likelihood.

6.1 Adaptive DropConnect

The learning can be accelerated by application of NaaA to the optimizer. We designate the application of NaaA to SGD as *Adaptive DropConnect* (ADC), which is eventually a combination of DropConnect (Wan et al., 2013) and Adaptive DropOut (Ba & Frey, 2013). We introduce ADC herein as one application of NaaA.

ADC uses NaaA for supervised optimization problem with several revisions. First, an environment has an input state such as an image. The agent is expected to update its parameters to maximize its reward obtained from the criterion calculator. The criterion calculator gives batch-likelihood as the reward to the agent. The agent is a classifier which updates its weights to maximize the reward from the criterion calculator. The weights are recorded as an internal state. As a counterfactual return o_{ijt}, we used a heuristic that uses the absolute value of weight $|w_{ijt}|$, which is the same technique as that used by Adaptive DropOut. We use the absolute value of weights because it is the update amount for which the magnitude of error of the output of units is proportional to $|w_{ijt}|$.

The algorithm is presented as Algorithm 2. Because the algorithm is quite simple, its implementation can be performed easily. For that reason, it can be widely applied for most general deep learning problems such as image recognition, sound recognition, and even for deep reinforcement learning.

6.2 Experiment

6.2.1 Setup

In this experiment, we confirm two tasks of classification and single-agent reinforcement learning.

For classification task, we used three types of datasets, MNIST, CIFAR-10 and STL-10. The task given here is to predict the label for each image. The number of class is 10 in those three datasets. The first dataset, MNIST, is a collection of black and white images of handwritten digits whose size is 28x28. The training set and test set are composed of 60,000 examples and 10,000 examples respectively. The images in CIFAR-10 dataset are colored and the size of each image is 32x32. The task is to predict what is shown in each picture. This dataset contains 6,000 images per class (5,000

Algorithm 2 Adaptive DropConnect

for $t = 1$ **to** T **do**
  Compute a bidding price for every edge: **for** $(v_j, v_i) \in \mathcal{E}$ **do** $b_{ijt} \leftarrow |w_{ijt}|$
  Compute an asking price for every node: **for** $v_i \in \mathcal{V}$ **do** $\hat{q}_{it} \leftarrow \underset{q \in [0,\infty)}{\operatorname{argmax}}\, q d_{it}(q)$.
  for $(v_i, v_j) \in \mathcal{E}$ **do**
    Compute allocation: $g_{jit} \leftarrow H(b_{jit} - \hat{q}_{it})$
  end for
  Sample a switching matrix U_t from a Bernoulli distribution: $U_t \sim \text{Bernoulli}(\varepsilon)$
  Sample the random mask M_t from a Bernoulli distribution: $M_t \sim \text{Bernoulli}(1/2)$
  Generate the adaptive mask: $M'_t \leftarrow U_t \circ M_t + (1 - U_t) \circ G_{ijt}$
  Compute $\mathbf{h}_t$ for making a shipment: $\mathbf{h}_t \leftarrow (M'_t \circ W_t)\mathbf{x}_t + \mathbf{b}_t$
  Update W_t and $\mathbf{b}_t$ by backpropagation.
end for

for training and 1,000 for test). STL-10 is a dataset for image recognition, the number of which is 1,300 for each class (500 for training and 800 for test). The size of each image is 96x96. In this experiment, however, images were resized into 48x48, since the resolution is large compared to the datasets shown above and this dataset requires far more time and resource to compute.

Next, we set the single-agent reinforcement learning task. We used the CartPole task from OpenAI gym with visual input. In this setting, the agent must balance a pole while moving a cart. There is much non-useful information related to the image. For that reason, pruning the pixels is important.

6.2.2 MODEL

In this experiment, we compared two models, DropConnect and Adaptive DropConnect (proposed model in this paper). The baseline model is composed of two convolutional layers and two fully connected layers whose outputs are dropped out (we set the possibility as 0.5). The labels of input data are predicted using log-softmaxed value of last fully connected layer. In DropConnect model and Adaptive DropConnect model, first fully connected layer is replaced by DropConnected layer and Adaptive DropConnected layer respectively. Note that DropConnect model corresponds to the our method with ε = 1.0 and this means agents do not perform their auctions, and randomly mask the weights.

6.2.3 RESULTS

For the MNIST datasets, the models are trained for 10 epochs and then evaluated with the test data. The numbers of epochs for CIFAR-10 and STL-10 are 20 and 40 respectively. Experiments are repeated 20 times for each condition, and the average and standard deviation of error rate was calculated. The results is shown in Table 1. As expected, with the model using Adaptive DropConnect, the classification error rate was lower than both the baseline and DropConnect regardless of the datasets given in this experiment.

