energies

MDPI

Article

Peer-to-Peer Trading for Energy-Saving Based on Reinforcement Learning

Liangyi Pu [1,†], Song Wang [1,†], Xiaodong Huang [1,†], Xing Liu [2,†,‡], Yawei Shi [2] and Huiwei Wang [3,4,*]

[1] Chongqing Huizhi Energy Corporation Ltd., State Power Investment Corporation (SPIC), Chongqing 401127, China
[2] College of Electronics and Information Engineering, Southwest University, Chongqing 400715, China
[3] Key Laboratory of Intelligent Information Processing, Chongqing Three Gorges University, Chongqing 404100, China
[4] Chongqing Innovation Center, Beijing Institute of Technology, Chongqing 401120, China
* Correspondence: hwwang@swu.edu.cn
† These authors contributed equally to this work.
‡ Current address: Shaanxi Branch, Industrial and Commercial Bank of China (ICBC), Xi'an 710004, China.

Abstract: This paper proposes a new peer-to-peer (P2P) energy trading method between energy sellers and consumers in a community based on multi-agent reinforcement learning (MARL). Each user of the community is treated as a smart agent who can choose the amount and the price of the electric energy to sell/buy. There are two aspects we need to examine: the profits for the individual user and the utility for the community. For a single user, we consider that they want to realise both a comfortable living environment to enhance happiness and satisfaction by adjusting usage loads and certain economic benefits by selling the surplus electric energy. Taking the whole community into account, we care about the balance between energy sellers and consumers so that the surplus electric energy can be locally absorbed and consumed within the community. To this end, MARL is applied to solve the problem, where the decision making of each user in the community not only focuses on their own interests but also takes into account the entire community's welfare. The experimental results prove that our method is profitable both both the sellers and buyers in the community.

Keywords: peer-to-peer energy trading; multi-agent reinforcement learning; prosumer

Citation: Pu, L.; Wang, S.; Huang, X.; Liu, X.; Shi, Y.; Wang, H. Peer-to-Peer Trading for Energy-Saving Based on Reinforcement Learning. *Energies* **2022**, *15*, 9633. https://doi.org/10.3390/en15249633

Academic Editors: Chun-Cheng Lin, Igor Kotenko, Fausto Pedro García Márquez and Islam Md Rizwanul Fattah

Received: 24 October 2022
Accepted: 15 December 2022
Published: 19 December 2022

1. Introduction

Environmental benefits, financial incentives, energy conservation and emission reduction motivate households to install distributed power generation and energy storage devices [1]. Once users install renewable-based distributed generators, they have a revolutionary change in the role they pay in the electric energy market, that is, they become a prosumer who possess the capability to generate and consume electric energy and possibly also has demand response (DR) capacities [2]. As prosumers emerge, the energy infrastructure of a community should make innovative changes to adapt to the new concept.

An intuitive energy trading model of the community microgrid is directly trading with the main grid [3]. With this model, each user acts as an individual to independently interact with the main grid. After meeting daily demand, the user will sell the surplus electric energy to the main grid at a price lower than the market price to obtain economic benefits. One shortcoming is that in this model, the individual sells surplus electric energy to the supplier at a price much lower than the market price, which is fundamentally detrimental to the user. This model is relatively simple and has a formal operation in practice, but unfortunately, it fails [4].

The P2P model is widely used for the resource sharing problem in the field of computer science in which each computer (i.e., peers) located at the edge of a P2P network provides resources to the network and consumes resources that the network provides [5]. Similarly,

each prosumer in a community can generate electric energy to the community and also consume electric energy provided by the community; thus, the community can be naturally modeled as a P2P network [6]. The only difference is that the main grid will trade the electric energy with prosumers as a special participant. The P2P energy trading model is more flexible than the traditional energy trading model. One of the strengths of this P2P model is that users can choose the amount of energy to be traded according to their actual load requirements. Moreover, users can choose the transaction price, which allows users to maintain their own economic profits.

Recently, P2P energy trading has been attracting more and more attention from scientists and engineers. Most of them have primarily considered the model from two different points of view. One is the game-theoretic models have been proposed to deal with the P2P energy trading problem [7–9]. The other one is the P2P energy trading formulated as an optimization model and solved by some gradient-based algorithms [10,11]. In addition, Ref. [12] introduced blockchain technology to solve the P2P energy trading problem based on a decentralized platform. For more details, please refer to two recent survey papers [13,14] and the references therein.

Considering the drawbacks of model-based optimization approaches [15], reinforcement learning (RL) has been recently carried out to deal with the energy trading problem. The authors in [6] applied learning automaton-based RL algorithms to address energy trading among prosumers with incomplete information. Subsequently, batch RL algorithms were proposed to schedule controllable loads [16], which motivated us to explore the idea of appliance load division in this paper. The authors of [17,18] presented a dynamic pricing-based DR scheme, and Q-learning was employed for decision making. More recently, MARL has also dealt with the P2P energy trading problem. The authors of [19,20] proposed multi-agent deep deterministic policy gradient algorithms to deal with the double-side auction P2P energy trading problem by combining the technique of parameter sharing. The authors in [21] presented a long short-term delayed reward method to learn the long-term trading patterns and the short-term trading patterns at the same time for a better trading strategy. In addition, the concept of community energy storage was introduced in [22] for prosumers, and the authors proposed an RL-based algorithm for solving the P2P energy trading problem.

Despite these efforts, however, there still exist two significant limitations. First, most work focused on only one kind of appliance, such as thermostatically controlled load, that did not consider that reducing and shifting the elastically controllable load would trigger a variation in decision making. Second, to our best knowledge, research on multi-agents is relatively rare except [19,20] however, in that study, the authors formulated the P2P trading problem as a multi-agent coordination problem. We believe that in the P2P energy trading model, each user will decide the sell/buy volume and price of electric energy for maximizing his own profit, so each user is selfish and competitive with others, rather than cooperative.

