Intelligent Multi-microgrid Energy Management based on Deep Neural Network and Model-free Reinforcement Learning

Yan Du, *Student Member, IEEE* and Fangxing Li, *Fellow, IEEE*

***Abstract*—In this paper, an intelligent multi-microgrid (MMG) energy management method is proposed based on deep neural network (DNN) and model-free reinforcement learning techniques. In the studied problem, multiple microgrids are connected to a main distribution system and they purchase power from the distribution system to maintain local consumption. From the perspective of the distribution system operator (DSO), the target is to decrease the demand-side peak-to-average ratio (PAR), and to maximize the profit from selling energy. To protect user privacy, DSO learns the multi-microgrid response by implementing a deep neural network (DNN) without direct access to user's information. Further, the DSO selects its retail pricing strategy via a Monte Carlo method from reinforcement learning, which optimizes the decision based on prediction. The simulation results from the proposed data-driven deep learning method, as well as comparisons with conventional model-based methods, substantiate the effectiveness of the proposed approach in solving power system problems with partial or uncertain information.**

***Keywords*—Deep neural network (DNN), Monte Carlo method, multi-microgrid, reinforcement learning, peak-to-average ratio (PAR).**

I. Introduction

The latest advancement of deep learning has opened the door of new AI-driven approaches to solve a broad range of power system problems [1]. Demand-side resource management is one of such problems. In recent years, emerging demand-side resources are playing an increasingly important role in maintaining the economy and security of bulk power system operation [2]-[4]. Many existing research works have been dedicated to exploring the function of multifarious demand-side resources, e.g., distributed generators, plug-in electric vehicles, demand response programs, and microgrids, in providing energy and ancillary services to the utility grid in both normal and emergent status [5]. Compared with conventional stand-by units, the demand-side resources hold the merit of high flexibility because they are free from ramping constraints. Their diversity in type adds additional reliability for serving as alternative power and frequency support to the bulk power system in case of contingency.

The increasing penetration of demand-side resources into the power system calls for demand-side energy management, which aims to enable a coordinated and mutually beneficial interaction between the main grid and the local resources. One of the primary goals of demand-side management is to reduce the peak-to-average ratio (PAR) of the load. A low PAR indicates a smooth load profile, which avoids overloading or underloading the system. Local consumers also benefit from a low PAR by shifting their energy consumption to off-peak hours with low prices.

This work was supported in part by the NSF Award ECCS-1809458 and in part by CURENT, a NSF/DOE Engineering Research Center under the NSF Award EEC-1041877.

Y. Du and F. Li are with Department of EECS, The University of Tennessee, Knoxville. TN 37996, USA (Emails: ydu15@vols.utk.edu; fli6@utk.edu). *Corresponding author: F. Li.*

There have been substantial efforts to investigate the optimal scheduling of demand-side resources in the literature. The concept of autonomous demand-side management is first introduced in [6], in which a non-cooperative game is formulated between the utility company and local customers. Iteratively, the utility provides dynamic pricing signals according to the aggregated consumer response, and the customers optimize their energy consumption schedules under the given price in a distributed manner. At the point of Nash Equilibrium, the minimum total energy cost and the decreased PAR is achieved. In [7], the temporally coupled constraints of the local consumer's energy scheduling problem are included, and the coupled-constrained game model is tackled by dual decomposition. In [8], the authors prove that the non-cooperative game between the users and the utility provider is the general case of the minimum peak-to-average ratio problem. In [9,10], the gradient method is utilized for solving local consumption schedule problem with fast convergence. In [11], an online learning algorithm is developed, where each user learns through past experience to approximate other users' decisions, and to optimize its own energy scheduling.

All the above methods can be categorized as model-based methods, where the mathematical equations are formulated to describe local users' energy scheduling. Because the demand-side management problem is usually a partially observable problem, i.e., unknown or uncertain information exists, the models are generally solved in an iterative way. There are two deficiencies of the iterative algorithm: 1) the convergence of the algorithm cannot always be guaranteed. The convergence can only be achieved under some strict prerequisites, e.g., convex payoff functions, which require certain assumptions and simplifications of the problem; 2) applying an iterative algorithm in the real-world can be impractical, especially in real-time scenarios. In real-world practices, it is more likely that the utility provider releases the price signal, and the consumers schedule their consumption accordingly, which tends to be a one-step process. The iterative interaction between the two sides can be both time-consuming and resource-consuming with the potential challenge of divergence.