Table 1: Experimental result for image classification tasks and single-agent RL

.

	MNIST	CIFAR-10	STL-10	CartPole
DropConnect (Wan et al., 2013)	1.72 ± 0.160	43.14 ± 1.335	50.92 ± 1.322	285 ± 21.5
Adaptive DropConnect	$\mathbf{1.36} \pm 0.132$	$\mathbf{39.84} \pm 1.035$	$\mathbf{42.17} \pm 2.329$	$\mathbf{347} \pm 29.4$

7 CONCLUDING REMARKS AND FUTURE WORK

This paper proposed a NaaA model to address communication in MARL without a TTP based on two key ideas: inter-agent reward distribution and auction theory. Existing MARL communication methods have assumed the existence of a TTP, and hence could not be applied in peer–to–peer environments. The inter-agent reward distribution, making agents redistribute the rewards they received

from the internal/external environment, was reviewed first. When an envy-free auction was introduced using auction theory, it was shown that agents would evaluate the counterfactual returns of other agents. The experimental results demonstrated that NaaA outperformed a baseline method and a CommNet-based method.

Furthermore, a Q-learning based algorithm, termed Adaptive DropConnect, was proposed to dynamically optimize neural network topology with counterfactual return evaluation as a further application. To evaluate this application, experiments were performed based on a single-agent platform, demonstrating that the proposed method produced improved experimental results relative to existing methods.

Future research may also be directed toward considering the connection between NaaA and neuroscience or neuroevolution. Edeleman propounded the concept of neural Darwinism (Edelman, 1987), in which group selection occurs in the brain. Inter-agent rewards, which were assumed in this paper, correspond to NTFs and could be used as a fitness function in genetic algorithms for neuroevolution such as hyperparameter tuning.

As NaaA can be applied in peer-to-peer environments, the implementation of NaaA in blockchain (Swan, 2015) is under consideration. This implementation would extend the areas where deep reinforcement learning could be applied. Bitcoin (Nakamoto, 2008) could be used for inter-agent reward distribution, and the auction mechanism could be implemented by smart contracts (Buterin et al., 2014). Using the NaaA incentive design, it is hoped that the world may be united, allowing people to share their own representations on a global scale.

REFERENCES

A. K. Agogino and K. Tumer. QUICR-learning for multi-agent coordination. AAAI'06, 2006.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 3084–3092, 2013.

Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

G. M Edelman. *Neural Darwinism: The theory of neuronal group selection.* Basic books, 1987.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. *arXiv:1705.08926*, 2017.

Jakob Foerster, Yannis Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016.

Andrew V Goldberg and Jason D Hartline. Competitiveness via consensus. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 215–222. Society for Industrial and Applied Mathematics, 2003.

Andrew V Goldberg, Jason D Hartline, Anna R Karlin, Michael Saks, and Andrew Wright. Competitive auctions. *Games and Economic Behavior*, 55(2):242–269, 2006.

Venkatesan Guruswami, Jason D Hartline, Anna R Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *ACM-SIAM symposium on Discrete algorithms*, 2005.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pp. 1–8. IEEE, 2016.

Wolfgang Ketter, John Collins, and Prashant Reddy. Power tac: A competitive economic simulation of the smart grid. *Energy Economics*, 39:262–270, 2013.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Rodrigue Talla Kuate, Minghua He, Maria Chli, and Hai H Wang. An intelligent broker agent for energy trading: An mdp approach. In *IJCAI*, pp. 234–240, 2013.

William Forster Lloyd. *Two lectures on the checks to population*. 1833.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Roger B Myerson. Mechanism design by an informed principal. *Econometrica: Journal of the Econometric Society*, pp. 1767–1797, 1983.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

David Pardoe and Peter Stone. An autonomous agent for supply chain management. *Handbooks in Information Systems Series: Business Computing*, 3:141–172, 2009.

Tuomas Sandholm and XiaoFeng Wang. (im) possibility of safe exchange mechanism design. In *Eighteenth national conference on Artificial intelligence*, pp. 338–344. American Association for Artificial Intelligence, 2002.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Peter Stone, Robert E Schapire, Michael L Littman, János A Csirik, and David McAllester. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. *Journal of Artificial Intelligence Research*, 19:209–242, 2003.

S. Sukhbaatar, R. Fergus, et al. Learning multiagent communication with backpropagation. In *NIPS'16*, 2016.

Melanie Swan. *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3): 58–68, 1995.

