# DSM Pricing Method Based on A3C and LSTM Under Cloud-edge Environment

Fangyuan SUN[1], Xiangyu KONG[1*], Jianzhong WU[2], Bixuan GAO[1], Ke CHEN[3], Ning LU [4]
(1. Key Laboratory of Smart Grid of Ministry of Education (Tianjin University), Nankai District, Tianjin 300072, China;
2. School of Engineering, Cardiff University, Cardiff CF24 3AA, UK;
3. Key Laboratory of Demand Side Multi-Energy Carriers Optimization and Interaction Technique, China Electric Power Research Institute, Beijing 100192, China;
4. Electrical and Computer Engineering Department, North Carolina State University, Raleigh, NC 27606, USA.;)

**Abstract—**Demand-side management (DSM) could realize "peak cutting and valley filling" of power load and improve the stability and efficiency of power system. With the development of information systems, more smart devices are being deployed on the demand side. It has become a challenge for DSM service providers to take full use of the edge computing capacity and demand-side information to improve the accuracy of DSM decision-making, without revealing user privacy. This paper proposes a distributed DSM pricing method for service provider, based on asynchronous advantage actor-critic (A3C) algorithm and long short-term memory (LSTM) network under cloud-edge environment. The on-site utilization of user information is realized through distributed training and centralized decision-making structure of A3C algorithm. The training process is accelerated by LSTM based virtual environment, which greatly reduces the training cost of the algorithm. Case study results shows that the proposed method is able to make pricing decision for DSM service provider under cloud-edge environment. Moreover, through the combination of LSTM based virtual environment and A3C algorithm, the proposed method requires less historical data than other methods and improves the profit of service providers.

**Index Terms**— Demand-side management, LSTM, A3C, cloud-edge environment.

## 1 INTRODUCTION

With the continuous access of new energy sources such as wind power and photovoltaic, and the increase of flexible loads such as electric vehicles, the source side and load side of the power system has a strong volatility. Demand-side management (DSM) has been more and more frequently used to reduce the fluctuation and peak-valley difference of power demand [1]. DSM can be divided into incentive-based DSM and price-based DSM. Price-based DSM can guide users' energy consumption behavior without direct control, and has received widespread attention due to its flexibility and openness [2].

The development of the electricity market has changed the DSM mode from administrative management method to dynamic pricing schemes, such as time of use (TOU), real time pricing (RTP), and critical peak pricing (CPP). This leads to a sharp increase in the number and diversity of users participating in DSM. It is obviously difficult for the power system dispatcher to directly control such a large number of end-users. Instead, the DSM process is usually assigned to DSM service providers (load aggregators), and the dispatcher only puts forward the requirements for DSM operation and rewards the service providers who achieves these requirements.

Apart from changes in the market environment, with the construction of power Internet of Things, the 'cloud-edge' structure is gradually adopted by power system [3], in which numerous preliminary calculations are finished by edge computing devices, and the processed information are uploaded to the decision center [4]. The above structure can effectively solve the communication and calculation problem when decentralized user participate in DSM in a large scale. However, because many kinds of user information are not available to DSM service provider, it is very difficult to accurately grasp user behavior characteristics and make precise decisions. How to implement price-based DSM under the 'cloud-edge' structure has become a hot spot in power market research.

### 1.1 Literature review and motivation

Different from traditional centralized DSM, DSM under cloud-side environment mainly focus on the hierarchical information application and the interaction between cloud-side and edge-side[5]. That is, how to make full use of load-side information and upload only the key information to decision center.

Current researches on cloud-edge DSM mainly use model-based methods, including load behavior model and optimization model. Different from traditional centralized DSM method, load behavior model describes not only the load's behavior characteristics but also the interaction with the cloud-side control center, and optimization model is usually multi-level model, taking both cloud-side benefit and edge side benefit into consideration. Jiang [6] built a DSM control model for air conditioning load under cloud-edge environment. Dual-feedback closed-loop control method is used to increase the robustness of the algorithm. G. Belli [7] proposed a hierarchical optimization model of demand response considering electrical and thermal equipment. The upper layer determines energy prices based on market environment, and the lower layer realize DSM through information interaction with the upper layer. Moghaddam [4] built an information interaction mode between load and power system based on fog nodes, and a day-ahead

• *E-mail addresses: eekongxy@tju.edu.cn (Xiangyu Kong), wuj5@cardiff.ac.uk (Jianzhong Wu).*

**Nomenclature**

*Sets and parameters:*

| | | | |
|---|---|---|---|
| $d_{max}$ | Maximum response in all training data | $\alpha$ | Learning rate for Actor network training |
| $d_{min}$ | Minimum response in all training data | $\beta$ | Learning rate for Critic network training |
| $d_{target,t}$ | Load regulation target at time $t$ | $\gamma$ | Discount factor for Markov decision process |
| $\overline{d}_{c,t}$ | Average response during previous DSM periods of the day | $\lambda_{r,t}$ | Reward price form utility company at time $t$ |
| $\overline{d}_{s,t}$ | Average response during period $t$ on last three days | $\lambda_{p,t}$ | Deviation punishment form utility company at time $t$ |
| $D_{c,t}$ | Impact factor of historical data in previous DSM periods of the day | $\delta$ | TD-error |
| $D_{s,t}$ | Impact factor of historical data in period $t$ on related days | $c$ | Entropy factor for Actor network training |
| $\mathbf{H}$ | Historical data set for LSTM training | $\omega$ | Parameters of Critic network |
| $\overline{p}_{c,t}$ | Average deviation proportion during previous DSM periods of the day | $\theta$ | Parameters of Actor network |
| $\overline{p}_{s,t}$ | Average deviation proportion during period $t$ on last three days | $\varepsilon_{t,t-j,c}$ | Parameter to calculate $D_{c,t}$ |
| $re$ | Number of related days | $\varepsilon_{t,t-nT,s}$ | Parameter to calculate $D_{s,t}$ |
| $\mathbf{R}_t$ | Relative information set for user response simulation | | |
| $t$ | Index of DSM time period | *Variables:* | |
| $T$ | Number of DSM time period in one day | $a_t$ | Action chosen by pricing policy at time $t$ |
| $x_{max}$ | Upper limit of DSM price | $d_t$ | End-users' response to DSM price $x_t$ at time $t$ |
| $x_{min}$ | Lower limit of DSM price | $r_t$ | Reward of action $a_t$ at time $t$ |
| $\overline{x}_{c,t}$ | Average DSM price in previous DSM periods of the day | $s_t$ | State of environment at time $t$ |
| $\overline{x}_{s,t}$ | Average DSM price in period $t$ on related days | $x_t$ | DSM price at time $t$ |

DSM optimization model that comprehensively considered the interests of users and utility companies was established. Dutra [8] proposed a two-phase optimization framework for DSM program. Both incentive-based and price-based demand response are modeled, and a distributed approach is applied to solve the resulting problem while preserving users' privacy. Heydar [9] proposed a tri-objective model for residential smart electrical distribution grid scheduling, considering renewable energy sources and DSM. Loss of load expectation and load deviation minimization are considered to maximize operation revenue of distribution grid.

There have been many researches on model-based DSM method. However, when the number of end-users is large, the behavior patterns of different users vary greatly, and drawbacks of model-based methods become more prominent [10]. For model-based methods, complexity of optimization is closely related to that of load model. Traditional universal load behavior models cannot properly reflect the diversity of small loads. Adopting more complex load behavior models is a feasible way to cover the diversity of end-users, but it brings great difficulty to the solution of optimization model [11]. Heuristic algorithm can solve this problem to a certain extent, but it usually needs hundreds or thousands of iterations to get only one optimal decision, which makes it unsuitable for scenarios that require fast decision-making, such as real-time DSM [12]. Another drawback of heuristic algorithms is that it is easy to fall into the local optimal solution when solving high-dimensional problems, and the stability of the algorithm cannot be always guaranteed [13].

Reinforcement learning (RL) is a model-free method that can decouple the complexity of load modeling and decision optimization, and it can build a decision strategy in advance to save the computing time in practice, avoiding drawback of heuristic algorithms[14]. RL can be regarded as a constant trial-and-error process, which is shown in Fig.1. In each RL training episode, RL algorithm takes an action according to current state, and environment returns the reward of the action. RL algorithm uses this feedback to update decision strategy. The environment is the application scenario, and it is usually substituted by a virtual environment such as load behavior model in DSM scenario. Different RL algorithms have different forms of decision strategy (neural networks, probability distributions, tables, etc.) and strategy updating method. The interaction in Fig.1 makes decision optimization and load modeling independent, avoiding the influence of complex load models on optimization process. There are two key points of RL based DSM method: RL algorithm and environment [15].

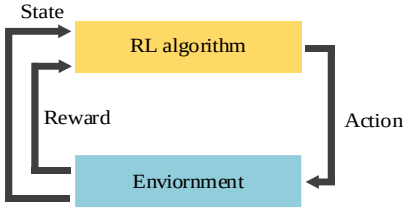



Fig.1 Interaction between RL algorithm and environment

At present, there have been many researches on the application of RL algorithm in DSM pricing scenarios. Kim [16] built a DSM pricing algorithm based on Q-learning algorithm for DSM implementation in microgrids. The convergence process of the algorithm was speed up by virtual experience method. Dehghanpour [12] built a hierarchical retail power market with air conditioning load DSM. Q-learning algorithm was used for day-ahead DSM pricing. Various retailer models

were analyzed to maximize the benefits of end-users and retailers. Lu [17] proposed a RL based pricing algorithm, which can balance the benefit of DSM service provider and end-users in hierarchical power market. The pricing process was based on end-user's response characteristics and dissatisfaction. Xu [18] adopted DDPG algorithm for joint bidding and pricing of Load Serving Entity, and the dynamical bid response and price response functions were simulated by Artificial Neural Network. Bahrami [19] adopted decentralized actor-critic RL algorithm to speed up the training process, and transformed the problem of updating neural network parameters into a sequence of semidefinite programs to deal with nonconvex power flow constraints. In [20], an operation model for buildings to participate in DSM program under cloud-edge environment is proposed, and the distributed training method is adopted to relieve communication pressure.

