Hindawi
Journal of Sensors
Volume 2021, Article ID 9651117, 13 pages
https://doi.org/10.1155/2021/9651117

Research Article

Trading and Pricing Sensor Data in Competing Edge Servers with Double Auction Markets

Bing Shi,[1,2] Zhaoxiang Song,[1] and Jianqiao Xu[3]

[1]*School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China*
[2]*Shenzhen Research Institute of Wuhan University of Technology, Shenzhen 518000, China*
[3]*Department of Information Security, Naval University of Engineering, Wuhan 430033, China*

Correspondence should be addressed to Jianqiao Xu; xujianqiao321@163.com

Received 6 November 2021; Revised 10 December 2021; Accepted 13 December 2021; Published 31 December 2021

Academic Editor: Guolong Shi

With the development of the IoT (Internet of Things), sensors networks can bring a large amount of valuable data. In addition to be utilized in the local IoT applications, the data can also be traded in the connected edge servers. As an efficient resource allocation mechanism, the double auction has been widely used in the stock and futures markets and can be also applied in the data resource allocation in sensor networks. Currently, there usually exist multiple edge servers running double auctions competing with each other to attract data users (buyers) and producers (sellers). Therefore, the double auction market run on each edge server needs efficient mechanism to improve the allocation efficiency. Specifically, the pricing strategy of the double auction plays an important role on affecting traders' profit, and thus, will affect the traders' market choices and bidding strategies, which in turn affect the competition result of double auction markets. In addition, the traders' trading strategies will also affect the market's pricing strategy. Therefore, we need to analyze the double auction markets' pricing strategy and traders' trading strategies. Specifically, we use a deep reinforcement learning algorithm combined with mean field theory to solve this problem with a huge state and action space. For trading strategies, we use the Independent Parametrized Deep Q-Network (I-PDQN) algorithm combined with mean field theory to compute the Nash equilibrium strategies. We then compare it with the fictitious play (FP) algorithm. The experimental results show that the computation speed of I-PDQN algorithm is significantly faster than that of FP algorithm. For pricing strategies, the double auction markets will dynamically adjust the pricing strategy according to traders' trading strategies. This is a sequential decision-making process involving multiple agents. Therefore, we model it as a Markov game. We adopt Multiagent Deep Deterministic Policy Gradient (MADDPG) algorithm to analyze the Nash equilibrium pricing strategies. The experimental results show that the MADDPG algorithm solves the problem faster than the FP algorithm.

1. Introduction

With the development of the IoT (Internet of Things), smart terminals embedded with a large number of sensors such as cameras, GPS, and gyroscopes are becoming more and more common in daily life [1], where massive amounts of data are collected [2]. In addition to being utilized by the local smart IoT applications, these valuable data can be traded in the connected edge servers, which can on one hand provide the computing resource for the smart phone applications, and on the other hand, provide a market mechanism for trading the data between the data users (referred as buyers) and data generators (referred as sellers) [3]. For example, traffic information can be collected from the smartphone to edge server, which can be sold to some navigation applications for optimizing the route planning. In this scenario, double auction, as an auction mechanism in which there are multiple buyers and sellers (referred as traders in the following) in the market can be used for trading data between data users (buyers) and data generators (sellers) by the edge server. In this mechanism, buyers and sellers can bid at any time during the trading, and the market will match buyers with sellers who have submitted the bids at a specified time. This mechanism allows traders to enter the market at any

time and trade multiple commodities at the same time. Due to its high allocation efficiency, double auctions have been widely used to solve real-world resource allocation, such as in the stock market [4], the emission trading market [5], the spectrum auction market [6], cloud computing resource allocation market [7], and the sensors networks resource allocation market [3]. In such markets, both traders and the double auction market need to adopt efficient trading strategies and market mechanism [8].

In the real world, there may exist a large number of traders on the edge server to trade data. Furthermore, there may exist multiple edge servers running double auction markets. At this moment, traders need to decide which double auction to participate in and how to bid in the selected double auction market, while the double auction market needs efficient mechanism to improve the allocation efficiency to attract more traders. Since the pricing strategies can determine the price at which traders will trade, it will affect the traders' profit significantly. At the same time, the trading strategies (i.e., how to choose the market and bidding) will in turn affect the market's pricing strategies, thereby affecting the market allocation efficiency. Therefore, we need to analyze the trading strategies of traders and pricing strategies of double auctions in the environment with multiple competing edge servers running double auction markets.

In double auctions, both the market and traders are self-interested agents, and their strategies are affected by each other. Game theory is widely used to analyze the strategic interactions of self-interested agents, in which Nash equilibrium is an important solution concept. Therefore, we will analyze the Nash equilibrium trading strategies and pricing strategies in this competing environment. Specifically, this problem involves a large number of traders, which may have continuous bidding space and private preference. Although the generalized FP algorithm can solve similar problems, it will be difficult to solve the Nash equilibrium in a feasible time when there are a large number of traders. In this paper, we will analyze the Nash equilibrium trading strategies and pricing strategies based on deep reinforcement learning and mean field theory. For the Nash equilibrium trading strategies, we combine the Independent Parametrized Deep Q-Network (I-PDQN) algorithm, which is suitable for solving the problem with hybrid actions, with the mean field theory [9, 10] to solve the Nash equilibrium trading strategies. The experimental results show that this algorithm can solve the problem significantly faster than the FP algorithm. For the Nash equilibrium pricing strategies, we adopt the Multiagent Deep Deterministic Policy Gradient (MADDPG) algorithm. We also find that the Nash equilibrium pricing strategy obtained by this algorithm is the same as the solution of FP algorithm, and the MADDPG algorithm can solve the problem faster than the FP algorithm. The experimental results can also provide useful insights for designing the practical trading strategies and pricing strategies in the real world.

The rest of this paper is structured as follows. In Section 2, we introduce the related work. In Section 3, we introduce the basic settings of the double auction market. In Section 4, we analyze the Nash equilibrium trading strategy based on the I-PDQN algorithm and mean field theory. In Section 5, we use MADDPG algorithm to solve the Nash equilibrium pricing strategy. Finally, we conclude the paper in Section 6.

2. Related Work

There exist a number of works about data acquisition [11–14]. Specifically, Sangoleye et al. studied the data acquisition problem from the SIoT nodes following a techno-economics-based approach via exploiting Contract Theory. Chung et al. proposed a test-bed that consists of on-body sensors and an Android mobile device to acquire the human activity data and then used LSTM network to recognize human behavior. Ho et al. proposed a frame that uses unmanned aerial vehicles (UAVs) to collect data and used Particle Swarm Optimization (PSO) method to find the optimal topology in order to reduce the energy consumption, Bit Error Rate (BER), and UAV travel time. Maksymova et al. studied liDAR sensor data acquisition and compression for automotive applications.

There also exist a lot of works about data trading, such as [15–17]. Specifically, Tian et al. proposed a market mechanism considering the privacy leakage for trading IoT data in one-to-many trading scenarios [18]. They further proposed a many-to-many data trading strategy, which redefines some unreasonable assumptions of the existing mechanisms [19]. Yu et al. proposed a market model to trade mobile data between mobile users by taking into account data demands and demand uncertainty [20]. Hui et al. proposed a sensing service system by considering the utilities of data providers and data service providers with a data pricing strategy in the vehicle sensor networks [21]. Niyato et al. proposed a data market model for IoT data [22]. Al-Fagih et al. proposed a data pricing model for public sensing data by considering delay, quality of services, and trust factors [23]. Furthermore, double auction, as a highly efficient resource allocation mechanism, has been widely used in the data trading market. For example, Jiao et al. designed a double auction-based data market model and pricing mechanism to maximize the profit [24]. Chen et al. used double auction to trade sensor data [25]. Sun et al. used edge servers as a double auction market to solve the problem of insufficient computing resources [26]. Cai et al. proposed a truthful double auction mechanism for data trading market against three major challenges, including diverse market preferences, the complex conflicts of interest relations of data consumers, and the strategic behaviors of both sides [27].