To enrich the research into P2P energy trading, in this paper, we first provide novel insight into P2P energy trading in which we treat each prosumer in the community as an agent who interacts with other community members to realize economic efficiency. For individual users, we try to find a trade-off between comfort level and economic benefits by adjusting the elastic appliance loads and selling surplus electric energy. At the same time, from the perspective of the entire community, we need to balance the interests of sellers and buyers and consume the surplus electric energy locally. The main contributions of this paper are as follows:

- We build two interactive energy trading models, one is the household energy management system (HEMS), which ensures a trade-off between satisfaction and profits. The other one is the community energy management system (CEMS), which can balance the interests of sellers and buyers and absorb the surplus electric energy locally.
- We propose a new P2P energy trading model in the community as the interaction between multiple agents, which can be naturally applied. The decision making of each

user not only focuses on their own interests but also takes into account the welfare of the entire community.
- We perform simulated experiments to show that the balance between buyers and sellers in P2P energy trading is achievable and the energy trading is significant for both sides. Moreover, experimental results also report the trading behaviour of individual users in the community for more profits.

2. Preliminary

2.1. Reinforcement Learning and Q-Learning

All MARL frameworks can be described as a standard interaction process in which each agent interacts with the environment in search of a reward-maximizing policy [15], as shown in Figure 1. The environment receives the action from the agent and executes this action; the new state of the environment and a scalar reward are fed to the agent; the agent uses the information to optimize its policy and choose the next action. This relationship is modeled by the Markov Decision Process (MDP) via a quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$:

- $\mathcal{S}$ is the set of environment states, each of which is used to describe the real condition of the environment, maybe a tuple or a big matrix.
- $\mathcal{A}$ is the set of actions. An action $a \in \mathcal{A}$ shows the agent's behaviour in the environment. For example, we can control the direction and speed of the car when we drive.
- r is a scalar denoting the reward that evaluates the impact of the agent's current action on the environment.
- $\mathcal{P}$ is the state transition distribution that characterizes the rule of environmental changes, since the environment faces a new state after we take an action.
- $\gamma \in (0, 1)$ is defined as the discounted coefficient/factor.

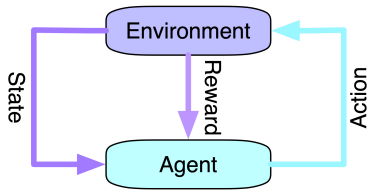


Figure 1. Reinforcement learning process between agent and its environment.

Focusing on the RL process, at each discrete time-step t, for given the current state $s_t \in \mathcal{S}$ and the policy $\pi : \mathcal{S} \to \mathcal{A}$, the agent selects an action $a_t \in \mathcal{A}$ and then receives a reward $r(s_t, a_t)$ and the next state s_{t+1} of the environment; the agent chooses the next action a_{t+1}. This process loops until time t is the final time-step. The main aim of RL is to maximize the cumulative reward in the process, that is to say, to maximize the return $R_t = \sum_{\tau=t+1}^{T} \gamma^{\tau-t-1} r(s_\tau, a_\tau)$, where T is the final time-step, and the return indicates the priority of long-term rewards.

The most famous algorithm in RL is Q-learning to solve the discrete Markov decision problem. First, the Q-learning process is based on the action–value function, which describes the expected reward after selecting an action a_t in state s_t and thereafter executing according to policy π satisfying:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_{\tau \geq t}, s_{\tau > t} \sim \mathbf{E}, a_{\tau > t} \sim \pi}[R_t | s_t, a_t] \tag{1}$$

where **E** is a stochastic environment. Action–value functions satisfy a recursive relationship expressed by the Bellman equation:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim \mathbf{E}}\left[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^\pi(s_{t+1}, a_{t+1})]\right].$$

Q-learning starts with a Q-table composed of all state–action pairs, then updates this table with the Bellman equation until it converges to the optimal policy [23]. Specifically, the

optimal policy is obtained by executing a greedy search strategy $\pi^* = \arg\max_a Q(s,a)$ on each state. Since the update in Q-learning is based on an action–value function, a convergent Q-table is necessary which decides whether Q-learning can converge to the optimal policy. Many RL algorithms are presented based on Q-learning to solve some existing questions [24,25].

2.2. Multi-Agent Reinforcement Learning and Value-Decomposition Networks

Many MARL frameworks have been developed to cooperatively solve the optimal policy in which each agent observes its own (local) reward and is responsible for seeking the current optimal action from its own action set only based on local observation. Traditional RL algorithms such as Q-learning can seldom be used to solve MARL problems. Generally, each agent has to face a non-stationary situation where the dynamic of its environment will change as other agents select some different actions. In addition, please notice that from an individual agent's perspective, the environment is partially observable; some surplus reward information may come from other agents' actions. The literature [26] showed that independent Q-learners cannot distinguish other agents' exploration due to stochasticity and fail to solve a very simple MARL problem.

A value-decomposition network (VDN) was proposed in [27], which solved those two problems existing in MARL frameworks. MARL is organized as a decentralized partially observable Markov decision problem (Dec-POMDP); each agent has its local observation o^i and action a^i. The core idea of VDN is:

$$Q(s,a) \approx Q(o^1,a^1) + Q(o^2,a^2) + \ldots \tag{2}$$

which decomposes the global Q-function into additional local Q-functions and verifies the effectiveness of VDN. For details, refer to [27].

3. Peer-to-Peer Energy Trading Model

In this section, we consider a community energy trading scheme involved with several household prosumers. Table 1 and Figure 2 show the notations used in the P2P energy trading model and the graphical model of this paper, respectively.

Table 1. Parameter notations of peer-to-peer energy trading system model.

Notation	Description
H	Time-step indicating an hour moment in a day
i	Index of each household energy management system
E_{ge}	Electricity volume of distributed energy generator
E_{ne}	Necessary and irreducible appliance load in HEMS
E_{con}	Elastic and controllable appliance load in HEMS
p	Unit electricity price of power grids
e_{ne}	Real consumption of necessary and irreducible appliance loads
e_{con}	Real consumption of elastic and controllable appliance loads
U	Electricity volume of HEMS sell to peers
P	Electricity price of HEMS wish to sell
α, β	Parameters of dissatisfaction function

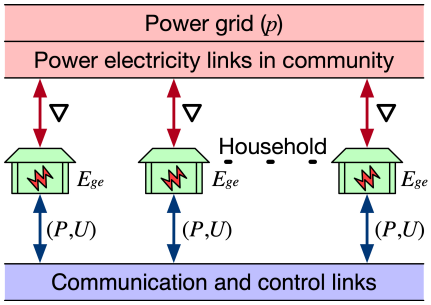


Figure 2. Peer-to-peer energy trading system model.