In DSM scenario, environment is generally end-users or electrical devices participating in DSM. However, it usually takes thousands of trial-and-error steps to acquire a proper DSM pricing strategy. It is obviously impractical to implement this training process in real environment [21]. Establishing a virtual environment as a substitution of the real environment to participate in training process is major solution to this problem [22]. Wang [23] proposed a deep RL method for DSM of interruptible load. Virtual environment is composed of power flow model of distributed network and behavior model of interruptible loads. Because the behavior logic of interruptible loads is fixed, the virtual environment can precisely describe the real environment. Li [24] proposed a DSM scheduling algorithm based on deep RL. A systematic simulation model of different kinds of appliances and resident's activities was built to replace real end-users in RL training process. Du [25] proposed a multi-microgrid energy management based on RL. To protect user privacy, deep neural network is used to simulate the response behavior of microgrid and participate in RL training process. Lork [26] proposed a Q-learning based algorithm for air-conditioner load management. A Bayesian Convolutional Neural Network based virtual environment is established to model air-conditioner loads and room temperature.

Table 1 is the comparison of current researches on RL based DSM. From Table 1, most of current researches adopt centralized training structure and model-based virtual environment. Although many researches and methods have been obtained on RL based DSM pricing, there are two main gaps under cloud-edge environment.

(1) To establish an accurate load model, current researches tend to divide load into several types, such as unchangeable load, shiftable load [12][17], interruptible load [23] or some special load like air conditioning (AC) load [24], and analyse them separately. However, due to user privacy rules, it is hard for service providers to get the necessary user information such as the type or capacity of users' electrical appliances, which makes it difficult to distinguish these kinds of load. Generally, only historical data of DSM price and response amount are available, and how to accurately simulate users' response behavior through basic energy consumption data without revealing user privacy is the first problem.

(2) When users participate in DSM on a large scale, transmitting user data directly to cloud side will cause huge communication and calculation pressure [3]. Therefore, RL training process should be implemented in a distributed way. Building a distributed pricing method to fit the cloud-edge physical system is the second problem.

(3) Q-learning or deep Q-learning (DQN) are the most popular algorithm in current researches. Because Q-learning algorithm needs to calculate a certain value for every feasible decision, when the problem has a high dimensional decision space, Q-learning algorithm have difficulty in convergence or large errors. Therefore, when the feasible region of DSM price is large, accuracy of Q-learning based pricing algorithm is unstable. Building a DSM pricing method suitable for the large decision space is the third problem.

For gap (1), a virtual environment based on Long short-term memory (LSTM) can be established. Neural network can realize the accurate simulation of user behavior through certain historical data without obtaining detailed user privacy information, such as the type and capacity of user's appliances. LSTM is a form of recurrent neural network and often used for the prediction of sequence data with strong correlation [27]. Due to the strong time-related characteristics of user response behavior, LSTM is very suitable for simulation of user's response behavior [28].

For gap (2) and gap (3), the distributed Asynchronous advantage actor-critic (A3C) algorithm for DSM pricing can be used [29]. For gap (2), the training process in A3C is accomplished in a multi-thread way, and each edge device forms a thread. The multi-thread just fit the cloud-edge environment, making most of the calculations finished by edge devices, and reducing the information transmission pressure. For gap (3), A3C algorithm can generate the probability distribution of the feasible region through neural network. Comparing to calculating a certain value for every feasible decision in Q-learning, this can significantly reduce the complexity of the algorithm, especially in high dimensional cases. At the same time, A3C algorithm has stronger exploration ability due to its multi-thread learning structure, which greatly reduces the training time of the algorithm [14].

Table 1 comparison of researches on RL based DSM

| Reference | RL algorithm | Virtual environment | Training structure | Objective |
|---|---|---|---|---|
| [12] | Q-learning | model-based | centralized | DSM pricing |
| [16] | Q-learning | model-based | centralized | DSM pricing |
| [17] | Q-learning | model-based | centralized | DSM pricing |
| [18] | DDPG | model-based | centralized | DSM pricing |
| [19] | actor-critic | model-based | decentralized | Load control |
| [20] | A3C | model-based | centralized | Load control |
| [23] | Q-learning | model-based | centralized | Load control |
| [24] | Q-learning | model-based | centralized | Load control |
| [25] | Q-learning | model-free | centralized | Load control |
| [26] | Q-learning | model-free | centralized | Load control |

### 1.2 Objective and contributions

A DSM pricing method for DSM service providers based on A3C and LSTM under cloud-edge environment is proposed. The A3C based pricing method is in a distributed way to fit the cloud-edge structure. A LSTM based virtual environment is developed to substitute for real environment in A3C training process. The main contributions of this paper include:

(1) A A3C based cloud-edge DSM framework is proposed, containing utility company, DSM service provider and end-user. With the framework, DSM pricing in day-ahead and real-time cases can be implemented under cloud-edge structure.

(2) A LSTM based virtual environment is established to substitute for real environment in training process. Through LSTM method, the simulation accuracy is guaranteed without revealing users' privacy.

(3) A A3C based DSM pricing algorithm is proposed. The distributed training structure of A3C algorithm is designed to fit the cloud-edge environment and improve DSM pricing accuracy.

The remainder of this paper is organized as follows: Section II introduces the scenarios and the basic process of DSM based on A3C. Section III introduces the Markov decision process of DSM and the establish of virtual environment based on LSTM network. Detailed training process based on A3C algorithm is provied in Section IV. Section V introduces the evaluation of the proposed algorithm through simulations, and Section VI gives some valuable conclusions.

## 2 PROBLEM FORMULATION

### 2.1 DSM under cloud-edge environment

The DSM scenarios under cloud-edge environment described in this paper is shown in Fig.2. The DSM program is accomplished by utility company, service provider and end users.

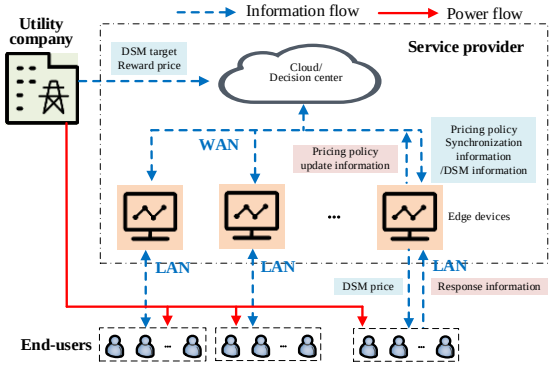



Fig.2 DSM control structure under cloud-edge environment

The utility company is generally a power grid company, and it gives guidance for DSM according to the power supply and demand relationship during DSM periods. The guidance includes DSM target, reward price and deviation punishment price. When the imbalance of energy supply-consumption is serious, the utility company will increase reward price and appropriately reduce deviation punishment price to improve the service provider's income and promote users to participate in DSM. Service provider can get reward by responding to DSM target, and the reward will be reduced if there are deviation between DSM target and actual response.

Service provider is the implementer of demand side management, which is composed of decision center and demand-side edge devices. It sets DSM prices to achieve peak-cutting or valley-filling. Service provider uses DSM compensation price to contact with end-users, rather than through direct load control. End-users won't be punished for excessive or insufficient response amount, and service provider should set an appropriate price to reduce the deviation between DSM target and actual response. The DSM price is decided in decision center. Edge measuring devices obtain the actual response amount and other relevant information, and edge computing devices participate in part of the calculation process of DSM pricing.

End-users obtain the current DSM price from the service provider in each DSM period, and adjust their power consumption to get DSM compensation. To guarantee the implementation of DSM, end-users should sign a contract with the service provider before participating in DSM. The contract includes detailed process of the reporting of user response capacity, the calculation of load base line, and the settlement of response compensation. However, end-users can independently decide whether to respond and how much to respond.

### 2.2 Service provider model

The objective of the proposed method is to maximize service provider's reward by choosing the most profitable DSM price. The objective function, and constraints of DSM pricing under cloud-edge environment can be expressed by (1) and (2):

$$\max R = \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} \left(\lambda_{\mathrm{r},t} - x_t\right) d_t - \lambda_{\mathrm{p},t} \left| d_t - d_{\mathrm{target},t} \right| \tag{1}$$

$$s.t. \quad \begin{cases} x_{\min} \le x_t \le x_{\max} \\ 0 \le d_t \le \eta d_{\mathrm{target},t} \end{cases} \tag{2}$$

Where, $r_t$ is the reward of service provider during one DSM period. $T$ is the amount of DSM period in a day. $x_t$ is the DSM price at period $t$, and it is the independent variable. $d_t$ and $d_{\mathrm{target},t}$ are the end-user's actual response and load regulation target at period $t$. $\lambda_{\mathrm{r},t}$ and $\lambda_{\mathrm{p},t}$ are reward price and deviation punishment price, $\lambda_{\mathrm{r},t}$ means how much utility company will pay service provider for load regulation. $\lambda_{\mathrm{p},t}$ means the punishment service provider should afford if there is deviation between $d_t$ and $d_{\mathrm{target},t}$. $\eta$ is the upper limit factor of users' total response, and if $d_t$ exceeds $\eta d_{\mathrm{target},t}$, there will be a risk of imbalance between supply and demand in power grid.

The main difficulty of this problem is described as follows:

The service provider should not only consider the reward of a single period, but should also take total reward $R$ into consideration. In the day-ahead DSM cases, $d_{\mathrm{target},t}$, $\lambda_{\mathrm{r},t}$, and $\lambda_{\mathrm{p},t}$ in each period are known, and the above model can be solved directly. However, in other cases, such as real-time DSM, the $d_{\mathrm{target},t}$, $\lambda_{\mathrm{r},t}$, and $\lambda_{\mathrm{p},t}$ in latter periods are unknown, so the model cannot be solved directly.

What's more, because the end-users' current response can influence their future behaviors, just maximizing $r_t$ in current period may reduce long-term reward[30]. For example, if an end user can only participate DSM for one hour and is more willing

to response at 10:00 p.m., the global optimal solution will be an appropriate DSM price at 10:00 p.m. But if the algorithm can't foresee this in advance, it may adopt a higher DSM price to stimulate this end user to response at an earlier period, rather than waiting until 10:00 p.m.

To adapt the pricing method to various DSM cases, and to enable the pricing method to have a high decision-making ability, A3C algorithm is used for DSM pricing.

### 2.3 Structure of proposed pricing method

The structure of A3C based DSM pricing method is shown in Fig.3, which is composed of training process and application process. Training process is to get the optimal decision strategy, and application process is to use this strategy to make decisions. In proposed method, the decision strategy is represented by neural networks, which are referred to as decision network in following part of this paper. Input of decision network is only the current state, and it doesn't need information in latter periods. This makes it suitable for both day-ahead and real-time DSM. Output of decision network will be the optimal DSM price, if it gets sufficient training.