Gerald Tesauro and Jonathan L Bredin. Strategic sequential bidding in auctions using dynamic programming. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pp. 591–598. ACM, 2002.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Daniel Urieli and Peter Stone. Tactex'13: a champion adaptive power trading agent. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 1447–1448. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pp. 1058–1066, 2013.

Michael P. Wellman, Peter R. Wurman, Kevin O'Malley, Roshan Bangera, Daniel Reeves, William E Walsh, et al. Designing the market game for a trading agent competition. *IEEE Internet Computing*, 5(2):43–51, 2001.

Shih-Hung Wu and Von-Wun Soo. Game theoretic reasoning in multi-agent coordination by negotiation with a trusted third party. In *Proceedings of the third annual conference on Autonomous Agents*, pp. 56–61. ACM, 1999.

APPENDIX

A.1 PROOF OF THEOREM 3.1

The optimization problem in Eq:10 is made of two terms except of the constant, and the only second term is depends on $\mathbf{b}$. Hence, we consider to optimize the second term. The optimal bidding prices $\hat{\mathbf{q}}_t$ is given by the following equation.

$$\begin{aligned}\hat{\mathbf{b}}_{it} &= \underset{\mathbf{b}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{q}_t}\left[\mathbf{g}_{it}(\mathbf{b})^{\mathrm{T}}(\mathbf{q}_t - \gamma \mathbf{o}_{it})\right] = \underset{\mathbf{b}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{q}_t}\left[H(\mathbf{b}-\mathbf{q}_t)^{\mathrm{T}}(\mathbf{q}_t - \gamma \mathbf{o}_{it})\right] \\ &= \underset{\mathbf{b}}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{q}_t}\left[\sum_{j=1}^{N} H(b_j - q_{jt})(q_{jt} - \gamma o_{ijt})\right] = \underset{\mathbf{b}}{\operatorname{argmin}} \sum_{j=1}^{N} \mathbb{E}_{q_{jt}}\left[H(b_j - q_{jt})(q_{jt} - \gamma o_{ijt})\right],\end{aligned} \tag{11}$$

From independence, the equation is solved if we solve the following problem.

$$\hat{b}_{ijt} = \underset{b}{\operatorname{argmin}}\, \mathbb{E}_{q_{jt}}\left[H(b - q_{jt})(q_{jt} - \gamma o_{ijt})\right], \;\; \forall j \in \{1, \ldots, N\} \tag{12}$$

Hence, $\hat{b}_{ijt}$ can be derived as the solution which satisfies the following equation.

$$\frac{\partial}{\partial b}\mathbb{E}_{q_{jt}}\left[H(b - q_{jt})(q_{jt} - \gamma o_{ijt})\right]\Big|_{b=\hat{b}_{ijt}} = 0, \;\; \frac{\partial^2}{\partial b^2}\mathbb{E}_{q_{jt}}\left[H(b - q_{jt})(q_{jt} - \gamma o_{ijt})\right]\Big|_{b=\hat{b}_{ijt}} > 0$$

For simplicity, we let $q = q_{jt}$ and $o = o_{ij,t+1}$. Then, the following equation holds.

$$\begin{aligned}\frac{\partial}{\partial b}\mathbb{E}_q\left[H(b-q)(q-\gamma o)\right] &= \frac{\partial}{\partial b}\int_0^{\infty} H(b-q)(q-\gamma o)p(q)dq \\ &= \frac{\partial}{\partial b}\int_0^{b} (q-\gamma o)p(q)dq \\ &= (b-\gamma o)p(q=b),\end{aligned} \tag{13}$$

$$\frac{\partial^2}{\partial b^2}\mathbb{E}_q\left[H(b-q)(q-\gamma o)\right] = p(q=b) + (b-\gamma o)\frac{\partial}{\partial b}p(q=b) \tag{14}$$

Then, condition $\frac{\partial}{\partial b}\mathbb{E}_{q_{jt}}\left[H(b - q_{jt})(q_{jt} - \gamma o_{ijt})\right]|_{b=\hat{b}_{ijt}} = 0$ is satisfied only by $\hat{b}_{ijt} = \gamma o_{ijt}$. We substitue $\hat{b}_{ijt}$ into Eq:14,

$$\frac{\partial^2}{\partial b^2}\mathbb{E}_{q_{jt}}\left[H(b - q_{jt})(q_{jt} - \gamma o_{ijt})\right]\Big|_{b=\hat{b}_{ijt}} = p(q=\hat{b}_{ijt}) + 0 > 0 \tag{15}$$

Therefore, $\hat{b}_{ijt} = \gamma o_{ijt}$ is a unique solution as the minimum point. From generality, $\hat{\mathbf{b}}_{it} = \gamma \mathbf{o}_{it}$ holds.