3.1. Household Energy Management System (HEMS)

We consider a household is equipped with one HEMS and several appliances, especially distributed energy generator devices, with the aim of optimizing energy consumption. HEMS is connected with other users in the community through a two-way transmission line that enables the exchange of electric energy. Each HEMS can receive information about energy consumption in the house as well as the price of electricity and decides the amount U and price P of energy to sell to peers. The electric appliances are separated into two types based on their characteristics and preferences of users, i.e., necessary load E_{ne} and controllable load E_{con}. For simplicity, the following mathematical analysis omits the index of HEMS.

The necessary load indicates the electric energy that users must consume in daily life. This part of electric energy guarantees their most basic life needs, such as refrigerators, lights, etc. Without this part of consumption, users will encounter difficulties in their daily life. So the real consumption e_{ne}^H of necessary load matches its demand E_{ne}^H, i.e.,

$$e_{ne}^H = E_{ne}^H. \tag{3}$$

The controllable load of appliances shows the flexibility of users. They may prefer to reduce electric energy consumption to reduce their bills. This type of equipment includes air conditioners and washing machines, which change the electric consumption by adjusting their working levels or scheduling their operation periods. The real consumption e_{con} of the controllable load is calculated by

$$e_{con} = E_{ge}^H - E_{ne}^H - U^H \tag{4}$$

where E_{ge} is the energy generated from a locally distributed energy generator, which is different at time H. U^H is the amount of energy that HEMS decides to sell at time H.

Selling surplus electric energy will realize a profit, and at the same time, users may give up the comfort level in the household to a certain extent. There are many satisfaction functions to describe satisfaction/dissatisfaction level, among which the most widely used are the quadratic function and the logarithmic function. In this work, we use the quadratic function to define the dissatisfaction of household users:

$$\varphi^H = \frac{\alpha}{2}(E_{con}^H - e_{con}^H)^2 + \beta(E_{con}^H - e_{con}^H) \tag{5}$$

where α and β are constants defined in Table 1.

3.2. Constraints of HEMS

The constraints are mostly focused on the amount and price of the electric energy to sell, namely U and P. Back to Equation (4), it should at least satisfy that $e_{con} = E_{ge}^H - E_{ne}^H - U^H \geq 0$, so that we have $U^H \leq E_{ge}^H - E_{ne}^H$. Now, considering the case that $E_{ge}^H - E_{ne}^H - E_{con}^H > 0$,

the HEMS should sell the surplus energy to peers because there are no energy storage devices in the house. So, we attain the range of U:

$$E_{ge}^{H} - E_{ne}^{H} - E_{con}^{H} \leq U^{H} \leq E_{ge}^{H} - E_{ne}^{H}. \tag{6}$$

At time H, when $U^H > 0$, it means that HEMS is selling electric energy to the P2P trading system and vice versa.

About P, we make the following assumption to illustrate constraints on it.

Assumption 1. *Peers are always the first choice.*

At time H, the electricity price of the power grid is p^H, the electricity price that the user directly sells to the power grid is $p^H - k$, where k is the residual constant. Therefore, a constraint to P^H is $p^H - k < P^H < p^H$. This assumption essentially encourages sellers/buyers to participate in the P2P energy trading system instead of directly interacting with the power grid.

3.3. Community Energy Management System (CEMS)

In the previous part, we mainly focused on a single HEMS. Its most important function is to determine the amount and electricity price at which the user wants to sell. We introduced the dissatisfaction function for a household and explained the constraints on its decision. With this subsection, we illustrate the energy dispatch after all households have their decisions on U^H and P^H. From Figure 2, we see that each HEMS decides to sell energy U^H, but how the sold energy is dispatched to other users still needs a strategy to solve this problem. Given this problem, we solve it with the following process.

Step 1. Classification: Dividing users into two sets, namely a buyer set B and a seller set F, according to U^H. When $U^H > 0$, the user belongs to the seller set F and vice versa.

Step 2. Sorting and Allocation: Denote an energy tuple (U_i^H, P_i^H) of a user i. First, we sort the seller set F in ascending order by P_i^H. Then, beginning allocation, we take out the user with the lowest price from the seller set and dispatch the electricity to each buyer in proportion. Therefore, the electric energy allocated to each buyer with P_i^H is:

$$Disp_i = \frac{U_{i\_buy}^H}{\sum_{i \in B} U_{i\_buy}^H} \times U_{i\_sell}^H. \tag{7}$$

After the electricity with the lowest price is allocated, the user with the second-lowest price is taken from the seller set B to continue the above process until alternatively the seller set or the buyer set is empty.

Step 3. Fixing the Result: As mentioned in Step 2, the allocation process will end when the seller set or the buyer set is empty. However, there may still be unmet needs of some users; so in the end, users who have demand will directly interact with the power grid after the P2P trading process ends.

After the above three steps, we can compute that the income obtained by each seller in the seller set B is:

$$Utility_i = P_i^H \times \sum_{i \in B} Disp_i + \nabla \tag{8}$$

where ∇ shows the profit at which the user sells the remaining energy to the power grid. The cost of the each buyer in buyer set F is:

$$Cost_i = \sum_{i \in F} (P_i^H \times Disp_i) + \nabla. \tag{9}$$

So, the objectives of seller and buyer at time H, respectively, are:

$$Objective_{seller} = Utility_i - \varphi^H, \tag{10}$$

$$Objective_{buyer} = Cost_i + \varphi^H. \tag{11}$$

Now, we know that the goals of the entire P2P community energy trading model are maximizing Equation (10) and minimizing Equation (11). Since the goal of reinforcement learning is to maximize the reward function, we write the objective function of the P2P energy trading community as:

$$Objective = \rho * Objective_{seller} - (1 - \rho) * Objective_{buyer} \tag{12}$$

where ρ is a mixed parameter.