Trading strategies and pricing strategies play an important role in the double auction market, and therefore, there exist a number of works about trading strategies and pricing strategies in double auctions. For trading strategies, Gode et al. proposed the "Zero Intelligence" (ZI) trading strategy for the first time [28]. Traders can only randomly select bids, and all bids are uniformly distributed. Brown and Von proposed the fictitious play algorithm (FP algorithm for short) [29], in which each trader estimates the FP beliefs of other traders through historical bids and calculates the current best response strategy based on this. But the original

algorithm can only solve the complete information game, so Rabinovich et al. proposed a generalized FP algorithm to analyze the continuous type of incomplete information game [30], but this generalized algorithm is only suitable for unilateral auctions. Shi et al. made improvements on this basis to analyze the incomplete information game problem under double auctions [31]. For the first time, Schvartzman and Wellman combined empirical game theory with the Q-learning algorithm in reinforcement learning to analyze the optimal trading strategy of traders in the double auction market [32], but this algorithm is only suitable for a small and discrete space of bidding actions. Chowdhury et al. proposed a trading strategy using Monte Carlo Tree Search (MCTS) [33]. However, this algorithm is suitable for discrete bidding sets and cannot deal with bidding problems with continuous types and action spaces. Bredin and Parkes designed a framework of truthful bidding in double auction market [34].

Furthermore, there also exist some works analyzing the market pricing strategies in the competing environment with multiple double auction markets. Miller and Niu experimentally analyzed traders' market selection strategies in the competing marketplaces trading environment [35]. Cai et al. analyzed the impact of different adaptive strategies on the trading strategy and its own earnings in the market competition environment [36]. Shi et al. considered two different pricing strategies and analyzed how to adjust their pricing strategies to attract traders in two competing markets [37], and then they considered four typical types of fees in pricing strategies, to analyze the Nash equilibrium market selection in competing environment [38].

From the above work about data trading strategies, we can find that there exist few works on trading with continuous types and action spaces under incomplete information, and most of the above works only consider a small number of traders when analyzing the Nash equilibrium solution. Regarding the market pricing strategies, although there exist some works considering the competing environment, these works have not considered how the market should adjust the pricing strategy under the incomplete information game of a large number of traders. In this paper, we will analyze the Nash equilibrium trading strategies of sensor data and market pricing strategies of the double auction market running on the edge server in a competing environment with a large number of traders.

3. Basic Settings

In this section, we will introduce the basic settings of traders and the double auction market running on the edge server. We will describe the basic settings of traders and introduce how to compute the expected profits of traders. We then introduce the pricing strategy of double auctions and describe how to compute the allocation efficiency of the double auction market.

3.1. Basic Setting of the Trader. In this paper, the traders consist of data buyers and data sellers. The set of buyers is denoted as $\mathscr{B} = \{1, 2, \cdots, \mathbb{B}\}$, and the set of sellers is denoted as $\mathscr{S} = \{1, 2, \cdots, \mathbb{S}\}$. The set of all markets is denoted as $\mathscr{M} = \{1, 2, \cdots, \mathbb{M}\}$. Each trader has a type, and the type of the seller is the lowest price it is willing to sell. The buyer's type indicates the highest price at which the buyer is willing to buy an item, and the seller's type is the lowest price at which the seller is willing to sell the item. The type actually indicates the trader's preference on the item. The types of a buyer and a seller are denoted as $\theta^{\tilde{b}} \in [0, 1]$ and $\theta^{\tilde{s}} \in [0, 1]$, respectively, which is private information, that is, the type of each specific buyer or seller is unknown to others. However, the types of all buyers and sellers are assumed to be common knowledge, and *i.i.d* drawn from the cumulative distribution functions $F^{\tilde{b}} \in [0, 1]$ and $F^{\tilde{s}} \in [0, 1]$, respectively, there are assumed to be differentiable, and the probability density functions are $f^{\tilde{b}}$ and $f^{\tilde{s}}$, respectively. We assume that a small cost τ will occur when the trader enters the market (for example, the time it takes for online trading). Therefore, when the buyer's type is too low or the seller's type is too high, they choose not to enter the market. In doing so, the behavior of buyers bidding low offers and not entering the market, and the behavior of sellers bidding high offers and not entering the market can be distinguished. Next, we describe how traders choose a market and bid in the market. We define the action of a buyer as a tuple $\delta^{\tilde{b}} = m, d_m^{\tilde{b}}$ and thus the buyer's market choice and the bid in the selected market are treated as trading strategy. Note that $m \neq 0$ means that the buyer bids $d_m^{\tilde{b}}$ in market m, and $m = 0$ means that the buyer does not enter any market. Similarly, we use $\delta^{\tilde{s}} = m, d_m^{\tilde{s}}$ to represent the seller's action.

3.1.1. Trader's Expected Utility. In this section, we introduce how to compute the expected utilities of traders. In what follows, we introduce how a seller's expected utility is calculated. Similarly, the buyer's expected utility can also be derived in the same way. The expected utility of a seller is determined by its type $\theta^{\tilde{s}}$, its action $\delta^{\tilde{s}} = m, d_m^{\tilde{s}}$, and its belief about the actions of other buyers and sellers $\Omega_m^{\tilde{b}}$, $\Omega_m^{\tilde{b}}$, We define $\Omega_m^{\tilde{s}}$ as a tuple $d_{1,m}^{\tilde{s}}, d_{2,m}^{\tilde{s}}, \cdots, d_{\eta,m}^{\tilde{s}}$, where $d_{i,m}^{\tilde{s}}$ represents the ith smallest seller's bidding action in market m. In particular, the number of sellers taking each different action is denoted as a tuple $\bar{x} = x_1, x_2, \cdots, x_\eta$ where x_i represents the number of sellers choosing action $d_{i,m}^{\tilde{s}}$. Now, the seller's position is determined as follows. We use $X_m^{<}(\Omega_m^{\tilde{s}}, d_{i,m}^{\tilde{s}})$ to represent the number of other sellers who have a lower bidding than $d_{i,m}^{\tilde{s}}$ in market m, and it can be calculated as

$$X_m^{<}\left(\Omega_m^{\tilde{s}}, d_{i,m}^{\tilde{s}}\right) = \sum_{\phi_l \Omega_m^{\tilde{s}} : \phi_l < \phi_l \Omega_m^{\tilde{s}}} x_l. \tag{1}$$

Similarly, excluding the seller itself, we use $X_m^{=}(\Omega_m^{\tilde{s}}, d_{i,m}^{\tilde{s}})$ to represent the number of sellers who have the same bid as

the seller, and it can be calculated as

$$X_m^{=}\left(\Omega_m^{\tilde{s}}, d_{i,m}^{\tilde{s}}\right) = \sum_{\phi_l \in \Omega_m^{\tilde{s}}: \phi_l = d_{i,m}^{\tilde{s}}} x_l. \tag{2}$$

Now, any position from $X_m^{<}(\Omega_m^{\tilde{s}}, d_{i,m}^{\tilde{s}}) + 1$ to $X_m^{<}(\Omega_m^{\tilde{s}}, d_{i,m}^{\tilde{s}}) + X_m^{=}(\Omega_m^{\tilde{s}}, d_{i,m}^{\tilde{s}}) + 1$ can be seller's bidding action $d_{i,m}^{\tilde{s}}$, and this position is denoted as $v_m \in \mathbb{V}_m$, where $\mathbb{V}_m$ is the set of all possible positions. So the probability of any v_m in the set $\mathbb{V}_m$ is

$$P(v_m) = \frac{1}{X_m^{=}\left(\Omega_m^{\tilde{s}}, d_{i,m}^{\tilde{s}}\right) + 1}. \tag{3}$$

Now, the seller's expected value can be calculated as

$$\tilde{V}_m\left(v_m, \Omega_m^{\tilde{b}}, \theta^{\tilde{s}}, d_{i,m}^{\tilde{s}}\right) = \varphi\left(v_m, \Omega_m^{\tilde{b}}, d_{i,m}^{\tilde{s}}\right) \times \theta^{\tilde{s}}, \tag{4}$$

where $\varphi(v_m, \Omega_m^{\tilde{b}}, d_{i,m}^{\tilde{s}})$ represents whether the seller can trade in the market.

$$\varphi\left(v_m, \Omega_m^{\tilde{b}}, d_{i,m}^{\tilde{s}}\right) = \begin{cases} 1 & \text{if } Y_m^{\geq}\left(\Omega_m^{\tilde{b}}, d_{i,m}^{\tilde{s}}\right) \geq v_m, \\ 0 & \text{if } Y_m^{\geq}\left(\Omega_m^{\tilde{b}}, d_{i,m}^{\tilde{s}}\right) < v_m, \end{cases} \tag{5}$$

where Y_m represents the total number of buyers with a bid of $d_{i,m}^{\tilde{s}}$ or less in the market.

$$Y_m^{\geq}\left(\Omega_m^{\tilde{b}}, d_{i,m}^{\tilde{s}}\right) = \sum_{\phi_l \in \Omega_m^{\tilde{b}}: \phi_l \geq d_{i,m}^{\tilde{s}}} y_l. \tag{6}$$

Considering all v_m in the market m, the expected value of the seller is

$$\tilde{V}_m^{\tilde{s}}\left(\theta^{\tilde{s}}, d_{i,m}^{\tilde{s}}, \Omega_m^{\tilde{b}}, \Omega_m^{\tilde{s}}\right) = \sum_{v_i \in V_i} P(v_i) \times \tilde{V}_m\left(v_m, \Omega_m^{\tilde{b}}, \theta^{\tilde{s}}, d_{i,m}^{\tilde{s}}\right). \tag{7}$$

The derivation process of the buyer's expected value is the same.