RL (including A3C) based training process can be denoted by markov decision process, as shown in Fig.1. Essentially, it can be regarded as a trial-and-error process. Decision network output the optimal action $a_t$ according to current state $s_t$, and environment returns the action's reward $r_t$ and new state $s_{t+1}$($t$ represents the time). Then, the decision network is trained through these key variables ($s_t$, $a_t$, $r_t$, and $s_{t+1}$), to improve the total reward.

For A3C based DSM pricing markov decision process (DPMDP), decision maker is service provider, and environment is external power system, including end-users, power grid and utility company. A distributed training process is also adopted based on A3C algorithm to fit cloud-edge structure, which is shown in Fig.3. Through this distributed training structure, key variables ($s_t$, $a_t$, $r_t$, and $s_{t+1}$) are only used on edge side, greatly reducing calculation and communication pressure.

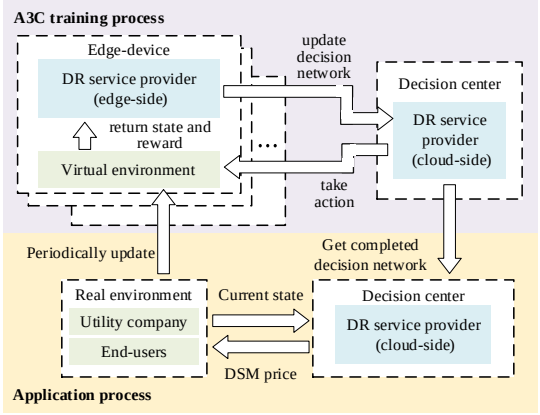



Fig.3 Structure of A3C based DSM pricing method

It usually needs thousands of training steps to acquire optimal decision network. During the beginning of the training process, the network can hardly make optimal decisions. Therefore, it will generate high cost to implement the training process in real environment. To solve this problem, a virtual environment is established as a substitution of real environment in training process, and real environment only participate in application process, which is also shown in Fig.3. Considering that users' response behaviors have strong time-related characteristics, virtual environment is composed of LSTM networks.

From the structure of the proposed method, there are three main steps to implement above-mentioned training process, which are shown in Fig.4.

(1) DPMDP is established, including key variables and a virtual environment. To guarantee the optimality of pricing, key variables contain all factors that reflects the load behavior and affects the revenue of service provider. Virtual environment is established through LSTM to simulate end-users' response behavior.

(2) A3C based DSM pricing method is established, including DSM pricing by decision center, decision network updating by edge devices, and illustration of information transmission under cloud-edge environment.

(3) Test the method in actual environment, including virtual environment accuracy test and A3C algorithm decision ability test.

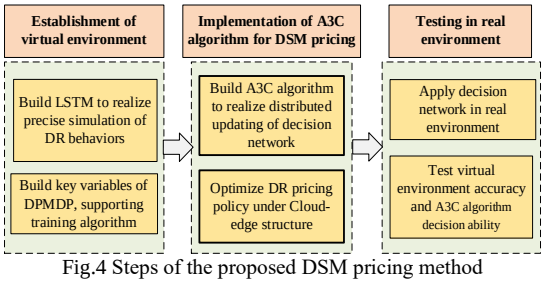



Fig.4 Steps of the proposed DSM pricing method

## 3 Establishment of DPMDP

### 3.1 Key variables of DPMDP

The key variables of DPMDP are state $s_t$, action $a_t$, reward of the action $r_t$, and the next state $s_{t+1}$. In DSM scenario, state $s_t$ represents the load regulation target and other related information at time $t$. Action $a_t$ represents the DSM price $x_t$ adopted by service provider. Next state $s_{t+1}$ represents the state after adopting DSM price $x_t$. Reward $r_t$ represents the profit of the service provider during 1 time period after adopting DSM price $x_t$. The specific equations of the above key variables are expressed by (3), (4), (5) and (6) :

$$a_t = x_t \in \left[x_{\min}, x_{\max}\right] \tag{3}$$

$$s_t = \left(d_{\text{target},t}, \lambda_{\text{r},t}, \lambda_{\text{p},t}, \bar{d}_{c,t}, \bar{x}_{c,t}, \bar{p}_{c,t}, \bar{d}_{s,t}, \bar{x}_{s,t}, \bar{p}_{s,t}\right) \tag{4}$$

$$s_{t+1} = \phi\left(a_t, s_t\right) \tag{5}$$

$$r_t = \left(\lambda_{\text{r},t} - x_t\right)d_t - \lambda_{\text{p},t}\left|d_t - d_{\text{target},t}\right| \tag{6}$$

where, $\phi$ is the environment that generates $s_{t+1}$ based on $a_t$ and $s_t$. $x_{\min}, x_{\max}$ are the upper and lower limit of DSM price. $\bar{d}_{c,t}$ and $\bar{d}_{s,t}$ are the average responses in the previos DSM periods on the day, and in the same DSM period on relative days. $\bar{x}_{c,t}$ and $\bar{x}_{s,t}$ are the average DSM prices in the previos DSM periods on the day, and in the same DSM period on relative days. $\bar{p}_{c,t}$ and $\bar{p}_{s,t}$ are the average deviation proportions in the

previos DSM periods, and in the same DSM period on relative days. The deviation proportion $p_t$ represents the deviation between actual response and load regulation target, which can be expressed by (7):

$$p_t = \left| d_{\text{target},t} - d_t \right| / d_t \tag{7}$$

In $s_t$, the first three dimensions $\left(d_{\text{target},t}, \lambda_{\text{r},t}, \lambda_{\text{p},t}\right)$ are directly used for decision-making, and the last six dimensions $\left(\bar{d}_{c,t}, \bar{x}_{c,t}, \bar{p}_{c,t}, \bar{d}_{s,t}, \bar{x}_{s,t}, \bar{p}_{s,t}\right)$ are used to make the algorithm familiar with the user's response behavior characteristics.

$\phi$ in (5) has various forms of expression. For model-based virtual environment, $\phi$ can be a series of equations. For model-free virtual environment, $\phi$ can be neural networks. In this paper, it is a series of LSTM networks.

From (6), the revenue of service provider mainly depends on the accuracy of end-user's response behavior prediction. If DSM price is set too high or too low, there will be a big deviation between $d_t$ and $d_{\text{target},t}$, which will lead to a high penalty.

### 3.2 Virtual environment structure based on LSTM

In DSM scenario, the role of the virtual environment is to simulate the user's response $d_t$ to a certain DSM price $x_t$. Considering that the user's response behavior is often related to the response history in previous periods, and LSTM network is good at excavating the correlation in sequence data[31], several LSTM networks are used for the establishment of virtual environment. The LSTM-based response behavior simulation network is shown in Fig.5. Considering the different response behavior characteristics in different periods, LSTM networks are established for each response period, which is represented as $\mathrm{A}_i\,(i = 1,2,\mathrm{L}\ ,T)$. When the simulation is carried out to the $n$th day, the form of the time period $t$ can be recorded as $t = nT + i$.

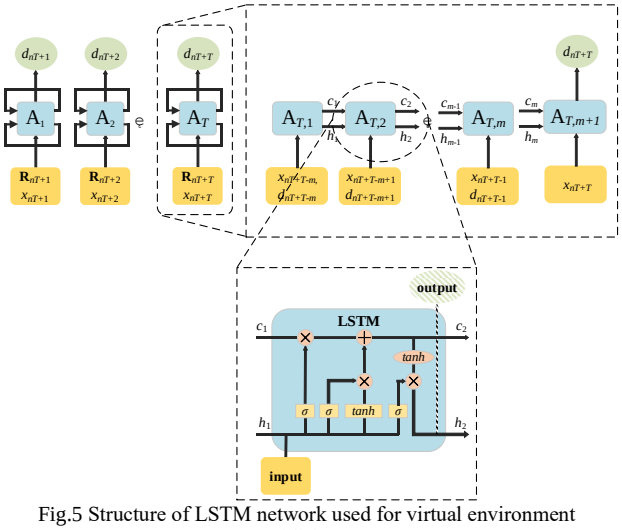



Fig.5 Structure of LSTM network used for virtual environment establishment

Assuming that the user's response is influenced by the response history of the previous $m$ periods, the input data includes not only DSM price $d_{nT+i}$, but also relative information set $\mathbf{R}_{nT+i}$, which is expressed by (8). When simulating users' response, the data pairs in $\mathbf{R}_{nT+i}$ will be sequentially input into LSTM unit, and the influence of each data pair will be transformed into two variables, $c_j$ and $h_j$ ($j \in [1, 2, \ldots, m]$), which will affect the final output of LSTM. In this way, the influence of each previous periods can be taken into account in the simulation. The way to calculate $c_j$ and $h_j$ can be found in Fig.5 and Appendix 1 (*tanh* and $\sigma$ in Fig.5 are activation functions). Besides, $c_j$ and $h_j$ reflect long-term and short-term memory of LSTM, and more detailed information can be found in [32]. Finally, after inputing all data pairs in $\mathbf{R}_{nT+i}$, LSTM outputs the user's response $d_t$ according to $x_t$.

$$\mathbf{R}_t = \left\{ \left(x_{t-m}, d_{t-m}\right), \left(x_{t-m+1}, d_{t-m+1}\right), \mathrm{L}\ , \left(x_{t-1}, d_{t-1}\right) \right\} \tag{8}$$

The LSTM network in Fig.5 is a simplified diagram, which has only one neuron. In actual simulation processes, the number of LSTM unit in each layer can be increased, or hidden layers can be added after each LSTM layer, ensuring the complexity of the network to improve accuracy.

To simplify the training process of LSTM network, the training data needs to be normalized by (9):

$$x_t = \frac{x_t - x_{\min}}{x_{\max} - x_{\min}}, \quad d_t = \frac{d_t - d_{\min}}{d_{\max} - d_{\min}} \tag{9}$$

where, $d_{\min}$ and $d_{\max}$ are the minimum and maximum response in all training data.

The historical data set $\mathbf{H}$ is divided into a training set and a validation set, and be expressed by (10):

$$\mathbf{H} = \left\{ (\mathbf{R}_1, d_1), (\mathbf{R}_2, d_2), \mathrm{L}\ , (\mathbf{R}_{nT-1}, d_{nT-1}), (\mathbf{R}_{nT}, d_{nT}) \right\} \tag{10}$$

Use root mean square error (RSME) [33] as the loss function of the training process, which is expressed by (11):

$$loss_{\text{LSTM}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d - d')^2} \tag{11}$$

where, $d$' represents the simulated user response.