4. Multi-Agent Reinforcement Learning for P2P Energy Trading

In this section, MARL is used to solve the problems formed in the previous section; that is to say, the ultimate goal is to let each user make the correct decision based on its own information. This correct decision can benefit both the buyers and the sellers in the community. Below, we first establish an MARL environment for the P2P trading model, and then we solve the optimal decision-making problem of each agent based on the VDN algorithm.

4.1. Establishing MARL Environment

Before starting this subsection, we need to launch an assumption that the future is predictable, which is reasonable. First of all, the electricity price can be divided into the static electricity price and dynamic electricity price. When our model uses the static electricity price, the power grid will release the corresponding electricity price one day in advance. For the dynamic electricity price and the electricity volume of distributed energy generators, the assumption is accomplishable relying on the current predictive technology based on deep neural networks [28–30].

Reinforcement environment for HEMS
Observation: $o = (E_{ge}, E_{ne}, E_{con}, p^H)$
Action: $a = (U^H, P^H)$
Reward: $r = Objective$ in (12)

For each agent, the RL environment is the same, as listed in the above table. Please notice that each agent does not have separate reward information, and the reward information is total for the entire community. From the above table, we view the tuple $(E_{ge}, E_{ne}, E_{con}, p^H)$ as the user's local observation information o^i. Here, HEMS is implemented by a neural network that uses o^i as input, and its output is the value function $Q(o^i, a^i)$ of each action. There is no reward information for individual HEMS, meaning the network cannot be trained. So, we superimpose the Q-function of each user and obtain the global Q-function of the entire community. That is to say, we suppose the global action value function $Q_{global}(S, A)$ can be decomposed into a local action value function:

$$\begin{aligned} Q_{global}(S, A) &= Q_{global}((o^1, o^2, \dots), (a^1, a^2, \dots)) \\ &\approx Q_{local}(o^1, a^1) + Q_{local}(o^2, a^2) + \dots. \end{aligned} \tag{13}$$

With this assumption, we can use the reward information of the community to achieve end-to-end training for the local network. The whole architecture of our networks is shown in Figure 3.

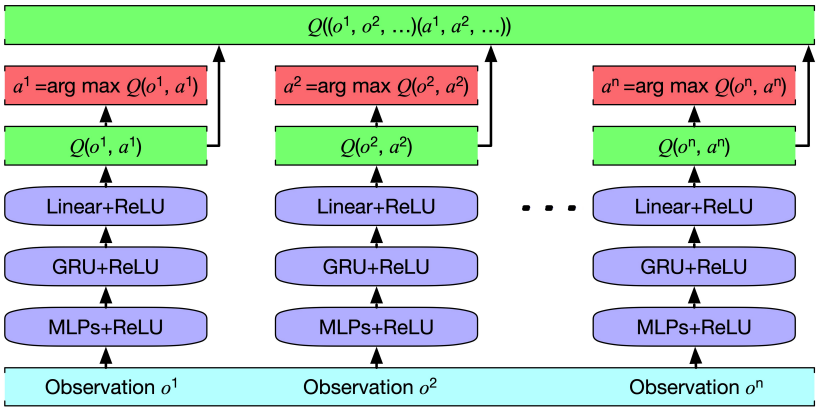


Figure 3. Architecture of neural networks used in the MARL environment.

It can be seen that the single-user HEMS architecture mainly uses the multi-layer perceptron (MLP) networks [31] and the gated recurrent unit (GRU) network [32]. MLP networks have been widely used in classification and nonlinear prediction. GRU is a new architecture of the recurrent neural network, which was proposed in 2014. Compared with long-short term memory (LSTM) neural network, GRU uses less training parameter and therefore uses less memory and executes faster on simple datasets. For the activation function of the network layer, we uniformly use the rectified linear unit (ReLU) nonlinear activation function [33], i.e., $\mathrm{ReLU}(x) = \max\{0, x\}$. The backpropagation gradient algorithm Adam [34] is used to train the networks.

4.2. Training and Executing

Centralized training and distributed execution (CTDE) have almost become the standard paradigm for multi-agent reinforcement learning [35]. In the training process, calculating the global reward must use the information of each agent. If the training is distributed, then training each agent requires other agents to transmit data to it, which will increase the cost. Distributed execution is completely feasible. For each agent, after the training, we can take a greedy strategy on the trained Q-network to obtain the optimal strategy, which only requires local information o^i as input.

In order to simplify training, we use the method of sharing parameters. Since the network structure of each agent (HEMS) is exactly the same, only a local network Q_{local} is needed to approximate the action–value function for each agent. It is worth noting that we add the additional input feature to this network for identifying which user is using this network. Specifically, we increase the input dimension and use the one-hot encoding to form the data to indicate the user index. In this way, the training process will become relatively simple because there is actually only one action–value network Q that needs to be trained.

The complete training process is shown in Algorithm 1, which is divided into two parts: interacting with the environment to obtain data and training a neural network. In the process of collecting data, we first select the action of each agent according to the ϵ-greedy strategy. That is to say, we randomly select the action with the probability of ϵ and choose the greedy action $a^i = \arg\max Q_i(o^i, a^i)$ with probability $1 - \epsilon$; ϵ-greedy policy actually expresses the balance between exploration and exploitation in our training process. Randomly selecting actions for exploration may encounter relatively low rewards, but at the same time, it is possible to seek the actions that have not been selected before. The greedy policy would obtain a relatively high reward, while it means there will never be an improvement. The parameter ϵ will decrease as the training episodes are increasing.