Then, we derive the equation of calculating the expected payment of the seller, and expected utility can be calculated as expected payment minus expected value. We can determine the equilibrium price range $[d_l, d_h]$ and the price is $p_k = d_l + k \times (d_h - d_l)$ according to the equilibrium k pricing strategy. Then, the seller's expected payment for bidding $d_{i,m}^{\tilde{s}}$ is

$$\tilde{\mathscr{P}}_m\left(v_m, d_{i,m}^{\tilde{s}}, \Omega_m^{\tilde{s}}, \Omega_m^{\tilde{b}}, k_m\right) = \begin{cases} p_k + \tau & \text{if } \varphi\left(v_m, \Omega_m^{\tilde{b}}, d_{i,m}^{\tilde{s}}\right) = 1, \\ \tau & \text{if } \varphi\left(v_m, \Omega_m^{\tilde{b}}, d_{i,m}^{\tilde{s}}\right) = 0 \end{cases} \tag{8}$$

Now the seller's expected utility for bidding $d_{i,m}^{\tilde{s}}$ is

$$\tilde{v}_{i,m}^{\tilde{s}}\left(\theta^{\tilde{s}}, \Omega_m^{\tilde{s}}, \Omega_m^{\tilde{b}}, \delta^{\tilde{s}}\right)|k_m = \tilde{V}_m^{\tilde{s}}\left(\theta^{\tilde{s}}, d_{i,m}^{\tilde{s}}, \Omega_m^{\tilde{b}}, \Omega_m^{\tilde{s}}\right) - \tilde{\mathscr{P}}_m\left(v_m, d_{i,m}^{\tilde{s}}, \Omega_m^{\tilde{s}}, \Omega_m^{\tilde{b}},\right. \tag{9}$$

The derivation process of the buyer's expected utility is similar.

Now at market auction stage t, assuming that the seller's trading strategy is $\delta^{\tilde{b}} = (g, d_g^{\tilde{b}})$, the seller's immediate reward is

$$r_t^{\tilde{s}}|\delta^{\mathscr{M}} = \begin{cases} \tilde{v}_{i,g}^{\tilde{s}}\left(\theta^{\tilde{s}}, \Omega_g^{\tilde{s}}, \Omega_g^{\tilde{b}}, \delta^{\tilde{s}}\right)|k_g & \text{if} \quad g \neq 0, \\ 0 & \text{if} \quad g = 0. \end{cases} \tag{10}$$

The accumulative reward of the seller is

$$R_t^{\tilde{s}} = \sum_{z=t}^{\infty} \gamma^{z-t} r_t^{\tilde{s}}, \tag{11}$$

where γ is the discount factor in reinforcement learning, indicating the degree of importance of future rewards. The derivation of the buyer is the same.

3.2. Market Setting. We now introduce the basic settings about the pricing strategy of double auctions.

3.2.1. Equilibrium K Pricing Strategy. In this paper, it is assumed that all markets adopt equilibrium k pricing strategy, in which the pricing parameter of the market is $k \in [0, 1]$. Therefore, it is stipulated that the pricing parameter of market m is k_m and the competitive pricing strategies of $\mathbb{M}$ markets are $\delta^{\mathscr{M}} = k_1, k_2, \cdots, k_M$.

In equilibrium k pricing, the equilibrium price range is E_p. After equilibrium matching, traders who successfully match (the matched seller's asking price does not exceed the buyer's bid) can trade at any price within the equilibrium price range. Therefore, the set of buyers and sellers who successfully match and can trade is $\{<\tilde{b}_1, \tilde{s}_1>, <\tilde{b}_2, \tilde{s}_2>, \cdots, <\tilde{b}_{lm}, \tilde{s}_{lm}>\}$, and the set of bidding is $\{<d_1^{\tilde{b}}, d_1^{\tilde{s}}>, \cdots, <d_{lm}^{\tilde{b}}, d_{lm}^{\tilde{s}}>\}$. According to the above conditions, the equilibrium price interval must be a subinterval of interval $(d_{lm+1}^{\tilde{s}}, d_{lm+1}^{\tilde{b}})$, a.k.a. $E_p \subseteq (d_{lm+1}^{\tilde{s}}, d_{lm+1}^{\hat{b}})$, which is recorded as $E_p = [d_l, d_h]$. Under equilibrium k pricing, all traders trade at the same price and are in E_p. The trading price is $p = d_l + (d_h - d_l) \times k$. Obviously, when k is larger, to the market biases to the seller, otherwise, it biases to the buyer.

3.2.2. Allocation Efficiency. We now introduce how to compute the allocation efficiency of the markets. Allocation efficiency is one of the most important metrics of measuring the performance of the double auction. The allocation efficiency is the ratio of the actual profit obtained by all buyers and sellers in the market to the maximum profit theoretically

obtained when they submit their types as the bid, which is

$$\mathrm{AE} = \frac{\sum_{i\in \mathrm{TB}} \left|\theta_i^b - \mathrm{TP}_i\right| + \left|\mathrm{TP}_i - \theta_i^s\right|}{\sum_{i\in \mathrm{TB}^*} \left|\theta_i^b - \mathrm{TP}_i^*\right| + \left|\mathrm{TP}_i^* - \theta_i^S\right|}, \tag{12}$$

where TB is the set of actual transactions made by traders, θ_i^b is the type of the buyer in transaction i, θ_i^s is the type of the seller in transaction i, TP_i is the transaction price of transaction i, TB^* is the set of transactions when traders submit their types as their bids, and TP_i^* is the transaction price of transaction i when traders submit their types as the bids.

3.2.3. Market Reward. In this paper, the competing double auction market intends to maximize the allocation efficiency by adopting an efficient pricing strategy in order to attract traders. Therefore, we take the market allocation efficiency as the market reward.

In each stage t, each market publishes its pricing action. Traders then choose a market to participate and bid according to the trading strategy. When all participated traders have bid, each market matches buyers with sellers according to the equilibrium matching strategies According to equation (12), the immediate reward of the market is expressed as follows:

$$r_{m,t} = \frac{\sum_{i\in \mathrm{TB}} \left|\theta_i^b - \mathrm{TP}_i\right| + \left|\theta_i^s - \mathrm{TP}_i\right|}{\sum_{i\in \mathrm{TB}^*} \left|\theta_i^b - \mathrm{TP}_i^*\right| + \left|\theta_i^S - \mathrm{TP}_i^*\right|}. \tag{13}$$

The accumulative reward of the market is

$$R_{m,t} = \sum_{z=t}^{\infty} \gamma^{z-t} \left(\frac{\sum_{i\in \mathrm{TB}} \left|\theta_i^b - \mathrm{TP}_i\right| + \left|\theta_i^s - \mathrm{TP}_i\right|}{\sum_{i\in \mathrm{TB}^*} \left|\theta_i^b - \mathrm{TP}_i^*\right| + \left|\theta_i^s - \mathrm{TP}_i^*\right|} \right). \tag{14}$$

4. Nash Equilibrium Trading Strategy

When traders choose the edge servers market to participate and bid, their strategies are affected by each other. Therefore, we need to derive the Nash equilibrium trading strategies. In this paper, all traders use reinforcement learning to improve their trading strategies until all traders have converged. At this moment, traders have reached the Nash equilibrium strategy. It should be noted that although the learning process is repeated, the game we study is essentially a one shot game. One shot game means that all participants play only one round of game. In this repeated learning process, agents will choose the action in the current state according to the observed information of previous states and the obtained profit and enter the next state at the same time. This process is a sequential decision-making process. Therefore, we model it as a Markov decision process and use the deep reinforcement learning algorithm to solve the Nash equilibrium strategy. We use I-PDQN (independent parameterized deep q-network) algorithm to analyze traders' Nash equilibrium trading strategy and evaluate it against the FP algorithm [40] in terms of computation speed and convergence result.