Based on the above challenges and motivations, we propose a data-driven method in this paper for optimizing demand-side energy management. Especially, we propose the combination of two techniques, the deep neural networks (DNN) and the reinforcement learning (RL) method to overcome the complexity and inefficiency of model-based methods. The recent years have witnessed the rapid advancement of deep neural network in a variety of applications, e.g., computer vision, machine translation, and remote sensing. In the field of power system, the deep neural network has been applied for prediction of uncertain factors [12]-[14] , smart meter data

identification [15], modeling of renewable energy [16], and energy storage dispatch [17]. The DNN is a data-driven method that does not rely on any analytical equations, but it utilizes voluminous existing data to formulate the mathematical problem and to approximate the solutions. The multiple hidden layers and the large number of neurons within the DNN can automatically extract features for data analysis to achieve an accurate model regression or classification. Once the DNN is well trained, it will develop high generalization and can be directly applied to new instances without costly numerical computation. Compared to the conventional model-based method, the DNN is highly computational efficient while maintaining considerable accuracy.

The reinforcement learning (RL) method is well known for its applicability in solving problems with hidden information. Reinforcement learning focuses on providing the optimal time-sequential decisions within an unknown environment. This is realized via continuous interactions between the decision-maker, which is called the agent, and the environment. Through this learning process, the agent is able to gain knowledge of the environment and to take actions that affect the environment in order to reach its objective. Currently, RL has been widely spotted in areas including robotics and automation, computer games, auto pilot, and dialog system.

There have also been significant efforts in implementing RL method for solving complex power system problems. The utilization of RL to optimize the residential demand response schedule is first discussed in [18]. The method is later decomposed to the device-level to achieve higher computational efficiency [19]. The research in [20] further includes the smart energy hub to the residential DR management to initiate a real-time energy monitoring and to boost the learning process. In [21,22], both a deep neural network and reinforcement learning are leveraged for an economically efficient residential load control. The deep neural network is used to estimate the potential reward of each move of the consumer, and the reinforcement learning is used to coordinate the actions from a long-term perspective. This combination is called deep reinforcement learning (deep RL). The authors in [23] proposed the application of deep RL to optimize the real-time electric vehicle charging schedule with the consideration of future electricity price. The feasibility of applying deep RL to load frequency control with stochastic renewable energy penetration is investigated in [24]. More potential applications of deep RL in power system studies have been discussed in [25].

Inspired by the previous works, in this paper, we also propose the utilization of both deep neural network and reinforcement learning method to solve the problem of multi-microgrid (MMG) energy management. Different from the load control model in the previous works, a microgrid contains both generation and consumption units, leading to more variables and constraints with higher model complexity. In such cases, the conventional model-based method may become inapplicable due to the computational burden, which makes the data-driven method a more desirable and efficient alternative solution.

The main contributions of this work are summarized as follows:

1) A data-driven DNN is constructed to model the multi-microgrid response under dynamic retail price signals. The DNN is trained based on historical data and without requiring the user information from local microgrid operators. Uncertain factors within the microgrid system are also included in the training set. The well-trained DNN has high generalization and can automatically generate multi-microgrid power exchange under the new given input.

2) A model-free RL technique is applied for the distribution system operator (DSO) to optimize the retail pricing for local microgrids. The RL method aims to maximize the profit of selling power while reducing the peak-to-average ratio. The DSO is able to achieve a near-optimal pricing strategy with the substantial exploration ability of the proposed RL method.

3) A comprehensive performance evaluation of the proposed method is provided through various simulations to verify its feasibility in practical scenarios. A comparison with model-based method is also presented to demonstrate the superiority of the proposed reinforcement learning method.

The rest of the paper is organized as follows: Section II presents the mathematical model of the MMG energy management problem; Section III demonstrates the detailed design of the proposed DNN and the training process; Section IV elucidates the model-free RL algorithm for retail price setting of DSO; Section V provides the simulation results of the proposed algorithm as well as observations and analysis; finally, Section VI concludes the paper.