Mean absolute percent error (MAPE) [34] is used to evaluate the simulation accuracy of LSTM, expressed by (12):

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{d - d'}{d}\right| \times 100\% \tag{12}$$

## 4. DSM PRICING METHOD BASED ON A3C ALGORITHM

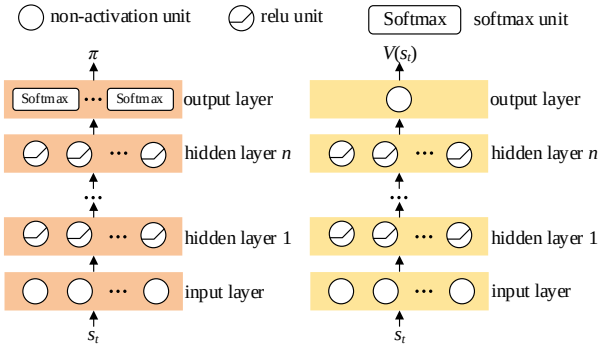
### 4.1 DSM pricing by cloud-side decision center based on decision network

The pricing process in the decision center depends on decision network. Decision network in A3C algorithm is divided into Actor network and Critic network. In DSM scenario, their input and output are shown in Fig.6. The detailed calculation process of these two neural networks can be found in Appendix 1. The current state $s_t$ is the input of Actor network and Critic network. The output of Actor network is pricing policy $\pi$. In DSM scenario, it is the probability distribution of each optional price. The output of Critic network is the reward expectation $V(s_t)$ from current period to the last period of the day, assuming that the current pricing policy $\pi$ keeps being used during the day. $V(s_t)$ can be expressed by (13).

$$V(s_t) = \mathrm{E}\left[ R_t \middle|\ s = s_t \right], \quad R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k} \tag{13}$$

where, $\gamma$ is discount factor, and its range is [0, 1]. $\mathrm{E}[\ ]$ means mathematical expectation. This expectation takes into account short-term reward ($r_t$) and long-term reward ($\gamma r_{t+1}$, $\gamma^2 r_{t+1}$, … , $\gamma^{(T-}$

$^{t)}r_T$), and it is a more comprehensive evaluation method than just using $r_t$. Based on $V(s_t)$, the Critic network can guide the Actor network to adjust the policy $\pi$ to obtain the maximum reward, and this process will be introduced in section 4.2.




(a)Actor network (b) Critic network

Fig.6 Structure of decision network

Critic network can't output accurate $V(s_t)$ without training. Using (13) to calculate the truth value of $V(s_t)$ and adjust parameters of Critic network to make its output close to $V(s_t)$ seems to be a feasible method. But calculating long-term reward ($\gamma r_{t+1}$, $\gamma^2 r_{t+1}$, … , $\gamma^{(T-t)} r_T$) needs many times of simulation, which takes a lot of time. Temporal difference error (TD-error) is another method to evaluate the accuracy of $V(s_t)$. After the distributed network adopting DSM price $x_t$, virtual environment can simulate end-user's response $d_t$, calculate reward $r_t$ by (6), and aquire the next state $s_{t+1}$. Then, input $s_t$ and $s_{t+1}$ into Critic network respectively, and TD-error can be calculated by (14):

$$\delta = r_t + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) \tag{14}$$

where, $\delta$ is TD-error. It can be seen from (13) that the realistic meaning of TD-error is the deviation between Critic network output and the truth value of $V(s_t)$, still reflecting the accuracy of Critic network. Therefore, the training target of Critic network is to make TD-error close to 0.

It can be learnt from (14) that if $\gamma$=0, only when the evaluation value $V(s_t)$ is infinitely close to $r_t$ can TD-error approach 0. It means that the Critic network only focuses on short-term reward and completely ignores the long-term reward, and the algorithm becomes myopic algorithm. Correspondingly, when $\gamma$ keeps increasing, it means that the Critic network pays more attention to long-term reward.

## 4.2 Decision network updating based on edge-side computing devices

In A3C based training process, apart from the decision network in decision center, there are also decision networks in each edge devices, which are called distributed network. The decision network in decision center is called global network. Distributed networks just assist in training global network, and only global network participate practical pricing process in proposed pricing method. The traning of global network is accomplished through the interaction between distributed networks and virtual environments. This calculation process is finished by edge computing devices. The interaction between distributed networks and virtual environments is a simulation that DSM service provider adopt a DSM price and end-users' response to this price, during which the adjustments to global network parameters are calculated and uploaded.

The parameter adjustment sent to the global network are the accumulating gradients of the parameters, which is obtained through TD-error.

The TD-error based Critic network training loss function and parameter adjustment can be expressed by (15) and (16):

$$loss_{\text{critic}} = \delta^2 \tag{15}$$

$$d\boldsymbol{\omega} \leftarrow d\boldsymbol{\omega} + \beta \partial \delta^2 / \partial \boldsymbol{\omega} \tag{16}$$

where, $loss_{\text{critic}}$ is the loss function of Critic network. $\boldsymbol{\omega}$ is parameters of Critic network. $\beta$ is the learning rate. The training target of Critic network is to minimize TD-error.

The TD-error based Actor network training loss function and parameter adjustment can be expressed by (17) and (18):

$$loss_{\text{actor}} = \delta \cdot \log \pi\left(a_t \mid s_t, \boldsymbol{\theta}\right) + c \cdot H\left(\pi\left(s_t, \boldsymbol{\theta}\right)\right) \tag{17}$$

$$d\boldsymbol{\theta} \leftarrow d\boldsymbol{\theta} + \alpha \delta \nabla_\theta \log \pi\left(a_t \mid s_t, \boldsymbol{\theta}\right) + c \nabla_\theta H\left(\pi\left(s_t, \boldsymbol{\theta}\right)\right) \tag{18}$$

where, $loss_{\text{actor}}$ is the loss function of Actor network. $\boldsymbol{\theta}$ is parameters of Actor network. $\alpha$ is the learning rate. $H$ is the entropy of the probability distribution ($\pi$), and $c$ is the coefficient. The first item of the parameter adjustment function can improve the reward of the policy, and the second item can reduce contingency and speed up algorithm convergence. It can be seen from (14) and (18) that $V(s_t)$ plays an important role in Actor network training. If the reward $r_t$ of decision $x_t$ exceeds the expectation of Critic network ($V(s_t)$- $\gamma V(s_{t+1})$), $\delta$ is positive and this decision will be a positive experience to update the Actor network, otherwise, this decision will be a negative experience to update the Actor network.

In A3C algorithm, the accumulate gradients of the parameters are calculated in edge-devices by (16) and (18), but they are only uploaded to decision center and not used to update distributed network. When interacting with virtual environment, parameters in distributed networks are constant, but after the interaction, parameters in distributed network are synchronized with global network regularly.

From the above process, it can be learnt that by setting the value of $\gamma$ in (14), the importance of long-term reward to Critic network in the evaluation process can be adjusted. Actor network can be further guided to make decisions that comprehensively consider short-term and long-term reward, guaranteeing the DSM pricing strategy prospective.

The detailed updating process in one edge-device is shown Algorithm 1:

**Algorithm 1.** Pseudocode of the updating global network in one edge-device in one episode

//Assume global shared parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$
//Assume thread-specific parameter vectors $\boldsymbol{\theta}'$ and $\boldsymbol{\omega}'$.
//Assume maximum DSM period =$T$
A1-1: Initialize DSM period $t$=1
A1-2: Initialize learning rate $\alpha$, $\beta$ and discount factor $\gamma$
A1-3: Reset gradients: $d\boldsymbol{\theta}$=0, $d\boldsymbol{\omega}$=0
A1-4: Get state $s_t$
A1-5: **repeat**
A1-6: Choose a DSM price $x_t$ according to policy $\pi$ generated by Actor network
A1-7: Form $\mathbf{R}_t$ by $x_t$ and historical data, input $\mathbf{R}_t$ to virtual environment, and get end-user's response $d_t$
A1-8: Calculate reward $r_t$ by Eq.(6), and get next state $s_{t+1}$
A1-9: **if** $t$=$T$
A1-10: Calculate state value $V\left(s_t\right)$ of $s_t$ by Critic network, $V\left(s_{t+1}\right)$=0
A1-11: **else**:

A1-12: Calculate state value $V(s_t)$ and $V(s_{t+1})$ of $s_t$ and $s_{t+1}$ by Critic network
A1-13: Calculate TD-error by Eq.(14), and calculate accumulate gradients of Actor network and Critic network by Eq.(18) and Eq.(16)
A1-14: Upload $d\boldsymbol{\theta}$ and $d\boldsymbol{\omega}$ to decision center to update global network
A1-15: $t$=$t$+1;
A1-16: **until** $t$=$T$+1

## 4.3 Distributed training structure of decision network based on cloud-edge interaction

The global network updating process in decision center is shown in Algorithm 2. In each training episode, global network in cloud-side is updated $T$ times by each edge-device, and at the end of the episode, the parameters in distributed networks are synchronized with global network. The cloud-edge training structure based on A3C algorithm is shown in Fig.7. The information interaction between decision center and edge devices are synchronizing parameters, first three dimensions of st and accumulate gradients. Multiple edge devices can realize the multi-threaded update of global network, and the optimization ability can be greatly increased. In the above information interaction process, user data is only used for calculating parameter adjustment and isn't uploaded to the decision center. The information interaction between cloud-side and edge-side is changed from user data to decision network parameters. This structure can greatly relieve calculation pressure in cloud-side and communication pressure between cloud-side and edge-side.