We assume that there are two competing edge server double auction markets. When the number of markets is greater than 2, our method is still applicable. In each stage, traders need to select a market and bid. Therefore, the trading strategy consists of two parts, choosing a market, where the action space is discrete, and bidding, where the action space is continuous. Therefore, the whole trading action is a hybrid action with continuous and discrete action. Furthermore, this problem involves a large number of traders. Therefore, we intend to solve the trading strategy problem of a large number of traders with hybrid actions based on I-PDQN algorithm and mean field theory.

4.1. I-PDQN Algorithm. As we have discussed in the above, P-DQN algorithm [41] is applicable to the hybrid action space of a single agent. This algorithm is then extended to the environment with multiple cooperative agents [42]. However, traders in the double auction market are not cooperative, and therefore, we extend it to the environment with multiple noncooperative agents, called I-PDQN algorithm. In the following, we first briefly introduce P-DQN algorithm and then introduce I-PDQN algorithm.

P-DQN algorithm can deal with the problem with hybrid action space. The idea is to update discrete action strategy and continuous action strategy, respectively, in combination with DQN algorithm [43] and DDPG algorithm [44]. In the P-DQN algorithm, first, the low-level parameters related to each high-level discrete action are selected, and then, the discrete-continuous hybrid action pairs that can maximize the action value function are calculated. More specifically, the discrete-continuous hybrid action space $\mathscr{A}$ can be defined as

$$A = \{(e, x_e) \mid x_e \in X_e \quad \text{for all} \quad e \in [E]\}, \tag{15}$$

where $[E] = 0, 1, \cdots, E-1$ is the set of discrete actions, and $\mathscr{X}_e$ is set of all discrete actions $e \in [E]$ corresponding to the continuous actions. Therefore, a deterministic function can be defined to map the state and each discrete action e to the corresponding continuous parameter x_e

$$x_e = \mu_e(s; \theta), \tag{16}$$

where θ is the weight of deterministic policy network. A discrete action value function $Q(s, e, x_e; \omega)$ is further defined to map the state s and all hybrid actions to the actual value. ω is the weight of the discrete action value network. P-DQN updates the discrete action policy network parameters through the following loss function:

$$l^Q(\omega) = \frac{1}{2}\left[Q(s, e, x_e; \omega) - y\right]^2, \tag{17}$$

where the expression of y is

$$y = r + \max_{e\in[E]} \gamma Q\left(s', e, \mu_e\left(s'; \theta\right); \omega\right), \tag{18}$$

where s' is the next state after the hybrid action (e, x_e) is taken. The policy update of the continuous parameter part is through fixing the parameter ω and minimizing the loss function $l^{\Theta}(\theta)$:

$$l^{\Theta}(\theta) = -\sum_{d=1}^{D} \boxtimes Q(s, e, \mu_e(s;\theta);\omega). \tag{19}$$

Therefore, the action value function $Q(s, e, x_e;\omega)$ mainly plays two roles. First, it outputs the greedy strategy for all discrete actions (consistent with DQN), and second, it provides a gradient for the policy update of continuous parameters.

After introducing P-DQN algorithm, we now introduce I-PDQN algorithm for multiple non-cooperative agents. I-PDQN is an algorithm with low space and time consumption. Specifically, I-PDQN has a space complexity of $O(\lambda n)$ at each round. Where n is the size of replay memory, and replay memory is cleared for each round, which means our algorithm does not take up much memory space. Note that it is difficult to get an exact value for the time complexity of deep reinforcement learning. However, in our experiments, we can get the convergence result in a reasonable time. In more detail, the algorithm takes the competing market pricing parameters, the number of buyers and sellers and the bidding space as the input, and finally outputs the Nash equilibrium trading strategy. Because each trader intends to maximize its own profit, it learns the best trading strategy independently. Therefore, the I-PDQN algorithm adopts the autonomous learning paradigm, and each trader has an independent P-DQN algorithm to learn [45, 46]. Because this game involves a large number of traders, we introduce the mean field theory into I-PDQN algorithm, to describe the state of the market. The detail of the algorithm is shown in Algorithm 1.

4.2. Experiment Analysis. The experiments are run on the system with configuration Intel(R) Core(TM) i7-8700 CPU, 12 CPU cores, 7 GB GPU memory, CUDA version 10.2, and Ubuntu 16.04.6 LTS 4.15.0-45 genetic GNU/Linux.

4.2.1. Parameter Setting. We now experimentally analyze the Nash equilibrium trading strategy. In the experimental analysis, we consider 50 buyers and 50 sellers. For the hybrid action of each trader, the discrete action set is expressed as $[E] = \{0, 1, \cdots, E-1\}$, where E is the total number of discrete actions, and the continuous action parameter corresponds to each discrete action $x_e \in [0, 1]$. In the action selection stage, each trader first generates the continuous parameters corresponding to all discrete actions according to the observed states. The exploration probability is set as $\varepsilon = 0.995^t$, and the exploration probability will gradually decrease with the increase of training iterations. For the selection of discrete actions, traders randomly select a discrete action from the uniform distribution of $\zeta \tilde{} \mathrm{U}(0, 3)$ with the probability of ε for exploration. [0,3] uniform distribution is used for random exploration of discrete actions. Traders randomly select a number in [0,3] uniform distribution. If it is between [0,1], it means that the choice of discrete action is 0, they do not enter the market; if (1,2] means that the choice of discrete action is 1, select market 1 to enter; if (2,3] means that the choice of discrete action is 2, select market 2 to enter. There are six states in each stage of the market, and the specific parameters are explained in Table 1. For the replay memory $D_i = 200000$ of each trader i, the sample size $\mathbb{R} = 64$ is selected in batch, the update ratios of θ_i and ω_i are $\alpha = 0.01$ and $\beta = 0.001$, and the discount factor of y is $\gamma = 0.95$.

4.2.2. Experimental Results. We selects two typical pricing strategies $<0, 1>$ and $<0.5, 0.5>$ for analysis. These are the most common strategies in the economic market. The two markets compete with each other. I-PDQN algorithm is used to train the Nash equilibrium trading strategies to get which market traders will enter and how many to bid in the equilibrium state.

Figure 1 shows the changes of traders' market choice in the iterative process with a combined pricing strategy of $<0, 1>$ where the pricing strategy of market 1 is $k = 0$ and the pricing strategy of market 2 is $k = 1$. At this time, market 1 is completely biased towards the buyer, while market 2 is completely biased towards the seller. As can be seen in Figure 1(a), with the passage of training, sellers with types less than 0.5 will gradually enter market 2. This is because market 2 is completely biased towards sellers, and thus, market 2 will attract sellers to participate. However, since sellers with types 0.5 cannot win in the competition, they will choose to go to market 1 in order to successfully trade. The analysis of buyers in Figure 1(b) is the same as the analysis of sellers' market selection strategy. In Figure 1(b), market 1 will eventually attract buyers with type greater than 0.5 and sellers with type greater than 0.5, while market 2 will attract traders with smaller type. This shows that through continuous learning, buyers and sellers will choose a market that is conducive to their own market or can trade successfully.

Figure 2 shows the convergence results of traders' Nash equilibrium trading strategy in the competing environment with a pricing strategy of $<0, 1>$. Note that the training process of Algorithm 1 can only output the equilibrium action of the specific types. Based on these equilibrium actions corresponding to trader types, we further use neural network to fit the final trading strategy, which is a mapping from trader types to actions. The results show that in the equilibrium state, how traders select the market, and bid in the participated market. We also can find that both markets can attract traders, and the markets can coexist. According to the market choice of traders, it can be seen that traders with larger types enter market 1, while traders with smaller types will enter market 2. In market 1, because it is completely biased towards the buyer, buyers are willing to bid close to their types, a.k.a. bidding truthfully, while sellers want to hide their bids more. In market 2, sellers are willing to bid close to their true types because market 2 is completely biased towards sellers. Sellers will try to bid truthfully in order to improve the matching probability. From Figure 2, we also find that when the type of buyer is less than 0.12 and the type of seller is greater than 0.88, they will choose not to