Algorithm 1 Training Process of RL Algorithm

Initialize: Action value network $Q(o, a|\theta)$ and $Q'(o, a|\theta')$
Initialize: Replay buffer R size M
  for episode $= 1, 2, \cdots, N$ **do**
    Initialize an environment state $(o^1, o^2, \dots)$ for begin exploration
    for time $H = 1, 2, \cdots,$ **do**
      for each agent i = 1,2, …, **do**
        Select a random action a_H^i with probability ϵ
        Otherwise select $a_H^i = \max_a Q(o_H^i, a)$
        Execute the action a_H and observe the next o_{H+1}^i
      end for
      Get global reward r_H and store transition $(o_H^i, a_H^i, r_H, o_{H+1}^i)$ in R
      Sample random minibatch of transitions $(o_H^i, a_H^i, r_H, o_{H+1}^i)$ from R
      Compute target y_H with (14) and (15)
      Using (16) as loss to update the network Q
    end for
    Every T episodes, copy parameters of Q to Q'
  end for

When we have collected enough data, we will start training. Our training process is similar to the deep Q-learning (DQN) algorithm [36] because both of them are to train the action–value function network until it converges. The difference is that our environment has only one global reward r; we train the network through the following process. First, we calculate the target value y_H

$$y_H = \begin{cases} r_H, & H \text{ is the terminal} \\ r_H + \gamma \max Q_{global}(S_H, A_H), & \text{Otherwise} \end{cases} \tag{14}$$

where H represents the time, and γ is the discount coefficient. We substitute (13) into (14) to obtain the target value when the current time is not the terminal

$$\begin{aligned} y_H &= r_H + \gamma \max(Q_{global}((o_H^1, o_H^2, \dots), (a_H^1, a_H^2, \dots))) \\ &\approx r_H + \gamma \max(Q_{local}(o_H^1, a_H^1) + Q_{local}(o_H^2, a_H^2) + \dots) \\ &= r_H + \gamma \max(Q(o_H^{1'}, a_H^1) + Q(o_H^{2'}, a_H^2) + \dots) \\ &\approx r_H + \gamma \max(Q'(o_H^{1'}, a_H^1) + Q'(o_H^{2'}, a_H^2) + \dots) \end{aligned} \tag{15}$$

where Q' is the target network whose structure is exactly the same as the network Q. If we use the network Q to calculate the target value, it may make the training process unable to converge. So, we use the target network Q' to replace the network Q to approximately calculate the target value, where $o_H^{1'}$ and $o_H^{2'}$ are input data that contain features to identify the user index. This training technique has proven its effectiveness in DQN [36] and the deep deterministic policy gradient (DDPG) algorithm [37].

After calculating the target value y_H, we employ the following loss function and Adam [34] optimization algorithm to train the network.

$$Loss = (y_H - (Q(o_H^{1'}, a_H^1) + Q(o_H^{2'}, a_H^2) + \dots))^2. \tag{16}$$

When the training process is completed, we will obtain a convergent network to approximate the action–value of each agent. The network can be executed in a distributed manner. For each agent, its local state o and user index are the input data, and the output is the result of the action–value function for each action. Here, we use a greedy policy to obtain the optimized action; that is, the action with the largest action–value is taken as the

decision for the user at this time-step, which includes the amount of energy to be traded and the transaction price.

5. Experiment Detail

5.1. Experiment Setting

This subsection first gives the parameter setting in a small-scale CEMS environment and the RL training process as shown in Tables 2 and 3.

Table 2. Parameter setting of CEMS environment.

Notation	Value	Notation	Value
Time interval ΔH	1 h	α	20
Number of users in community	5	β	0.1

Table 3. Parameter setting of RL training process.

Notation	Value	Notation	Value
Episodes N	6000	Minibatch size	512
Experience replay size M	1500	Reward decay coefficient γ	0.9
ϵ	1	ϵ reduce every episode	2.5×10^{-4}
Target network update interval T	100		

In the previous section, we assumed that the grid-side electricity price, user load, and electricity volume of the user's distributed power generation devices are known. Now, we generate these data to simulate the real scenario. For the electricity volume data of the user's distributed generation devices, we randomly sample data from the Poisson distribution [38], while at the same time limiting its maximum size to 12. In 1 day, we divide 24 h into 4 regions, where each region has different Poisson distribution parameter λ, as shown in Table 4. For the user's different loads, we acquire the user's necessary load E_{ne} and controllable load E_{con} from the two sequences $[3, 4, 5, 6]$ and $[2, 3, 4, 5]$ with different probabilities, respectively, as also shown in Table 4.

Table 4. Poisson distribution parameter and load choosing probability for different users and regions.

User	Poisson Distribution: λ for Region Index				Probability: Pro. for Region Index			
Index	1	2	3	4	1	2	3	4
1	2.667×10^{-7}	0.541	6.5965	4.3712	0.5	0.2	0.2	0.1
2	8.8281	10.2997	9.8301	9.7514	0.2	0.5	0.1	0.2
3	8.8281	10.2997	9.8301	9.7514	0.2	0.1	0.5	0.2
4	2.667×10^{-7}	0.541	6.5965	4.3712	0.1	0.2	0.2	0.5
5	8.8281	10.2997	9.8301	9.7514	0.4	0.5	0.1	0

In order to illustrate this further, we list the load and the electricity volume of the distributed generation devices of "User 1" in one day, as shown in Figure 4. From the picture, we can see that at some specific time-slots, the user would have surplus electricity after the user meets the necessary load, such as $H = 18$. In contrast, the user has no surplus electricity to sell at time $H = 4$ and may buy energy from other users. For the model adapted to electricity price, we use the square electricity price model [39] which belongs to the static price model. When the time $H > 7$, the price is 20, otherwise the price is 15. Finally, we set the parameter of Assumption 1 as $k = 5$.

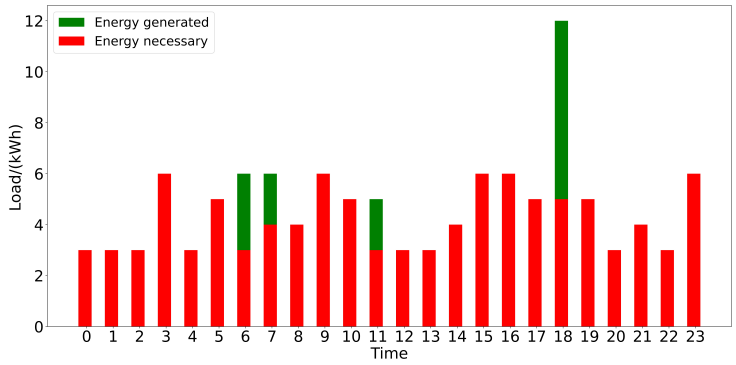


Figure 4. Necessary load and energy generation of "User 1".