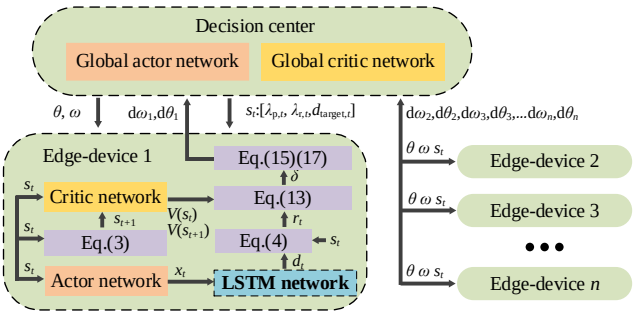



Fig.7 Cloud-edge training structure of decision network based on A3C algorithm

**Algorithm 2.** Pseudocode of updating global network in decision center based on cloud-edge interaction

//Assume global shared parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$
//Assume thread-specific parameter vectors $\boldsymbol{\theta}'$and $\boldsymbol{\omega}'$.
//Assume synchronization interval = 1 episode (1 day)
//Assume maximum train episode = $step_{\max}$, and maximum DSM period =$T$
A2-1: Initialize training episode $step$=1
A2-2: **repeat**
A2-3: Initialize DSM period $t$=1
A2-4: **repeat**
A2-5: Send $d_{\text{target},t}, \lambda_{r,t}, \lambda_{p,t}$ to edge-side
A2-6: Get $d\theta$ and $d\omega$ from edge-devices and update parameters in global network.
A2-7: $t$=$t$+1
A2-8: **utill** $t$=$T$+1
A2-9: Synchronize parameters in distributed network: $\boldsymbol{\theta}'$=$\boldsymbol{\theta}$, $\boldsymbol{\omega}'$=$\boldsymbol{\omega}$
A2-10: $step$ = $step$ +1;
A2-11: **until** $step$ = $step_{\max}$ +1

## 4.4 Overall DSM pricing process under cloud-side environment

The proposed DSM pricing process is shown in Fig.8. The main steps of the proposed method include establishment of virtual environment, implementation of A3C based DSM pricing and testing in real environment. The edge computing device first trains LSTM network based on historical data set. After establishing virtual environment, edge computing devices keep updating the global network in decision center. Finally, the completed global network is used for DSM pricing in real environment, and edge devices collect user response information, test optimality of virtual environment and decision network, and update the historical data set for further optimization of the algorithm. The above algorithm changes the original distributed learning and distributed decision-making structure of A3C algorithm to a distributed learning and centralized decision-making structure, which is more appropriate in DSM scenarios. Most of calculating process is completed by edge devices, sharing the calculation pressure of the decision center. The decision center is responsible for integrating pricing experience obtained from edge devices during training process, and making decisions and integrating historical data after training process.

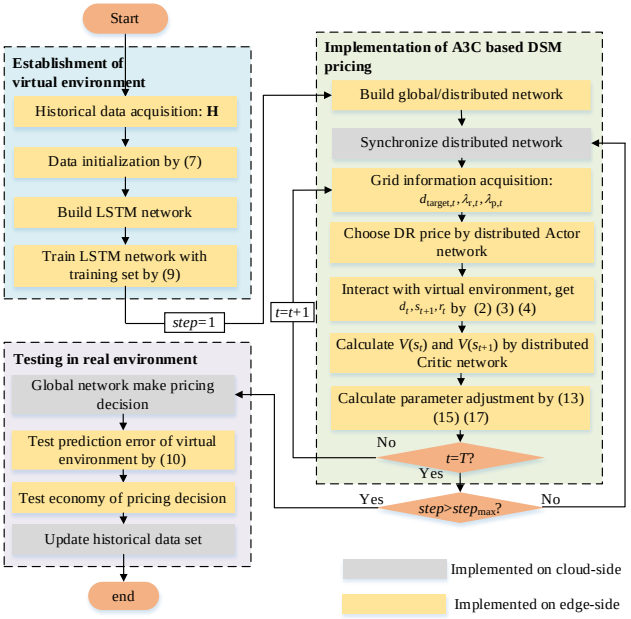



Fig.8 Proposed DSM pricing process for cloud-edge environment

The above-mentioned DSM pricing process can be used in two kinds of DSM cases: day-ahead DSM and real-time DSM.

Day-ahead DSM is used for day-ahead dispatching. The application flow of day-ahead DSM is shown in Fig.9. Abbreviations like A2-1 in Fig.9 are steps in Algorithm 1 or Algorithm 2. Before the start of the day, there is a decision network training period to get the completed decision network for DSM pricing of the day. Firstly, the utility company makes day-ahead load prediction, determines load regulation targets, compensation price and penalty price in each DSM period and sends them to service provider. Then, service provider finishes decision network training for the day. Especially, the information sent to edge-side in Algorithm 2 line A2-5 is just one day's information. When DSM begins, service provider

sends DSM price to end-users, and the responses of the end-users to DSM prices are recorded in historical data set. In day-ahead DSM, the DSM pricing process in Fig.8 is implemented once a day.

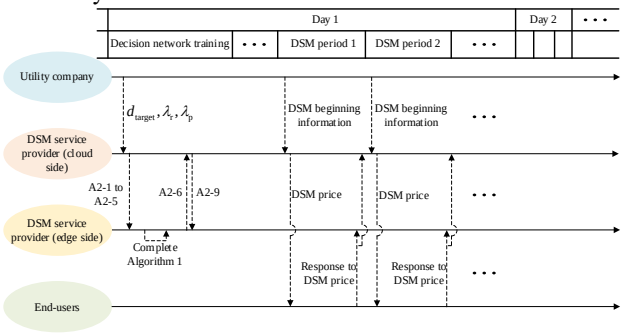



Fig.9 Application flow of day-ahead DSM

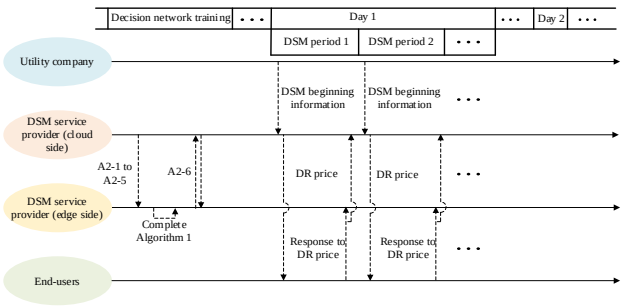



Fig.10 Application flow of real-time DSM

Real-time DSM is used for emergency situations such as sudden decrease in power supply or grid failure, and only the number of DSM periods is predetermined. The application flow of real-time DSM is shown in Fig.10. Abbreviations like A2-1 in Fig.10 are steps in Algorithm 1 or Algorithm 2. In real-time DSM, service provider can not get load regulation targets from utility company until real-time DSM is nearly implemented, and there isn't plenty time for decision network training. In this case, decision network should be trained in advance and suitable for DSM pricing in random days, and the information sent to edge-side in algorithm 2 line A2-5 isn't from utility company, but is sampled randomly in training process. After DSM, the responses of the end-users to DSM prices are recorded in historical data set and will be used in next training. In real-time DSM, the DSM pricing process in Fig.8 is implemented once every long time.

## 5. Case Study

### 5.1 Test of LSTM based virtual environment

In this case study, each LSTM network contains two layers. The first layer is an LSTM layer with 20 LSTM units. The second layer is a Dense layer with 1 unit. Improved price elasticity model is used for sample data acquisition. The improved price elasticity model is expressed by (19), (20) and (21). In order to ensure the diversity of users, 1000 end-users are simulated. The behavior parameters of the end-users are randomly sampled by normal distribution, and the parameter range is shown in Table 2. More detailed information of the elasticity model is shown in [35]. Daily DSM period number $T$=5, which means 5 LSTM networks are established.

$$d_t = b_t + k_t x_t + D_{c,t} + D_{s,t} \tag{19}$$

$$D_{c,t} = \sum_{j=1}^{t-1} (d_{t-j} + \varepsilon_{t,t-j,c}(x_t - x_{t-j}) d_{t-j} / x_{t-j}) \tag{20}$$

$$D_{s,t} = \sum_{n=1}^{re} (d_{t-nT} + \varepsilon_{t,t-nT,s}(x_t - x_{t-nT}) d_{t-nT} / x_{t-nT}) \tag{21}$$

where, $b_t$ and $k_t$ are elasticity parameters. $D_{c,t}$ reflects the impact of historical data in previous periods of the day on user behavior. $D_{s,t}$ reflects the impact of historical data in the same periods of previous few days on user behavior. $re$ is the number of related days. $\varepsilon_{t,t-j,c}$ and $\varepsilon_{t,t-nT,s}$ are parameters to calculate $D_{c,t}$ and $D_{s,t}$.

Table 2 Parameter ranges of price elasticity model

| $b_t$ | $k_t$ | $\varepsilon_{t,t-j,c}$ | $\varepsilon_{t,t-nT,s}$ |
|---|---|---|---|
| [0, 0.1] | [0.04, 0.20] | [0.01, 0.04] | [0.03, 0.08] |

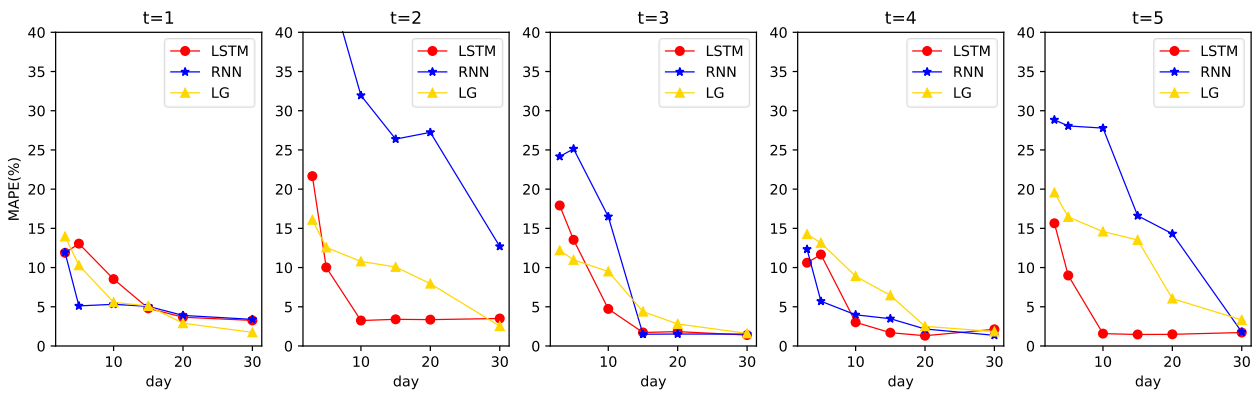



Fig.11 MAPE of virtual environments in each DSM period

The relevant parameters of the training are shown in Table 3. Because daily DSM period number $T$=5, Five LSTM networks are established, corresponding to the simulation of user behavior in five DSM periods respectively. Each LSTM network is constituted by a LSTM layer, containing 20 LSTM units, and an output layer, containing one Dense layer. Each LSTM network is trained up to 5000 times, but if the loss function does not change for 400 consecutive times, the training will stop in advance to prevent overfitting.

Apart from LSTM, standard RNN and linear regression based virtual environments are also established. For each DSM period (each LSTM, RNN or linear regression model), there are

100 historical data for model test. MAPE of each virtual environment based on different amounts of historical data is shown in Fig.11. Based on 15 days of historical data, MAPE of LSTM for different DSM periods can be lower than 5%, which means the simulation accuracy is more than 95%. But RNN and linear regression obviously need more days of historical data to achieve a high accuracy. This shows that LSTM has the lowest demand for historical data among these methods, which is especially critical when DSM implementation experience is insufficient.