Input: market pricing parameters k_1 and k_2, number of buyers B, number of sellers S, trader bidding space λ
Output: the Nash equilibrium trading strategy of the trader
1 **Initialization**: For each Trader $i \in B + S$, Initialize the Exploration Parameter ε, Batch Size $\mathbb{R}$, Uniform Distribution ξ, and Randomly Initialize the Network Weights $\omega_i i$ and θ_i, and $t = 0$, and the Initial State Is $s_0 i$
2 **while** The loss function of traders is not convergence **do**
3 For each trader i, Calculate the continuous parameter $x_{e^{i,t}} \longleftarrow \mu_i(s_{i,t}; \theta_{i,t})$ corresponding to all discrete actions according to the current state;
4 Select action $a_{i,t} = (e_{i,t}, x_e^{i,t}$ according to the following rules
5 $$a_{i,t} = \begin{cases} \textit{Select an action at random f from } \zeta & \boldsymbol{\varepsilon} \\ (e_{i,t}, x_e^{i,t}) e_{i,t} = \arg \max\limits_{e \in [E]} Q(s_t, e, x_e^{i,t}; \omega_t) & 1 - \varepsilon \end{cases}$$
6 When the bidding time of the current stage ends, each trader obtains its immediate return $r_{i,t}$ and the state $s_{t+1} i$ of the next stage through the market rules;
7 For each trader i, the tuple $s_t^i, a_{i,t}, r_{i,t}, s_{t+1}^i$ is stored in replay memory D_i;
8 **Strategy training**:
9 Each trader i takes $\mathbb{R}$ samples from replay memory D_i and calculates y_i according to equation:
10 $$y_{i,b} = \mathbf{r}_{i,b} + \max_{e \in [E]} \gamma Q(s_{b+1}, e, x_e(s_{b+1}, \theta_{i,t}); \omega_{i,t})$$
11 Calculate the random gradients $\nabla_\omega l_{i,t}^Q(\omega)$ and $\nabla_\theta l_{i,t}^{\boldsymbol{\Theta}}(\theta)$ according to equations (17) and (19), and update the weight according to equation:
12 $\omega_{t+1} \longleftarrow \omega_t - \alpha_t \nabla_\omega l_{i,t}^Q(\omega_{i,t})$ and $\theta_{t+1} \longleftarrow \theta_t - \beta_t \nabla_\theta l_{i,t}^{\boldsymbol{\Theta}}(\theta_{i,t})$

Algorithm 1. I-PDQN algorithm.

Table 1: State parameters of traders.

Parameter	Description
H_1	Normalization of recent average transaction price P
H_2	Best bid for traders with opposite roles
H_3	Best bid for traders with the same roles
H_4	Average bid of traders with the same roles
H_5	Average bid of traders with opposite roles
H_6	Last round transaction price

enter the market because of the fixed cost (such as time cost) of entering the market.

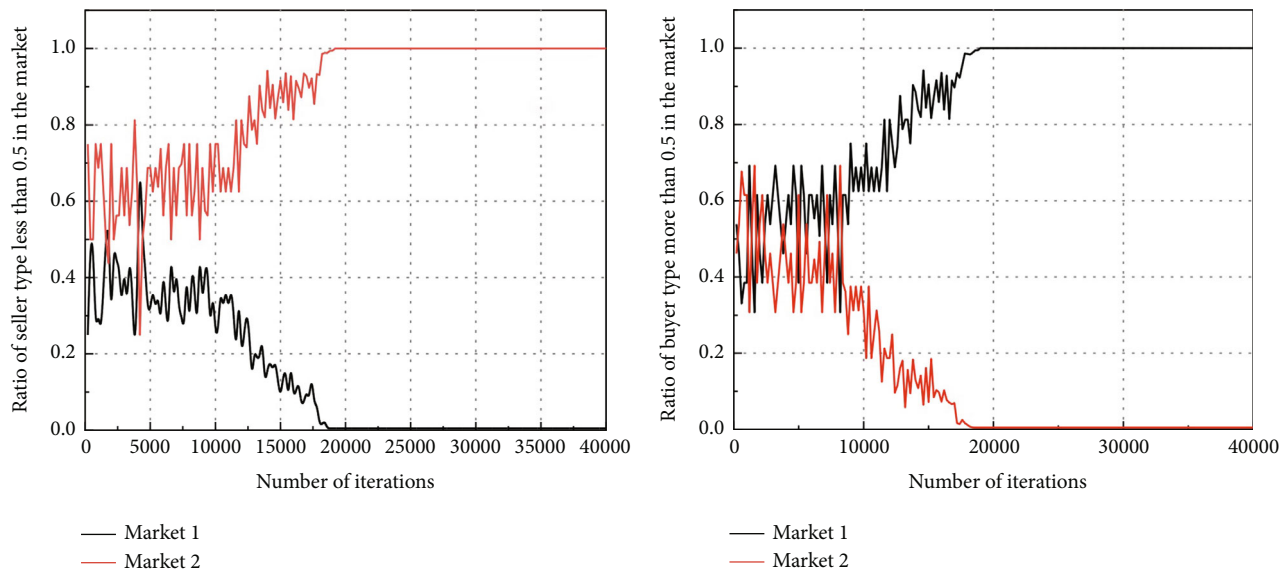
For the competing markets with pricing strategy $< 0.5, 0.5 >$, the results show that under the same pricing strategy, traders will eventually converge to only one market, where which market to be converged is random, and the bidding strategies of all traders are similar to those in a single market. This is because when the two competing markets are the same, traders will choose the market with more participants to enter in order to improve their probability of being matched. This leads to that only one market can survive in the end.

4.3. Experimental Evaluation against FP. Another way to solve the game with continuous private type is to use the generalized FP algorithm. Therefore, we will evaluate our algorithm against the FP algorithm. In this evaluation, we still consider that there are two competing markets, and the market pricing strategy is $<0.1, 0.9>$. We also assume that there are 50 buyers and 50 sellers. We use the two algorithms to train the traders' trading strategies, respectively, to obtain the final Nash equilibrium trading strategy. The experiment is repeated for 50 times. In each experiment, I-PDQN algorithm will initialize the types of traders randomly in [0,1] under the uniform distribution. For FP algorithm, the types of traders and initial FP beliefs are also initialized randomly.

Figure 3 shows the average profits of traders when entering different markets when Nash equilibrium trading strategies are obtained by different algorithms. It can be seen that the results obtained by the above two algorithms are almost the same, which can prove that the I-PDQN algorithm can achieve the Nash equilibrium strategy the same as FP algorithm.

We also evaluate the computation speed of the two algorithms. We calculate the number of iterations and the computation time of each iteration. The average and standard deviation are calculated and the results are shown in Table 2.

The results show that although the I-PDQN algorithm has more iterations when converging to the equilibrium, the single iteration computation time of FP is about 5.031 times that of I-PDQN algorithm, and therefore, the total average time of FP algorithm is 4.6745 times that of I-PDQN algorithm. Therefore, we can see that using I-PDQN algorithm can compute the trader's Nash equilibrium trading strategy faster. The reason is that traders use I-PDQN algorithm interacts with the environment and others constantly, and they can obtain more experience tuples to train their own policy network, and therefore, they need more iterations. However, the algorithm only needs to calculate their own hybrid actions according to the current observed state, and therefore, it takes less time. In FP algorithm, traders need to calculate the current best response strategy against FP beliefs every iteration and update their FP beliefs. All traders will repeat this process until convergence. Therefore, with the increased number of traders, the computation time of each iteration in FP algorithm will increase, resulting in the increased total convergence time.



(a) Market distribution ratio of sellers

(b) Market distribution ratio of buyers

Figure 1: Market selection of traders with pricing strategy <0, 1 > .

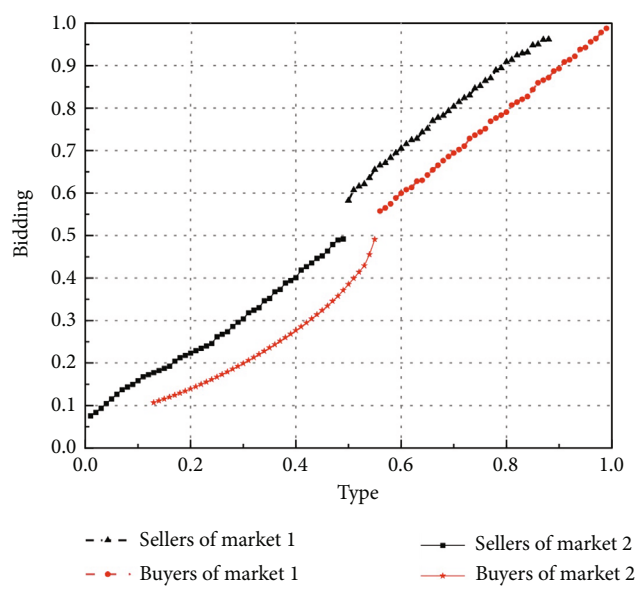


Figure 2: Nash equilibrium trading strategy of traders.