II. Modeling of Multi-Microgrid Energy Management

In this section, we first introduce the mathematical model of the proposed multi-microgrid energy management problem. The interaction between the MMG and distribution system is shown in Fig. 1. In the figure, a bi-directional communication channel is constructed between the microgrids and the DSO, where the DSO releases its retail price to the microgrids, and the microgrids send back the amount of power to purchase. The goal of MMG energy management is to smoothen the hourly power exchange profile of the MMG with proper retail price setting strategies.

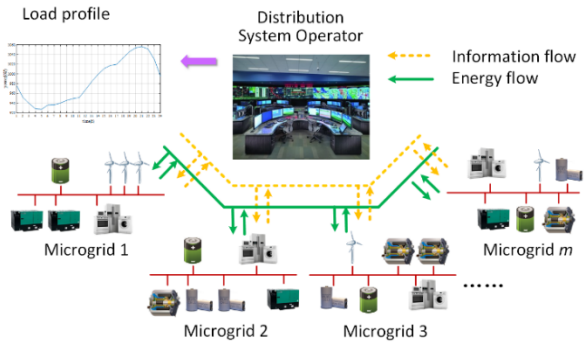


Fig. 1. Multi-microgrid energy management under DSO pricing control

From the perspective of an individual microgrid, each microgrid operator attempts to minimize its operation cost under the given retail price, which leads to the following microgrid economic dispatch (ED) model:

$$Min\sum_{t=1}^{N_T}\Big(\sum_{k\in m}C_{DG}^{P}(P_k^{DG}(t))+\eta_m\lambda(t)P_m^{grid}(t)+\sum_{z=1}^{Z}ec_m^z q_m^z(t)u_m^z(t) + \rho_{es}\,|\,SOC_{es}(t)-SOC_{es}(t-1)\,|\Big) \quad (1)$$

$$C_{DG}^{P}(P_k^{DG}(t)) = a_k^p + b_k^p P_k^{DG}(t) + c_k^p (P_k^{DG}(t))^2 \quad (2)$$

The objective function (1) represents the operation cost of the m^{th} microgrid over dispatch cycle N_T, which is usually 24 hours. The first term in (1) is the generation cost of the k^{th} dispatchable generator, which has a quadratic form of the generation quantity $P_k^{DG}(t)$, as shown in (2). The second term in (1) is the power exchange cost, where $\lambda(t)$ is the retail price at the point of common coupling (PCC), and η_m is a factor to represent network losses. $P_m^{grid}(t)$ is the power purchased by the microgrid. Note that η_m can differ among different microgrids, because the locations of the microgrids within the distribution network may vary. Thus, each microgrid bears different network losses and receives different retail prices, which is also known as distribution locational marginal price (DLMP). The third term in (1) is the cost of dispatching DR resources that reside in the microgrid, where $u_m^z(t)$ is a 0-1 binary variable indicating whether the z^{th} demand response block $q_m^z(t)$ is dispatched or not, and ec_m^z is the unit price [26]. And the last term is the degradation cost of energy storage. The change between two consecutive states of charge (SOC) is measured as the energy storage life degradation caused by charging or discharging [27]. Microgrid economic dispatch should also satisfy the following constraints:

$$P_k^{DG,\min} \le P_k^{DG}(t) \le P_k^{DG,\max} \quad (3)$$

$$0 \le \sum_{z=1}^{Z} q_m^z(t) u_m^z(t) \le P_m^{Load}(t) \quad (4)$$

$$u_m^{z-1}(t) \ge u_m^z(t), \text{ for } z = 2,...,Z \quad (5)$$

$$0 \le P_{es}^{ch}(t) \le P_{es}^{ch,\max}, 0 \le P_{es}^{dis}(t) \le P_{es}^{dis,\max} \quad (6)$$

$$SOC_{es}(t) = SOC_{es}(t-1) + \eta_{es} P_{es}^{ch}(t)\Delta - P_{es}^{dis}(t)/\eta_{es}\Delta \quad (7)$$

$$SOC_{es}^{\min} \le SOC_{es}(t) \le SOC_{es}^{\max} \quad (8)$$

$$P_m^{Load}(t) - P_m^{grid}(t) - \sum_{k\in m} P_k^{DG}(t) - \sum_{es\in m} (P_{es}^{dis}(t) - P_{es}^{ch}(t)) - \sum_{z=1}^{Z} q_m^z(t) u_m^z(t) = 0 \quad (9)$$