Simulation accuracy of LSTM based on 50 days of historical data for each DSM period is shown in Table 4. The accuracy of the virtual environment can reach more than 96%, which means it capable of replacing actual environment to interact with A3C algorithm.

Table 3 Parameter settings of LSTM training

| Paramenters | settings | Paramenters | settings |
|---|---|---|---|
| LSTM units | 20 | Maximum epochs | 5000 |
| Optimizer | Adam | | |
| Learning rate | 0.0001 | Patience | 400 |
| Loss function | Eq. (11) | Bacth size | All data |

Table 4 Simulation accuracy of LSTM based on 50 days of historical data

| $T$ (h) | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Accuracy (%) | 96.815 | 97.058 | 98.818 | 98.826 | 98.695 |

## 5.2 Test of A3C based DSM

Case studies are implemented under two cases:

Case 1: Day-ahead DSM. The service provider can acquire next day's load regulation targets, and there is abundant time for training process.

Case 2: Real-time DSM. The service provider is unable to obtain load adjustment targets in advance, and need to make quick decisions.

### *5.2.1 Structure of decision network and A3C parameter settings*

The decision ranges of DSM price in Case 1 and Case 2 are [0.6, 1] CNY and [6, 10] CNY. Discrete the decision range into 40 values at intervals of 0.01 and 0.1, respectively. The other detailed information of Actor and Critic network is inTable 5. The global network is updated by 4 edge devices. The training optimizer is Adam [21]. The other detailed A3C algorithm information is in Table 6

Table 5 Parameter settings of Actor and Critic network

| Paramenters | Actor network | Critic network |
|---|---|---|
| Input | 9-dimension State | |
| Output | action probability | state value |
| Layer units | 100, 100, 40 | 100, 50, 1 |
| Activation | relu, relu, softmax | relu, relu, None |

Table 6 Parameter settings of A3C algorithm

| Paramenters | Settings |
|---|---|
| Optimizer | Adam |
| Learing rate | $\alpha$=0.0005, $\beta$=0.005 |
| Discount factor | $\gamma$=0.99 |
| Entropy factor | $c$=0.01 |
| Thread number | 4 |
| Update interval | 1 hour |
| Synchronization interval | 5 hours |

### *5.2.2 Case 1: Day-ahead DSM*

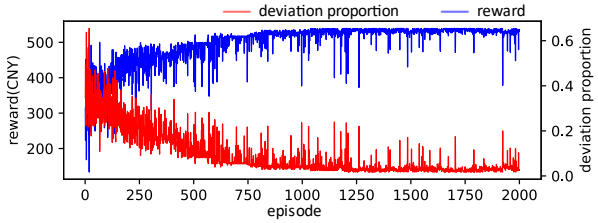
The training process in Fig.8 can be implemented every day in case 1, because of abundant decision time. Therefore, the decision network can be trained only for one day, and during day-ahead pricing process, the decision network for the next day is trained. In this case, the grid information acquisition process in Fig.8 input the next day's $d_{\text{target},t}$ $\lambda_{\text{r},t}$ and $\lambda_{\text{p},t}$ into decision network. The virtual environment used in training is the one established in section 5.2.1.

The decision network in this case study is trained for DSM pricing of a standard day. The $d_{\text{target},t}$ and $\lambda_{\text{r},t}$ of the day is shown in Table 7. $\lambda_{\text{p},t}$=0.75*$\lambda_{\text{r},t}$.

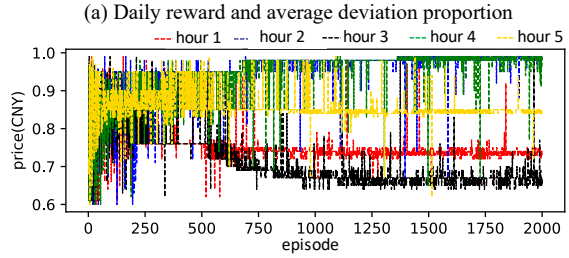
Table 7 DSM information of standard day

| $t$ (h) | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $d_{\text{target},t}$ (kW) | 228.2 | 205.3 | 207.1 | 241.5 | 200.8 |
| $\lambda_{\text{r},t}$ (CNY) | 1.33 | 1.42 | 1.17 | 1.31 | 1.54 |

The training process based on A3C algorithm is shown in Fig.12. (a) is the change process of the daily reward and average deviation proportion through out the day in training process. (b) is the change process of DSM price in each DSM period. As the training progresses, the deviation between user's response and load regulation target gradually approaches 0, and the daily reward obtained within a day also gradually approaches the optimal value, reflecting the algorithm's high decision ability. In addition, the daily reward and deviation proportion do not completely converge to the optimal value during the training process. The reason is that the DSM prices is sampled according to the probability distribution $\pi$, and there is no guarantee that the best price be always obtained. The price fluctuation in Fig.12 (b) can also prove this. The DSM price in the training process fluctuates in a small range near the optimal price, and sometimes are far away from the optimal price. This mechanism is slow down the convergence of the algorithm, but can promote the algorithm's exploration of the decision space and prevent convergence to the local optimal solution.




(a) Daily reward and average deviation proportion




(b) DSM price in each DSM period

Fig.12 A3C based training process in Case 1

This paper further compares the centralized Actor-Critic algorithm and Q-learning algorithm with the proposed

algorithm. The comparison of the daily reward during the training process is shown in Fig.13. The training speeds of the three algorithms are similar, and they all reach convergence after about 750 trainings, but the optimization capabilities of the three algorithms have a certain gap. The maximum daily reward obtained by A3C algorithm is 538 CNY, but the maximum daily reward of Actor-Critic algorithm and Q-learning algorithm both stop increasing after reaching about 500 CNY. Actor-Critic algorithm did not jump out of the local optimal solution until 1800 trainings. From the above analysis, A3C algorithm has better exploration capabilities, comparing with centralized RL algorithm. It should be noted that the information transmitted between cloud-side and edge-side under the proposed method is neural network parameters rather than end-users' energy consumption information, which can effectively relieve communication pressure. This is another advantage of the proposed method.

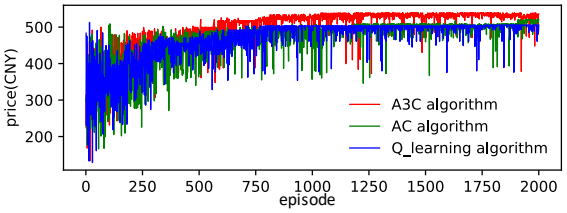



Fig.13 Comparison between A3C and centralized RL algorithm

### 5.2.3 Case 2: Real-time DSM

In the real-time DSM scenario, because DSM information such as load regulation targets cannot be obtained in advance, it is necessary to train a decision network that can deal with various emergencies in advance. This decision network is used in each real-time DSM, and the decision network training process shown in Fig.8 is only performed once. In this case, the grid information acquisition process in Fig.8 input random $d_{\text{target},t}$ $\lambda_{\text{r},t}$ and $\lambda_{\text{p},t}$ into decision network. The range of $d_{\text{target},t}$ and $\lambda_{\text{r},t}$ are [180, 250] kW and [10, 16] CNY. $\lambda_{\text{p},t}$=0.75*$\lambda_{\text{r},t}$. Considering the low response willingness of end-users in this case, parameters $k_{i,t}$, $\varepsilon_{i,t\text{-}i}$ and $\varepsilon_{i,t\text{-}s}$ in Table 2 are decreased by 10 times when training LSTM network.

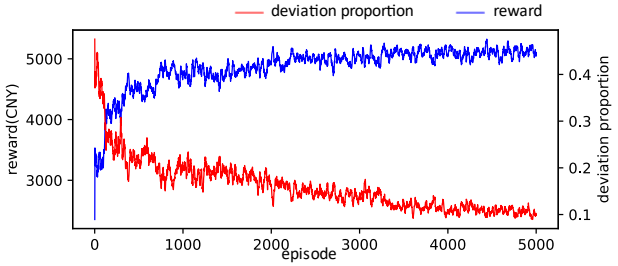



Fig.14 A3C based training process in Case 2

Fig.14 is the change process of the daily reward and average deviation proportion through out the day in training. Since the situation of each episode is different, the maximum daily reward in each period is also different, and the daily reward and deviation proportion during the training process always fluctuate. It can be seen from Fig.14 that, with the training process, the daily reward of service provider gradually stabilized around 5000 CNY, and the deviation proportion also decreased to less than 0.1. For many DSM periods, the reason why the deviation proportion cannot approach 0 is that the DSM price reaches the upper or lower limit of the decision range. There is also a certain probability that the optimal DSM price is not obtained in the training process, since DSM prices is sampled according to the probability distribution $\pi$.

After the training, the decision network trained by the algorithm is tested in 7-day demand response process in actual environment. The DSM program is implemented in the summer high temperature period, that is, from 12:00 to 16:00 every day, and the DSM price and actual response in 35 periods during 7 days are shown in Fig.15. The deviation between the user's actual response and the regulation target during most of the test periods is always small, reflecting a good decision ability of the algorithm. To be more intuitive, the load curve changes before and after DSM are shown in figures in Appendix 2.

Furthermore, the virtual environment in cas 1 is used in the training process of case 2 to compare the decision abilities of plan-based decision network and real-time decision network. $d_{\text{target},t}$ $\lambda_{\text{r},t}$ and $\lambda_{\text{p},t}$ are also generated randomly. The decision network obtained after 5000 episodes' training is tested on the standard day mentioned in Table 7. The results are compared with the optimal results in case 1. The comparison is shown in Fig.16. When using plan-based decision network, the user's response is closer to load regulation target, meaning that plan-based decision network has better decision ability than real-time decision network. Therefore, separate training for each day can improve the reward of DSM service provider.