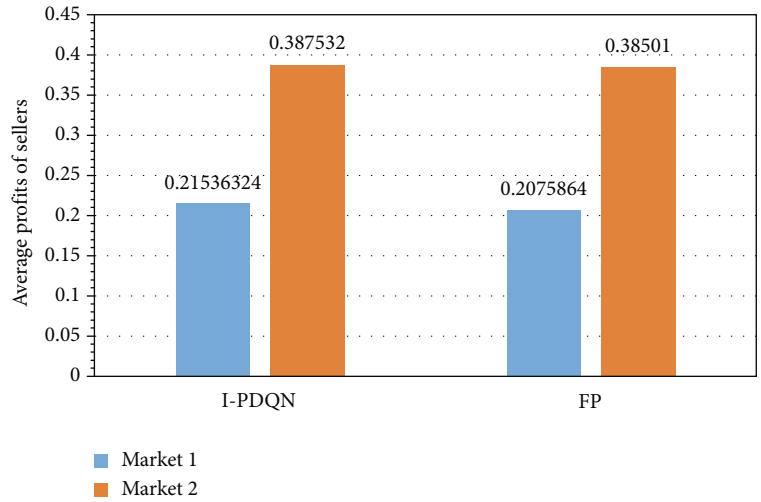
5. The Competing Pricing Strategy

After analyzing the Nash equilibrium trading strategy, we now analyze how the double auction market set the pricing strategy in Nash equilibrium. Specifically, we will use the MADDPG algorithm to design the competing pricing strategy and evaluate it against the FP algorithm in terms of computation speed and convergence result.

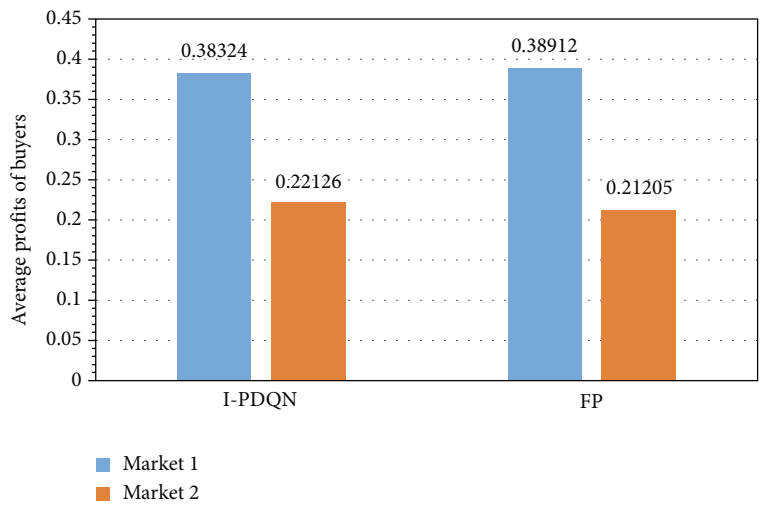
In the competing environment, the double auction market will adjust the pricing strategy in real-time in order to attract traders and obtain higher allocation efficiency. Intuitively, the pricing strategy and traders' Nash equilibrium strategy are affected by each other, and therefore, this is a joint learning process between the market and traders, which is shown in Figure 4. In the first stage, the market selects a pricing strategy based on the observed state. In the second stage, traders select the market and submit the bids according to the Nash equilibrium trading strategy. The competing market then compute the allocation efficiency according to the current actions of traders and then further update the pricing strategy in order to improve the allocation efficiency. This process is repeated to reach the equilibrium state. At this moment, we can obtain the Nash equilibrium pricing strategy and the Nash equilibrium trading strategy under this pricing strategy.

5.1. MADDPG Algorithm. As we have described in the above, the joint learning process is also a sequential decision process, and it involves two competing markets. This can be regarded as a Markov game. Therefore, we use Multiagent Deep Deterministic Policy Gradient algorithm [48] to analyze the Nash equilibrium pricing strategy.

The MADDPG algorithm is centralized trained and decentralized executed. Furthermore, each piece of experience replay contains the information of all agents at the current stage. Each agent learns multiple strategies, and at the same time use the overall effect of all strategies to do the optimization. The space complexity of MADDPG depends on the size of the replay memory D, which usually does not exceed the number of traders in the market, which is $O(\lambda n)$, where λ is the size of MDP tuples, and n is the size of replay memory. The same as I-PDQN, replay memory is cleared for each round. For the time complexity, it also cannot be accurately calculated. However, it can ensure that the convergence strategy is calculated in a reasonable time. Now, we will briefly introduce MADDPG. We use $\theta = [\theta_1, \cdots, \theta_n]$ to represent the parameters of the strategy of n agents, and $\pi = [\pi_1, \cdots, \pi_n]$ to represent the strategy of n agents. The cumulative expected reward for the agent i is $J(\theta_i) =$



(a) Average profits of sellers



(b) Average profits buyers

Figure 3: Average profits of traders in two markets of different algorithms.

Table 2: Number of iterations of different algorithms.

	Number of iterations		Each iteration time		Total time
	Average	Standard deviation	Average	Standard deviation	
I-PDQN	16215	0.08651	1.025	0.0958	16620
FP	15065	0.19156	5.157	0.1545	77690

$E_{s\sim\rho^{\pi},a_i\sim\pi_{\theta_i}}[\sum_{t=0}^{\infty}\boxtimes\gamma^t r_i, t]$, and for the deterministic strategy μ_{θ_i}, the gradient is

$$\nabla_{\theta_i} J(\mu_i) = E_{x,a\sim D}\left[\nabla_{\theta_i}\mu_i(a_i \mid o_i)\nabla_{a_i} Q_i^{\mu}(x, a_1,\cdots,a_n)\big|_{a_i=\mu_i(o_i)}\right], \tag{20}$$

where Q_i^{μ} is a value function for each agent. For the centralized critical, it is updated by minimizing the following loss function:

$$L(\theta_i) = E_{x,a,r,x'}\left[\left(Q_i^{\mu}(x, a_1,\cdots,a_n) - y\right)^2\right], \tag{21}$$

where the equation of y is

$$y = r_i + \gamma \bar{Q}_i^{\mu'}\left(x', a_1',\cdots,a_n'\right)\Big|_{a_j'=\mu_j'(o_j)}. \tag{22}$$

$\bar{Q}_i^{\mu'}$ is the target network, θ_j' is the parameter $\mu' = [\mu_1', \cdots, \mu_n']$ with lag update of the target strategy, and the strategies of other agents can be obtained by fitting approximation. It can be seen that critical can use global information, and actor only uses local observation information. If we know the actions of all agents, even if the strategies of each agent are constantly updated, the environment is stable. The MADDPG algorithm for designing the competing pricing strategy is shown in Algorithm 2, which takes the pricing

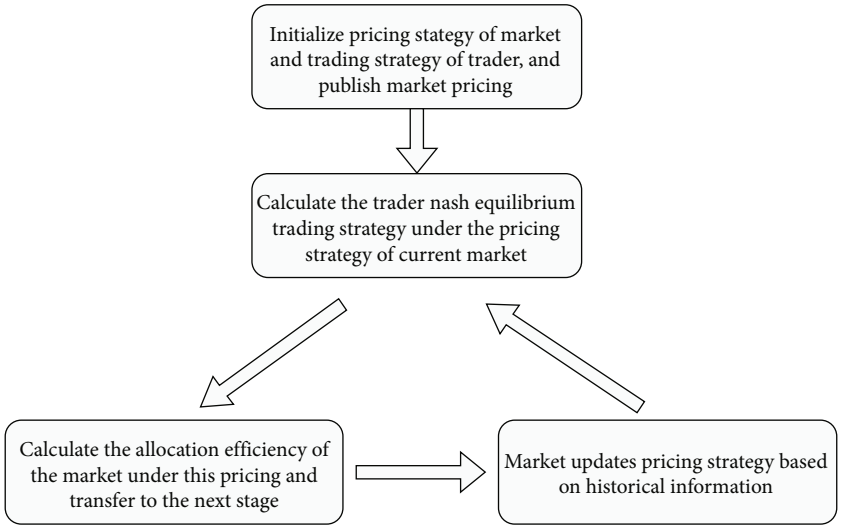


Figure 4: Dynamic pricing-joint learning process.