Now, we set the parameters for the neural network of our single agent. We actually train only one neural network when we use the method of sharing parameters; its architecture is exactly the same as that of the single network in Figure 3. The number of hidden layer neurons is set in Table 5. The input dimension of the network is nine, where the data dimension of the local state is four and the dimension of user identification is five. For the output of the network, we will make a specific description in the next subsection.

Table 5. Neuron number of our network.

Hidden Layer	Number of Neurons	Hidden Layer	Number of Neurons
Linear layer 1	256	Linear layer 2	128
GRU layer	128	Linear layer 3	128

5.2. Experimental Results

This paper employs an optimized policy to a convergent action–value function network. Now, we verify its convergence properties. In the training process, we use the ϵ-greedy strategy to achieve the balance between the exploration and exploitation of the action space in which, with the increase of training episodes, ϵ will gradually decrease to make the agent prefer choosing the greedy action to obtain the large reward. At the same time, choosing the appropriate environmental reward function has a decisive effect on RL performance. The reward function in (12) contains the mixed parameter ρ, which means different ρ may have have an impact on the convergence of loss and reward. Here, we evolve the loss and reward by choosing different values in $[0, 1]$, as shown in Figure 5, where we record the loss value for each episode and test the total reward every 50 episodes. From the figure, we can see that for all of ρ, the loss will eventually reach a stable convergence, but ρ has a greater impact on the final reward. When $\rho = 0.3$, the network can obtain a more expected final reward than other values of ρ.

We choose $\rho = 0.3$. Even so, we still need to verify in the environment whether this decision is beneficial for both the sellers and buyers in the community. The base situation we compared with is that users in the community are always directly trading with the grid side. We compute the seller's profits and the buyer's costs under different ρ in one day, as shown in Figure 6. Thanks to using the MARL guided P2P trading strategy, the total profits of all sellers in the community are higher than the base situation. At the same time, the total cost of buyers is less than when the user directly trades with the grid side. Therefore, the P2P energy trading model is profitable for both the buyer and the seller, which will always be of benefit to sellers and buyers in comparison with the base situation and also guide the users to sell/buy appropriate electricity to reach a better balance in the community.

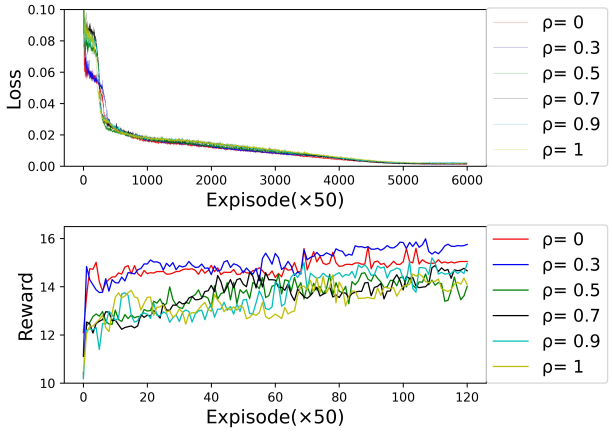


Figure 5. Evolution of loss and reward in training process.

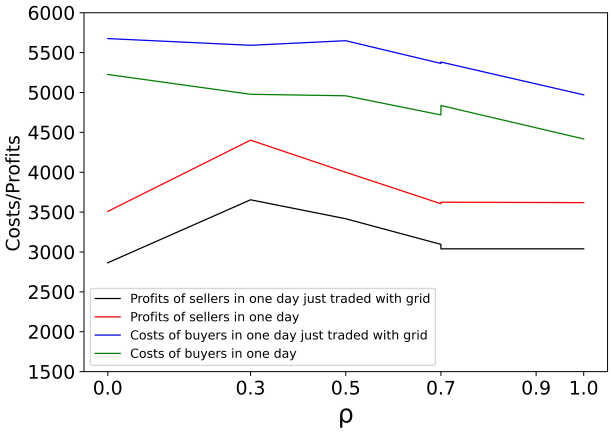
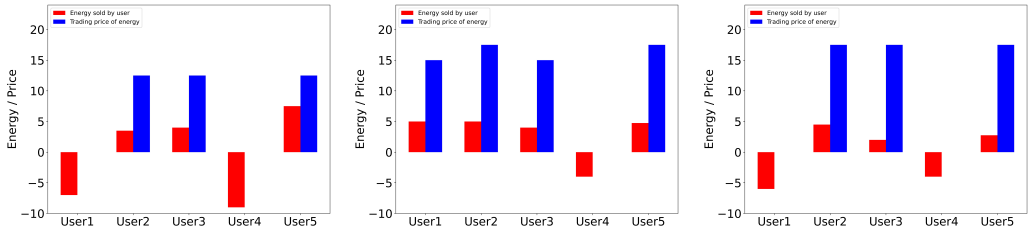


Figure 6. Profits/costs with different ρ.

From Figure 6, when $\rho > 0.3$, although the buyer's cost has been flat and falling as ρ increases, at the same time the seller's profit will be falling more and flat as ρ gradually increases. In this case, the total rewards will gradually decrease in comparison with the peak ($\rho = 0.3$) as ρ increases from 0.3. When $\rho < 0.3$, not only does the seller's profit decrease as ρ decreases, but also the buyer's cost increases as ρ decreases, whether trading with the main grid or the seller. In this case, the total rewards will gradually decrease in comparison with the peak ($\rho = 0.3$) as ρ decreases from 0.3. In summary, we can conclude that the parameter $\rho = 0.3$ is the best choice. Therefore, the following experiments will be conducted with $\rho = 0.3$.