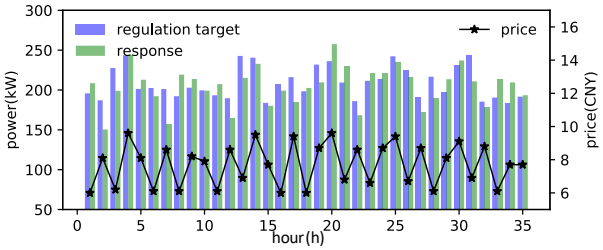



Fig.15 Algorithm performance in real environment

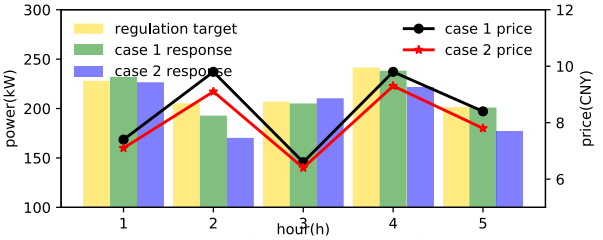



Fig.16 DSM results under Case 1 and Case 2

The influence of discount factor $\gamma$ is also analyzed. Training processes for real-time DSM with discount factor $\gamma$ varying from 0.05 to 0.99 are implemented. And the trained decision networks are tested in 10-day demand response process in actual environment, and the average daily reward with different $\gamma$ is shown in Table 8. With the reduction of $\gamma$, daily reward of the algorithm is also reduced. To further analyze the internal mechanism of this situation, the average DSM prices and average end-user responses of these 10 days in different DSM periods are compared in Fig.17 and Fig.18. It can be seen from Fig.17 that when $\gamma$ is close to 1, DSM prices in the first three periods are lower, such as hour 1, hour 2 and hour 3. That is because end-users' response in early periods will reduce their

willingness to respond in later periods. This point can be confirmed when combining Fig.17 and Fig.18: in the last DSM period of the day (hour 5), DSM prices obtained by cases with different $\gamma$ are very close, but there is a large gap in the response volume of end-users. To sum up, appropriately reducing the response volume in the first few DSM periods will improve users' willingness to response, which is conducive to improving the reward in subsequent DSM periods. However, when $\gamma$ is close to 0, the algorithm pays more attention to current reward and does not consider the impact of the current decision on subsequent DSM periods. The above results show that, by setting discount factor $\gamma$ close to 1, the proposed method described can remain forward-looking in real-time DSM pricing, that is, it can make decisions to reduce current reward to increase future reward, and has a high DSM pricing ability.

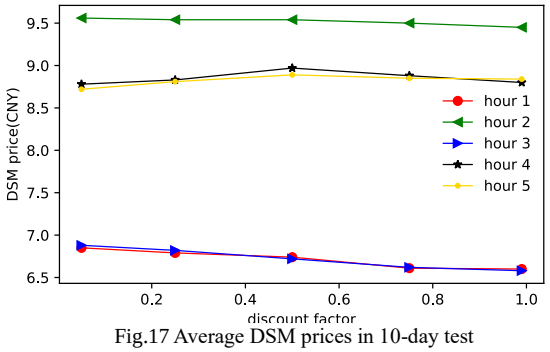



Fig.17 Average DSM prices in 10-day test

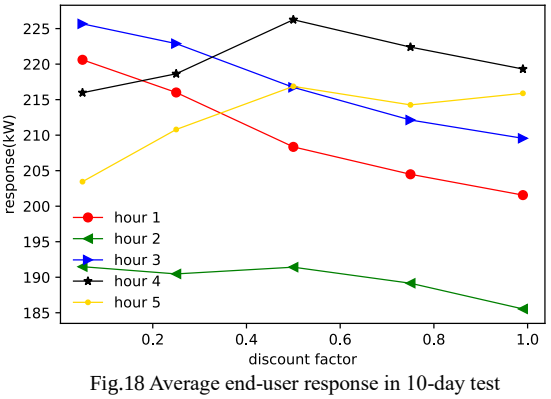



Fig.18 Average end-user response in 10-day test

Table 8 Average daily reward of decision networks with different $\gamma$

| $\gamma$ | 0.99 | 0.75 | 0.5 | 0.25 | 0.05 |
|---|---|---|---|---|---|
| Reward (CNY) | 5366.1 | 5293.7 | 5208.8 | 5157.0 | 5106.2 |

### 5.3 Run time analysis

The case study is preformed using TensorFlow 20.3.3 library in Python 3.7 on a PC configured with 8G RAM and Intel(R) Core (TM) i5-6400 CPU, 2.70 GHz, 64-bit operating system. The program is run several times. The average training time for LSTM network is 4.25min. The average training time for A3C algorithm in day-ahead DSM is 11.02min, and training time for A3C algorithm in real-time DSM is 31.35min. The training time of the A3C algorithm is too long for real-time DSM, but applicable to day-ahead DSM pricing. Decision speed of decision network is also tested, and the time to conduct 50 times of DSM pricing is 2.51s. It can be seen that if the training of decision network is completed in advance, the speed of pricing through existing decision network is very fast, which is capable to conduct real-time DSM pricing. Moreover, computers in service provider are much better, and less decision time will be spent. Journal of Cleaner Production

## 6. Conclusion

Considering the cloud-edge operation structure of the power system, this paper proposed a DSM pricing method based on A3C and LSTM under cloud-edge Environment. Through theoretical analysis and case studies, the following conclusions are obtained.

(1) The LSTM network has high accuracy in the simulation of end-users' response behavior, and can be used as virtual environment for the preliminary exploration of RL. Training based on 15 days of data can achieve an accuracy of more than 95%

(2) The distributed update process of A3C algorithm is not only structurally suitable for the cloud-edge environment, but also has a stronger optimization capability than centralized algorithm. The reward of the proposed algorithm is more than 5% higher than traditional RL algorithms.

(3) The proposed DSM pricing algorithm can be used for day-ahead DSM and real-time DSM. In Day-ahead DSM, due to the stronger pertinence of training, its pricing decision has a higher profit than in real-time DSM. Therefore, when the decision-making time is sufficient, the application flow of day-ahead DSM should be preferred.

(4) The algorithm described in this paper can realize on-spot utilization of demand side information and relieve communication and calculation pressure of the system. It can provide decision-making guidance for demand-side managers under cloud-edge environment.

## Acknowledgment

This work was Supported by Open Fund of Beijing Key Laboratory of Demand Side Multi-Energy Carriers Optimization and Interaction Technique (China Electric Power Research Institute) (YDB51202001977).

## Appendix 1

Each layer of a neural network is composed by several units, for normal neural network, each unit is an equation set which can be expressed by

$$h_j^l = \sum_{i=1}^{N_{k-1}} g\left(k_{ij} h_j^{l-1} + b_j^{l-1}\right) \tag{22}$$

Where, $g(\cdot)$is the activation function. $h_j^l$ and $h_j^{l-1}$ are the output of the $j$th unit in the $l$th and ($l$-1)th layer. $N_{l-1}$ is the number of the units in the ($l$-1)th layer. $k$ and $b$ are the weight and the bias, and it they are the parameters to be adjusted in the training process. In this paper, the parameters in Critic network are represented by $\boldsymbol{\omega}$, and the parameters in Actor network are represented by $\boldsymbol{\theta}$.

For more complex neural networks such as LSTM, the output of a unit is a combination of several activation functions. Each activation function in a LSTM unit has different weight and bias, and the combination method is shown in Fig.5. More detailed information about LSTM can be found in [27].

Different kinds of unit have different activation functions. In LSTM tanh function and sigmoid function (represented by $\sigma(\cdot)$) are used, which can be expressed as follows:

$$\text{tanh:}\quad f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$
$$\text{sigmoid:}\quad f(x)=\frac{1}{1+e^{-x}} \tag{23}$$

In Actor network and Critic network, relu function and softmax function are used, which can be expressed as follows:

$$\text{relu:}\quad f(x)=\begin{cases}0 & \text{if } x<0\\ x & \text{if } x>0\end{cases}$$
$$\text{softmax:}\quad f(x)=\left[\frac{e^{x_1}}{\sum_{c=1}^{C}e^{x_c}},\frac{e^{x_2}}{\sum_{c=1}^{C}e^{x_c}},\ldots,\frac{e^{x_C}}{\sum_{c=1}^{C}e^{x_c}}\right]^{\mathrm{T}} \tag{24}$$

Relu is easy to understand, and for softmax, its output is a $C$-dimensional vector. $C$ is the number of the choices, and each term in the vector is the probability of making that choice. For example, in this paper, the decision space is divided in to 40 choices, therefore, $C$=40, and each term is the probability of choosing that price.

## APPENDIX 2

The load curve changes before and after DSM in these 7 days are shown as follows.