Input: continuous action space P_1 and P_2 of Market
Output: Equilibrium pricing strategies π^1 and π^2 of market M_1 and M_2
Initialization: Initialize the actor network μ and critical network Q of M_1 and M_2, respectively, and initialize the respective parameters θ^μ and θ^Q to initialize the target networks μ' and Q' corresponding to the above two networks and parameters $\theta^{\mu\prime}$ and $\theta^{Q\prime}$, initialize replay memory D
Random initialization distribution N for action exploration;
Initial respective market states s_0^1 and s_0^2, and set the iteration cycle to $t = 0$
while *The loss function of traders is not convergence* **do**
  Action selection:
  The market selects actions a_t^1 and a_t^2 according to $a_i^m = \mu(s_t^m \mid \theta^\mu) + \mathbb{N}_t$, respectively
  Release the pricing action $a_t = (a_t^1, a_t^2)$, then the trader adjusts his equilibrium trading strategy (I-PDQN algorithm) under the pricing, and then the market calculates the reward $r_t = (r_t^1, r_t^2)$ and the new state $s_{t+1} = (s_{t+1}^1, s_{t+1}^2)$
  Store tuples s_t, a_t, r_t, s_{t+1} in the replay memory
  Strategy training:
  for $i = 1, 2$ (*Update the strategy network for the two markets, respectively*) **do**
    Randomly sample r tuples from replay memory D and calculate y_i^m
    The critical network Q is updated by minimizing the loss function of equation (21)
    The actor network μ is updated by maximizing the gradient of the sample strategy through equation (20)
    Update the target network parameters $\theta^{Q'}$ and $\theta^{\mu'}$ through equation $\theta_i' \longleftarrow \tau\theta_i + (1-\tau)\theta_i'$
  end
end

Algorithm 2. MADDPG algorithm.

parameter space of the two markets as the input and output the Nash equilibrium market pricing strategy.

5.2. Experimental Analysis of Pricing Strategy. We now experimentally analyze the Nash equilibrium pricing strategy. The experimental setting is the same as that in I-PDQN. The state of each market is a tuple, expressed as $s^i = n_i^{\tilde{s}}, n_i^{\tilde{b}}, \text{ask}^{avr}, bid^{avr}, g_i$, where $n^{\tilde{s}}$ and $n^{\tilde{b}}$ are the number of buyers and sellers entering the market, bid^{avr} and ask^{avr} are the average bidding of both buyers and sellers, g is the number of deals. The pricing parameter spaces P_1 and P_2 of the two markets are between [0,1] and the replay memory size $D = 200000$. For the generation of the original pricing action, we use the normal distribution $a_t \sim N(a_0, 0.5)$ as the noise exploration. The number of samples taken for training each time is $\mathbb{R} = 128$, the learning rate of actor network is $\alpha^a = 0.001$, the learning rate of critical network is $\alpha^c = 0.0001$, the update factor of target network parameters is $\tau = 0.001$, and the discount factor is $\gamma = 0.9$.

5.2.1. Experimental Results. In this experiment, the two markets obtain the Nash equilibrium pricing strategy through continuous training. Figure 5 shows the action selection trend of the competing market in the iterative process. It shows that market 1 chooses higher pricing parameter in the initial stage and finally stabilized at $k = 1$. For market 2,

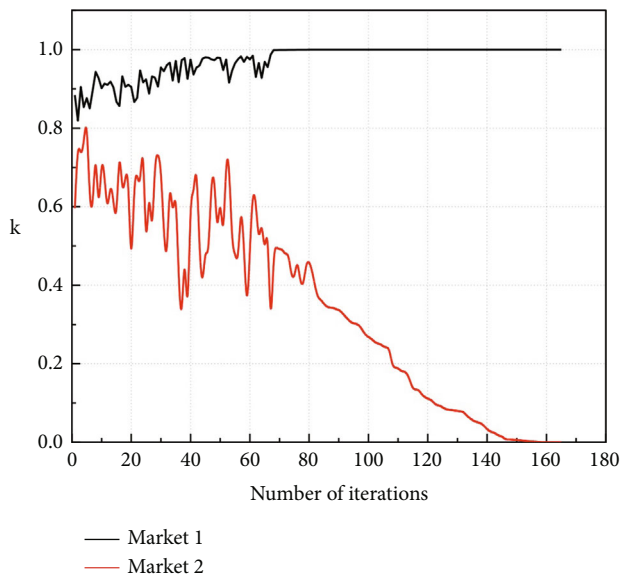


Figure 5: Dynamic pricing of competing markets.

since the higher pricing strategy of market 1 attracted a large number of sellers at the beginning, market 2 also tried to set a higher pricing parameter, that is, $k = 0.8$, but it cannot beat market 1. During this period, the action choice of market 2 fluctuated greatly, then gradually chooses a lower pricing parameter, and finally stabilized at $k = 0$. Actually, we have tried many experiments. The results show that the two competing markets will eventually stabilize at $k = 0$ and $k = 1$, that is, in the equilibrium state, the pricing parameter of market 1 is $k = 0$ and that of market 2 is $k = 1$ or vice versa, which is related to the initialized market network parameters. This means that the market will be in favor a class of traders, buyers, or sellers. In this case, the two markets can coexist. This further shows that in a highly competing environment, it is difficult for a market to attract all traders.

5.3. Experimental Evaluation against FP. Now, we evaluate our algorithm against FP algorithm in terms of the computation speed and the allocation efficiency. The parameters are the same as those in Section 4.2. Each experiment is repeated for 10 times, and then we compute the average result.

Under the equilibrium k pricing strategy, the experimental results show that when the algorithm converges, the final pricing strategies of the two algorithms are stable $k = 0$ and $k = 1$, where which pricing parameter the market adopts is related to the initial parameters or the initial FP belief. This shows that MADDGP algorithm will finally get the same result as FP algorithm.

Furthermore, we look into the convergence speed in different algorithms when they reach Nash equilibrium. The results are shown in Figure 6. It can be seen that when the pricing strategy converges, the average computation time of FP algorithm is 1.2 times that of MADDPG algorithm. This means that our algorithm can reach the equilibrium faster than FP.

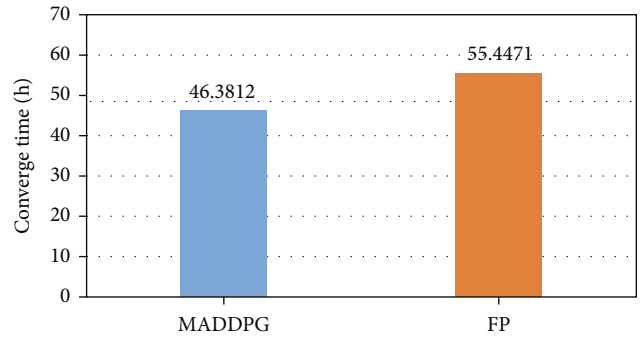


Figure 6: Dynamic pricing of competing market.

6. Conclusion

In this paper, we analyze the Nash equilibrium trading strategy of sensor data with a large number of traders in the competing environment with multiple edge servers running double auction markets. We adopt a deep reinforcement learning algorithm I-PDQN combined with the mean field theory to solve the Nash equilibrium trading strategy. In the experimental analysis, the Nash equilibrium result of the algorithm is consistent with that of FP algorithm, and the computation speed is significantly faster than FP algorithm. Then, we analyze how the edge server with double auction set the price effectively in the competing environment. We use MADDPG to compute the Nash equilibrium pricing strategy. The experimental result show that the Nash equilibrium pricing strategy of this algorithm is consistent with FP algorithm, and the computation speed is faster than FP algorithm. The analysis of this paper can provide some useful insights for designing the practical trading strategy and pricing strategy in the competing environment with multiple edge servers to trade sensor data.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper was funded by the Humanity and Social Science Youth Research Foundation of Ministry of Education (Grant no. 19YJC790111), the Philosophy and Social Science Post-Foundation of Ministry of Education (Grant no. 18JHQ060), and Shenzhen Fundamental Research Program (Grant no. JCYJ20190809175613332).

References

[1] Z. K. Xu, H. L. Zhang, X. Z. Yu, and Z. G. Zhou, "Combinatorial double auction-based allocation of retrieval tasks in internet of things," *Journal on Communications*, vol. 36, no. 12, p. 47, 2015.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[3] P. Chavali and A. Nehorai, "Managing multi-modal sensor networks using price theory," *IEEE Transactions on Signal Processing*, vol. 60, no. 9, pp. 4874–4887, 2012.