In the P2P energy trading, there are three states: demand over supply, supply over demand and balance between supply and demand in the community. Now, we will show the decision-making process for an agent under those three states. As depicted in Figure 7, we can see that when CEMS is in the state of demand over supply or balance, the electricity price of sellers will be higher, even approaching the grid price. For example, when $H = 21$, the electricity price of the grid is 20, the transaction price given by the seller is 17.5. It is understandable because as long as the price the seller charges is less than the grid price, the buyer's actual cost will be lower than directly trading with the grid. At the same time, referring to Figure 4, we see that when "User 1" is at $H = 21$ and $H = 4$, the electricity generated by its own distributed devices is not enough to support the operation of the controllable load, so "User 1" buys energy to meet the demand of the controllable load for increasing its satisfaction. When the system is in the state of supply over demand ($H = 18$), the transaction prices "User 1" and "User 3" charge are lower than those of "User 2" and "User 4". We have the following interpretation according to our previous distribution rules. Sellers with low prices preferentially sell energy to other users; sellers need to compete to

sell energy to buyers when the system is in the state of supply over demand, so "User 1" and "User 3" charge lower prices, ensuring participation in the P2P trading. Those results prove that the agent's policy obtained by MARL works in the P2P energy trading model.



(**a**) (**b**) (**c**)

Figure 7. Agent's decisions at different time-slots: (**a**) demand over supply ($H = 4$); (**b**) supply over demand ($H = 18$); (**c**) balance state ($H = 21$).

The output dimension of the network is actually the discretization size of the action space of the RL agent, defined by the notation 'DS' for short. In the above experiment, we employ DS $= 8$, which means that the output dimension of the network is 8. Below, we try to find the best action discretization size for training our network. Figures 8 and 9 show the training process and test results under different action–discretization scales, respectively. We can see that for the same training episodes (6000), the reward is almost stable when DS $= 8$, but when DS $= 16$ or DS $= 32$, the model has the potential to surpass the performance of the model with DS $= 8$. This is reasonable because the fine-grained dispersion of prices will cause the seller's agent to sell its surplus electricity at a level closer to the grid-side electricity price. When DS $= 16$ or DS $= 32$, however, the seller's profits in one day will increase, but at the same time, the buyer's costs will also increase. Therefore, we conclude that DS $= 8$ is enough to solve our current problem as well as reduce the amount of calculation.

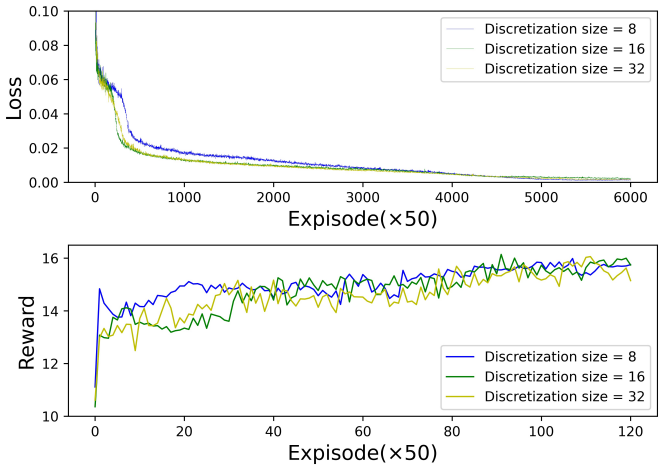


Figure 8. Evolution of loss and reward in the training process with different discretization size.

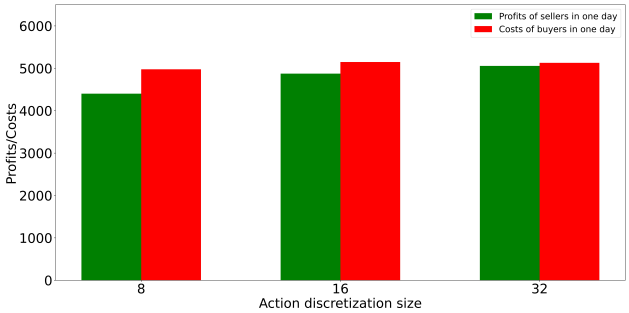


Figure 9. The performance of different action discretization size.

6. Conclusions

In this article, we propose a new perspective based on MARL to solve the P2P energy trading among prosumers in the community. In such a community, each prosumer, who aims at maximizing his own profit and increase his sense of well-being, can generate and consume electric energy. Hence, the P2P energy trading mechanism designed in this paper is based on people's feelings and needs. The developed MARL algorithm tries to address two problems at once: (1) for an individual user, it tries to balance the satisfaction and cost-saving (if buyer) or profit (if seller); (2) for the entire community, it also cares about the balance of interests between the sellers and buyers. Experimental results demonstrate that the method based on MARL can successfully solve the P2P trading model among members of the community.

Future works will mainly focus on addressing two limitations to this study. First, instead of generating the synthesized data with specific mean and deviations, we hope to apply real-world energy transaction data of prosumers in the P2P community. Second, the community energy storage is the development trend of the future [22]; we will consider the community energy storage integrated into the P2P energy trading, which implies the agent in RL algorithms will add an extra role that focuses on the profit only.

Author Contributions: L.P., S.W. and X.H. designed the P2P trading mechanism, Y.S. and H.W. conceived the algorithm and application details, X.L. performed the simulations and prepared the manuscript. S.W. and H.W. led the project and research. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Natural Science Foundation of Chongqing under Grant cstc2020jcyj-msxmX0057, in part by the Science and Technology Research Program of Chongqing Municipal Education Commission under Grant KJZD-M202201204, and in part by the China Postdoctoral Science Foundation under Grant 2022M720453.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kirthiga, M.V.; Daniel, S.A.; Gurunathan, S. A methodology for transforming an existing distribution network into a sustainable autonomous micro-grid. *IEEE Trans. Sustain. Energy* **2012**, *4*, 31–41. [CrossRef]
2. Kanchev, H.; Lu, D.; Colas, F.; Lazarov, V.; Francois, B. Energy management and operational planning of a microgrid with a PV-based active generator for smart grid applications. *IEEE Trans. Ind. Electron.* **2011**, *58*, 4583–4592. [CrossRef]
3. Chua, S.C.; Oh, T.H.; Goh, W.W. Feed-in tariff outlook in Malaysia. *Renew. Sustain. Energy Rev.* **2011**, *15*, 705–712. [CrossRef]
4. Jenkins, N.; Long, C.; Wu, J. An overview of the smart grid in Great Britain. *Engineering* **2015**, *1*, 413–421. [CrossRef]
5. Krishnan, R.; Smith, M.D.; Telang, R. The Economics of Peer-to-Peer Networks. 2003. Available online: https://ssrn.com/abstract=504062 (accessed on 1 June 2022).
6. Wang, H.; Huang, T.; Liao, X.; Abu-Rub, H.; Chen, G. Reinforcement learning for constrained energy trading games with incomplete information. *IEEE Trans. Cybern.* **2016**, *47*, 3404–3416. [CrossRef]
7. Zhang, C.; Wu, J.; Cheng, M.; Zhou, Y.; Long, C. A bidding system for peer-to-peer energy trading in a grid-connected microgrid. *Energy Procedia* **2016**, *103*, 147–152. [CrossRef]