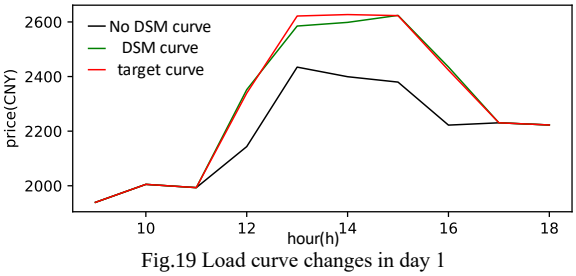


Fig.19 Load curve changes in day 1

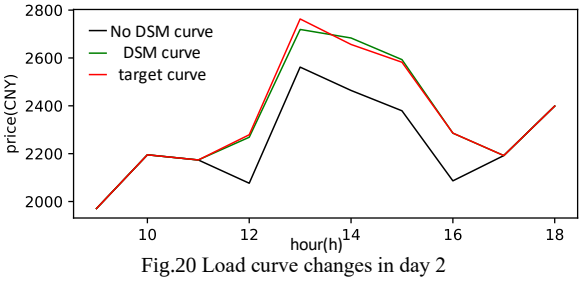


Fig.20 Load curve changes in day 2

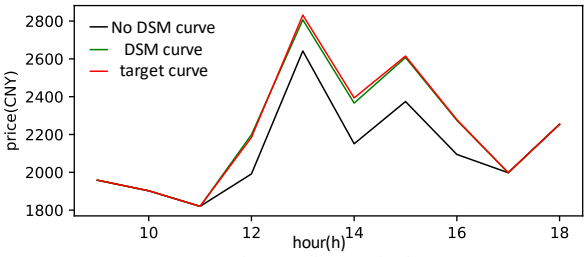


Fig.21 Load curve changes in day 3

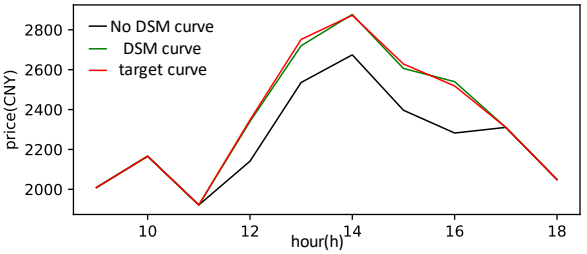


Fig.22 Load curve changes in day 4

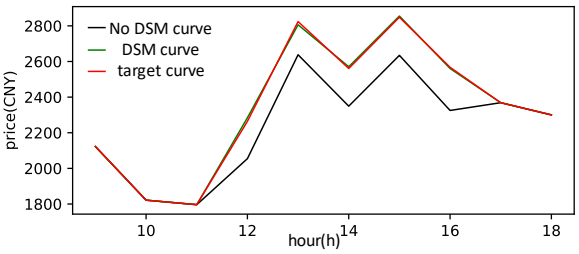


Fig.23 Load curve changes in day 5

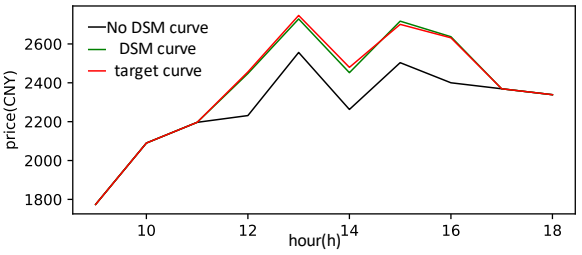


Fig.24 Load curve changes in day 6

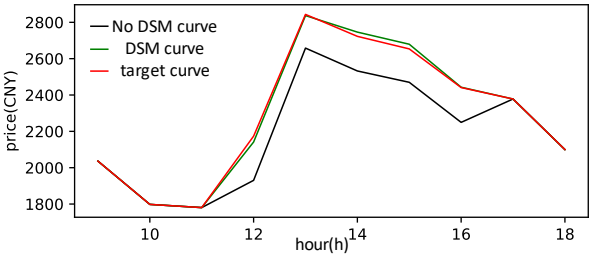


Fig.25 Load curve changes in day 7

## REFERENCES

[1] Fernandez-Blanco, R., Morales, JM, Pineda, S, " Forecasting the price-response of a pool of buildings via homothetic inverse optimization," *Applied Energy*, vol. 290, no. 15, pp. 116791, May 2021, doi: 10.1016/j.apenergy.2021.116791.

[2] Heydar Chamandoust, Ghasem Derakhshan, Seyed Mehdi Hakimi, and Salah Bahramarab 'Tri-objective optimal scheduling of smart energy hub system with schedulable loads', *Journal of Cleaner Production*, vol. 236, no. 117584, Nov. 2019

[3] X. Kong, F. Sun, X. Huo, X. Li and Y. Shen. "Hierarchical Optimal Scheduling Method of Heat-Electricity Integrated Energy System Based on Power Internet of Things." Energy, vol. 210, pp: 118590, 2020.

[4] M. H. Yaghmaee Moghaddam and A. Leon-Garcia, "A fog-based internet of energy architecture for transactive energy management systems," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1055-1069, April 2018, doi: 10.1109/JIOT.2018.2805899.

[5] Yang, T., Zhao, Y., Pen, H., and Wang, Z., "Data center holistic demand response algorithm to smooth microgrid tie-line power

fluctuation," *Applied Energy*, vol. 231, pp. 277-287, Dec. 2018, doi: 10.1016/j.apenergy.2018.09.093

[6] A. Jiang, H. Wei, J. Deng and H. Qin, "Cloud-Edge Cooperative Model and Closed-Loop Control Strategy for the Price Response of Large-Scale Air Conditioners Considering Data Packet Dropouts," *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4201-4211, Sept. 2020, doi: 10.1109/TSG.2020.2985741.

[7] Belli G., Giordano, A., Mastroianni, C., et al., "A Unified Model for the Optimal Management of Electrical and Thermal Equipment of a Prosumer in a DR Environment," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 1791-1800, March 2019, doi: 10.1109/TSG.2017.2778021.

[8] Dutra M., and N Alguacil. "Optimal Residential Users Coordination Via Demand Response: An Exact Distributed Framework." *Applied Energy* vol. 279, pp. 116701, May 2021, doi: 10.1016/j.apenergy.2021.116701.

[9] Heydar Chamandoust, Ghasem Derakhshan, SMH. A., and Salah Bahramarab "Tri-objective scheduling of residential smart electrical distribution grids with optimal joint of responsive loads with renewable energy sources." *Journal of Energy Storage*, vol.27, no.101112, Feb. 2020.

[10] Ruan, G, Zhong, H, Wang, J., Xia, Q, Kang, C., " Neural-network-based Lagrange multiplier selection for distributed demand response in smart grid," *Applied Energy*, vol. 264, pp. 114636, Apr. 2020, doi: 10.1016/j.apenergy.2020.114636.

[11] Bhatti, BA., Broadwater, R., "Energy trading in the distribution system using a non-model based game theoretic approach" *Applied Energy*, vol. 253, pp. 113532, NOV. 2019. Doi: 10.1016/j.apenergy.2019.113532

[12] K. Dehghanpour, M. H. Nehrir, J. W. Sheppard and N. C. Kelly, "Agent-Based Modeling of Retail Electrical Energy Markets With Demand Response," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3465-3475, July 2018, doi: 10.1109/TSG.2016.2631453.

[13] Y. Liang, L. He, X. Cao and Z. Shen. "Stochastic Control for Smart Grid Users With Flexible Demand," *IEEE Transactions on Smart Grid*, vol. 4, no. 4, pp. 2296-2308, Dec. 2013, doi: 10.1109/TSG.2013.2263201.

[14] Li Y., "Deep Reinforcement Learning", arXiv e-prints, 2018.

[15] Vazquez-Canteli, J. R. , and Z. Nagy . "Reinforcement learning for demand response: A review of algorithms and modeling techniques." *Applied Energy,* vol. 235, pp. 1072-1089, Feb. 2019, doi: 10.1016/j.apenergy.2018.11.002.

[16] B. Kim, Y. Zhang, M. van der Schaar and J. Lee, "Dynamic Pricing and Energy Consumption Scheduling with Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2187-2198, Sept. 2016. doi: 10.1109/TSG.2015.2495145.

[17] R. Lu, S. H. Hong and X. Zhang. "A Dynamic pricing demand response algorithm for smart grid: Reinforcement learning approach." *Applied Energy* vol. 220, pp:220-230, 2018.

[18] H. Xu, H. Sun, D. Nikovski, S. Kitamura, K. Mori and H. Hashimoto, "Deep Reinforcement Learning for Joint Bidding and Pricing of Load Serving Entity," *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6366-6375, Nov. 2019, doi: 10.1109/TSG.2019.2903756.

[19] S. Bahrami, Y. C. Chen and V. W. S. Wong, "Deep Reinforcement Learning for Demand Response in Distribution Networks," *IEEE Transactions on Smart Grid*, doi: 10.1109/TSG.2020.3037066.

[20] X. Zhang, D. Biagioni, M. Cai, P. Graf and S. Rahman, "An Edge-Cloud Integrated Solution for Buildings Demand Response Using Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 420-431, Jan. 2021. doi: 10.1109/TSG.2020.3014055.

[21] X. Kong, D. Kong, J. Yao, L. Bai and J. Xiao. "Online pricing of demand response based on long short-term memory and reinforcement learning." *Applied Energy*, vol. 271, pp: 114945, 2020.

[22] Zhe Wang, Tianzhen Hong. "Reinforcement learning for building controls: The opportunities and challenges" *Applied Energy*, vol. 269, pp. 115036, MAY. 2020. Doi: 10.1016/j.apenergy.2020.115036

[23] B. Wang, Y. Li, W. Ming and S. Wang, "Deep Reinforcement Learning Method for Demand Response Management of Interruptible Load," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3146-3155, July 2020, doi: 10.1109/TSG.2020.2967430.

[24] H. Li, Z. Wan and H. He, "Real-Time Residential Demand Response," in IEEE Transactions on Smart Grid, vol. 11, no. 5, pp. 4144-4154, Sept. 2020, doi: 10.1109/TSG.2020.2978061.

[25] Y. Du and F. Li, "Intelligent Multi-Microgrid Energy Management Based on Deep Neural Network and Model-Free Reinforcement Learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1066-1076, March 2020, doi: 10.1109/TSG.2019.2930299.

[26] Lork, C et al. "An uncertainty-aware deep reinforcement learning framework for residential air conditioning energy management." *Applied Energy*, vol. 276, no. 115426, Oct. 2020. doi: 10.1016/j.apenergy.2020.115426

[27] S. Hochreiter and J. Schmidhuber, ``Long short-term memory," *Neural Comput*, vol. 9, no. 8, pp. 1735-1780, 1997, doi: 10.1162/neco.1997.9.8.1735

[28] Yi, W., Gan, D., Sun, M., Ning, Z., and Lu, Z., "Probabilistic Individual Load Forecasting Using Pinball Loss Guided LSTM." *Applied Energy,* vol. 235, pp. 10-20, Feb. 2019. doi: 10.1016/j.apenergy.2018.10.078

[29] Rui Feng; Dariusz Czarkowski. Regularised dynamic optimal transportation of electric vehicles over networks considering strategic charging pricing, IET Energy Systems Integration, 2021, 3(1): 73–85, DOI: 10.1049/esi2.12005.

[30] R. Lu, R. Bai, Y. Huang, Y. Li, J. Jiang, and Y. Ding, "Data-driven real-time price-based demand response for industrial facilities energy management," *Applied Energy*, vol. 283, pp. 116219, Feb. 2021, doi: 10.1016/j.apenergy.2020.116291.

[31] C. P. Mediwaththe, E. R. Stephens, D. B. Smith and A. Mahanti, "A Dynamic Game for Electricity Load Management in Neighborhood Area Networks," in *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1329-1336, May 2016, doi: 10.1109/TSG.2015.2438892.

[32] FA. Gers, J. Schmidhuber, F. Cummins, "Learning to forget: Continual prediction with LSTM", *Neural Computation*, vol. 12, no. 10, pp: 2451-2471, Oct. 2000. DOI: 10.1162/089976600300015015

[33] Zhe Wang, et al."Building thermal load prediction through shallow machine learning and deep learning." *Applied Energy*, vol. 263, pp: 114683, Apr. 2020. doi: 10.1016/j.apenergy.2020.114683.

[34] G. Hafeez, KS. Alimgeer, I. Khan, "Electric load forecasting based on deep learning and optimized by heuristic algorithm in smart grid", *Applied Energy*, vol. 269, pp: 114915, Jul. 2020. Doi: 10.1016/j.apenergy.2020.114915.

[35] K. Khezeli and E. Bitar, "Risk-Sensitive Learning and Pricing for Demand Response," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 6000-6007, Nov. 2018. doi: 10.1109/TSG.2017.2700458.