[4] X. Xu, J. Ma, and X. Xie, "Price convergence under a probabilistic double auction," *Computational Economics*, vol. 54, no. 3, pp. 1113–1155, 2019.

[5] P. Khezr and I. A. MacKenzie, "Consignment auctions," *Journal of Environmental Economics and Management*, vol. 87, pp. 42–51, 2018.

[6] F. Hu, B. Chen, J. Wang, M. Li, P. Li, and M. Pan, "MastDP: matching based double auction mechanism for spectrum trading with differential privacy," in *2019 IEEE Global Communications Conference*, pp. 1–6, Waikoloa, HI, USA, 2019.

[7] D. Kumar, G. Baranwal, Z. Raza, and D. P. Vidyarthi, "A systematic study of double auction mechanisms in cloud computing," *Journal of Systems and Software*, vol. 125, pp. 234–255, 2017.

[8] X. Wang and M. P. Wellman, "Spoofing the limit order book: an agent-based model," in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 651–659, San Francisco, California, USA, 2017.

[9] A. Bensoussan, J. Frehse, and P. Yam, *Mean Field Games and Mean Field Type Control Theory[M]*, Springer, 2013.

[10] J. Lasry and P. Lions, "Mean field games," *Japanese Journal of Mathematics*, vol. 2, no. 1, pp. 229–260, 2007.

[11] F. Sangoleye, N. Irtija, and E. E. Tsiropoulou, "Data acquisition in social internet of things based on contract theory," in *IEEE International Conference on Communications*, pp. 1–6, Montreal, QC, Canada, 2021.

[12] S. Chung, J. Lim, K. J. Noh, G. Kim, and H. Jeong, "Sensor data acquisition and multimodal sensor fusion for human activity recognition using deep learning," *Sensors*, vol. 19, no. 7, p. 1716, 2019.

[13] D. T. Ho, E. I. Grøtli, P. B. Sujit, T. A. Johansen, and J. B. Sousa, "Optimization of wireless sensor network and UAV data acquisition," *Journal of Intelligent and Robotic Systems*, vol. 78, no. 1, pp. 159–179, 2015.

[14] I. Maksymova, C. Steger, and N. Druml, "Review of lidar sensor data acquisition and compression for automotive applications," *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, no. 13, p. 852, 2018.

[15] C. Niu, Z. Zheng, F. Wu, X. Gao, and G. Chen, "Trading data in good faith: integrating truthfulness and privacy preservation in data markets," in *The IEEE 33rd International Conference on Data Engineering*, pp. 223–226, San Diego, CA, USA, 2017.

[16] X. Cao, Y. Chen, and K. J. R. Liu, "An iterative auction mechanism for data trading," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5850–5854, New Orleans, LA, USA, 2017.

[17] Y. Jiao, P. Wang, D. Niyato, M. A. Alsheikh, and S. Feng, "Profit maximization auction and data management in big data markets," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, San Francisco, CA, USA, 2017.

[18] L. Tian, J. Li, W. Li, B. Ramesh, and Z. Cai, "Optimal contract-based mechanisms for online data trading markets," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7800–7810, 2019.

[19] J. Mao, L. Tian, J. Zhang, G. Duan, and C. Wang, "Many-to-many data trading algorithm based on double auction theory," *Procedia Computer Science*, vol. 174, pp. 200–209, 2020.

[20] J. Yu, M. H. Cheung, J. Huang, and H. V. Poor, "Mobile data trading: behavioral economics analysis and algorithm design," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 4, pp. 994–1005, 2017.

[21] Y. Hui, Z. Su, and S. Guo, "Utility based data computing scheme to provide sensing service in internet of things," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 2, pp. 337–348, 2019.

[22] D. Niyato, M. A. Alsheikh, P. Wang, D. I. Kim, and Z. Han, "Market model and optimal pricing scheme of big data and internet of things (IoT)," in *IEEE International Conference on Communications*, pp. 1–6, Kuala Lumpur, Malaysia, 2016.

[23] A. E. al-Fagih, F. M. al-Turjman, W. M. Alsalih, and H. S. Hassanein, "A priced public sensing framework for heterogeneous IoT architectures," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 133–147, 2013.

[24] Y. Jiao, P. Wang, S. Feng, and D. Niyato, "Profit maximization mechanism and data management for data analytics services," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2001–2014, 2018.

[25] C. Chen and Y. Wang, "SPARC: strategy-proof double auction for mobile participatory sensing," in *International Conference on Cloud Computing and Big Data*, pp. 133–140, Fuzhou, China, 2013.

[26] W. Sun, Y. Liu, and H. Zhang, "Double auction-based resource allocation for mobile edge computing in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4692–4701, 2018.

[27] H. Cai, Y. Zhu, J. Li, and J. Yu, "Double auction for a data trading market with preferences and conflicts of interest," *The Computer Journal*, vol. 62, no. 10, pp. 1490–1504, 2019.

[28] D. K. Gode and S. Sunder, "Allocative efficiency of markets with zero-intelligence traders: market as a partial substitute for individual rationality," *Journal of Political Economy*, vol. 101, no. 1, pp. 119–137, 1993.

[29] G. W. Brown and N. J. Von, "Solutions of games by differential equations," *Contributions to the Theory Games*, vol. 27, no. 4, pp. 73–79, 1950.

[30] Z. Rabinovich, E. Gerding, M. Polukarov, and N. R. Jennings, "Generalised fictitious play for a continuum of anonymous players," in *The 21st International Joint Conference on Artificial Intelligence*, pp. 245–250, Pasadena, California, USA, 2009.

[31] B. Shi, E. H. Gerding, P. Vytelingum, and N. R. Jennings, "An equilibrium analysis of competing double auction marketplaces using fictitious play," in *The 19th European Conference on Artificial Intelligence*, pp. 575–580, Lisbon, Portugal, 2010.

[32] L. J. Schvartzman and M. P. Wellman, "Stronger CDA strategies through empirical game-theoretic analysis and reinforcement learning," in *The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 249–256, Budapest, Hungary, 2009.

[33] M. M. Chowdhury, C. Kiekintveld, T. C. Son, and W. Yeoh, "Bidding strategy for periodic double auctions using Monte Carlo tree search," in *The 17th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1897–1899, Stockholm, Sweden, 2018.

[34] J. Bredin and D. C. Parkes, "Models for truthful online double auctions," in *The 21st Conference on Uncertainty in Artificial Intelligence*, Toronto, Ontario, Canada, 2012.

[35] T. Miller and J. Niu, "An assessment of strategies for choosing between competitive marketplaces," *Electronic Commerce Research and Applications*, vol. 11, no. 1, pp. 14–23, 2012.

[36] K. Cai, J. Niu, and S. Parsons, "On the economic effects of competition between double auction markets," *Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis*, p. 88, 2010.

[37] B. Shi, Y. Huang, S. Xiong, and E. H. Gerding, "Setting an effective pricing policy for double auction marketplaces," in *Pacific Rim International Conference on Artificial Intelligence*, pp. 457–471, Phuket, Thailand, 2016.

[38] B. Shi and X. Li, "A game-theoretical analysis of charging strategies for competing double auction marketplaces," *Multi-Agent Systems and Agreement Technologies*, pp. 100–115, 2020.

[39] S. Chen and C. Tai, "Trading restrictions, price dynamics and allocative efficiency in double auction markets: analysis based on agent-based modeling and simulations," *Advances in Complex Systems*, vol. 6, no. 3, pp. 283–302, 2003.

[40] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *International Conference on Machine Learning*, pp. 805–813, Lille, France, 2015.

[41] J. Xiong, Q. Wang, Z. Yang et al., "Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space," 2018, https://arxiv.org/abs/1810.06394.

[42] H. Fu, H. Tang, J. Hao, Z. Lei, Y. Chen, and C. Fan, "Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces," in *The 21st International Joint Conference on Artificial Intelligence*, pp. 2329–2335, Pasadena, California, USA, 2019.

[43] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[44] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

[45] A. Majumdar, P. Benavidez, and M. Jamshidi, "Multi-agent exploration for faster and reliable deep Q-learning convergence in reinforcement learning," in *2018 World Automation Congress (WAC)*, pp. 1–6, Stevenson, WA, USA, 2018.

[46] M. Tan, "Multi-agent reinforcement learning: independent vs. cooperative agents," in *Proceedings of the Tenth International Conference on Machine Learning*, pp. 330–337, Amherst, MA, USA, 1993.

[47] S. Richard and G. Andrew, *Reinforcement Learning: An Introduction*, The MIT Press, London, 2018.

[48] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017.