8. Paudel, A.; Chaudhari, K.; Long, C.; Gooi, H.B. Peer-to-peer energy trading in a prosumer-based community microgrid: A game-theoretic model. *IEEE Trans. Ind. Electron.* **2018**, *66*, 6087–6097. [CrossRef]
9. Liu, N.; Cheng, M.; Yu, X.; Zhong, J.; Lei, J. Energy-sharing provider for PV prosumer clusters: A hybrid approach using stochastic programming and stackelberg game. *IEEE Trans. Ind. Electron.* **2018**, *65*, 6740–6750. [CrossRef]
10. Long, C.; Wu, J.; Zhang, C.; Cheng, M.; Al-Wakeel, A. Feasibility of peer-to-peer energy trading in low voltage electrical distribution networks. *Energy Procedia* **2017**, *105*, 2227–2232. [CrossRef]
11. Long, C.; Wu, J.; Zhang, C.; Thomas, L.; Cheng, M.; Jenkins, N. Peer-to-peer energy trading in a community microgrid. In Proceedings of the 2017 IEEE Power & Energy Society General Meeting, Chicago, IL, USA, 16–20 July 2017; pp. 1–5.
12. Esmat, A.; de Vos, M.; Ghiassi-Farrokhfal, Y.; Palensky, P.; Epema, D. A novel decentralized platform for peer-to-peer energy trading market with blockchain technology. *Appl. Energy* **2021**, *282*, 116123. [CrossRef]
13. Soto, E.A.; Bosman, L.B.; Wollega, E.; Leon-Salas, W.D. Peer-to-peer energy trading: A review of the literature. *Appl. Energy* **2021**, *283*, 116268. [CrossRef]
14. Tushar, W.; Saha, T.K.; Yuen, C.; Smith, D.; Poor, H.V. Peer-to-Peer Trading in Electricity Networks: An Overview. *IEEE Trans. Smart Grid* **2020**, *11*, 3185–3200. [CrossRef]
15. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
16. Ruelens, F.; Claessens, B.J.; Vandael, S.; De Schutter, B.; Babuška, R.; Belmans, R. Residential demand response of thermostatically controlled loads using batch reinforcement learning. *IEEE Trans. Smart Grid* **2016**, *8*, 2149–2159. [CrossRef]
17. Lu, R.; Hong, S.H.; Zhang, X.; Ye, X.; Song, W.S. A perspective on reinforcement learning in price-based demand response for smart grid. In Proceedings of the 2017 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 14–16 December 2017; pp. 1822–1823.
18. Lu, R.; Hong, S.H.; Zhang, X. A dynamic pricing demand response algorithm for smart grid: Reinforcement learning approach. *Appl. Energy* **2018**, *220*, 220–230. [CrossRef]
19. Qiu, D.; Ye, Y.; Papadaskalopoulos, D.; Strbac, G. Scalable coordinated management of peer-to-peer energy trading: A multi-cluster deep reinforcement learning approach. *Appl. Energy* **2021**, *292*, 116940. [CrossRef]
20. Qiu, D.; Wang, J.; Wang, J.; Strbac, G. Multi-Agent Reinforcement Learning for Automated Peer-to-Peer Energy Trading in Double-Side Auction Market. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21), Montreal, QC, Canada, 19–27 August 2021.
21. Kim, J.G.; Lee, B. Automatic P2P Energy Trading Model Based on Reinforcement Learning Using Long Short-Term Delayed Reward. *Energies* **2020**, *13*, 5359. [CrossRef]
22. Zang, H.; Kim, J. Reinforcement Learning Based Peer-to-Peer Energy Trade Management Using Community Energy Storage in Local Energy Market. *Energies* **2021**, *14*, 4131. [CrossRef]
23. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
24. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
25. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
26. Bernstein, D.S.; Givan, R.; Immerman, N.; Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.* **2002**, *27*, 819–840. [CrossRef]
27. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv* **2017**, arXiv:1706.05296.
28. Tang, N.; Mao, S.; Yu, W.; Nelms, R.M. Solar Power Generation Forecasting with a LASSO-based Approach. *IEEE Internet Things J.* **2018**, *5*, 1090–1099. [CrossRef]
29. He, W. Load forecasting via deep neural networks. *Procedia Comput. Sci.* **2017**, *122*, 308–314. [CrossRef]
30. Kuo, P.H.; Huang, C.J. An electricity price forecasting model by hybrid structured deep neural networks. *Sustainability* **2018**, *10*, 1280. [CrossRef]
31. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
32. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
33. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the ICML, Haifa, Israel, 21–24 June 2010.
34. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
35. Hostallero, W.J.K.D.E.; Son, K.; Kim, D.; Qtran, Y.Y. Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research, Virtual, 16–18 April 2019.
36. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]

37. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
38. Consul, P.C.; Jain, G.C. A generalization of the Poisson distribution. *Technometrics* **1973**, *15*, 791–799. [CrossRef]
39. Zhang, Z.; Qin, Z.; Zhu, L.; Weng, J.; Ren, K. Cost-Friendly Differential Privacy for Smart Meters: Exploiting the Dual Roles of the Noise. *IEEE Trans. Smart Grid* **2017**, *8*, 619–626. [CrossRef]