Constraint (3) is the generator capacity constraint of DGs in the m^{th} microgrid; constraints (4)-(5) mean that the total demand response dispatched should not exceed the load $P_m^{Load}(t)$, and the demand response blocks are dispatched in an increasing order; constraint (6) is the charge/discharge rate limit of the energy storage, where $P_{es}^{ch}(t)$ and $P_{es}^{dis}(t)$ are the charging and discharging quantity of the energy storage; constraint (7) calculates the energy level of energy storage, which is $SOC_{es}(t)$, where η_{es} is its efficiency and Δ is the length of the time interval; constraint (8) is the capacity limit of energy storage; and finally, constraint (9) is the power balance constraint of the microgrid.

The DSO decides the retail price by solving the following optimization problem:

$$Max\ \alpha\left(\sum_{t=1}^{N_T}\lambda(t)\sum_{m=1}^{N_m}\varepsilon_m P_m^{grid}(t)\right)/\ profit_{base} - (1-\alpha)P_{\max}^{grid}/\ P_{avg}^{grid} \quad (10)$$

$$P_{\max}^{grid} \ge \sum_{m=1}^{N_m}\varepsilon_m P_m^{grid}(t),\ for\ t=1,...,N_T \quad (11)$$

$$P_{avg}^{grid} = \sum_{t=1}^{N_T}\sum_{m=1}^{N_m}\varepsilon_m P_m^{grid}(t)/N_T \quad (12)$$

In (10), the first term is the DSO's profit from selling energy to the microgrids, where N_m is the total number of microgrids. ε_m is a conversion factor. This is because $P_m^{grid}(t)$ is calculated by the local microgrid operators and does not include the network losses, hence cannot reflect the real amount of power exchange at PCC. The function ε_m is to transform the local power exchange to the power exchange at PCC. Due to the limit of the page lengths, we do not consider the detailed distribution network topology in this paper for a full-fledge DLMP model and assume that ε_m is a known value in the following simulations.

The second term in (10) is the peak-to-average (PAR) ratio over the entire dispatch cycle, which is the ratio between the maximum power exchange and the average power exchange of MMG. Since PAR is unitless, the first term is divided by a constant base $profit_{base}$ value to remove its unit. The DSO intends to find the optimal retail price $\lambda(t)$ that maintains a balance between the two objectives, hence there is a weighting factor α added before the two terms.

The difficulty of solving (10) is that the individual microgrid power exchange $P_m^{grid}(t)$ varies with the retail price $\lambda(t)$, hence it cannot be solved directly. In the following sections, we will introduce two data-driven techniques, the DNN and RL, to crack the above problem with high computational efficiency.

III. Multi-Microgrid Operation Simulation with Deep Neural Network

In this section, a deep neural network is applied to simulate the multi-microgrid operation under given price signals, i.e., to solve (1)-(9). There are two main advantages of utilizing the DNN:

1) The neural network is readily available as a toolbox. Once the parameters are well-trained, it has high generalization and can automatically generate the estimated amount of power exchange between the MMG and DSO under the new retail price. Given that the individual microgrid economic dispatch model is a nonconvex problem and that the number of microgrids can be large, solving the MMG power exchange using the conventional analytical method can be highly time-consuming. The data-driven DNN has much higher computational efficiency with considerable accuracy;
2) The individual microgrids do not need to expose their generation or consumption information to the distribution system operator (DSO), given that the DNN is trained using the historical retail price data and power exchange data. Therefore, the user privacy of microgrid owners is well protected.

A. Deep Neural Network Structure

The artificial neural network has long been recognized as an efficient regression tool for handling problems that are difficult to accurately model or with high computational complexity. MMG energy management fits this category. Hence, a DNN is constructed as follows:

As shown in Fig. 2, the input to the DNN is the retail price, and the output is the aggregated MMG power exchange with the distribution system under the given price signal. The goal of the DNN is to generate a simulated power exchange that is as close as possible to the actual MMG response.

Before sending the raw training data to the DNN for regression analysis, data preprocessing is implemented. The

function of data preprocessing is to minimize the deviation of the training data for improving the regression accuracy and computational efficiency.

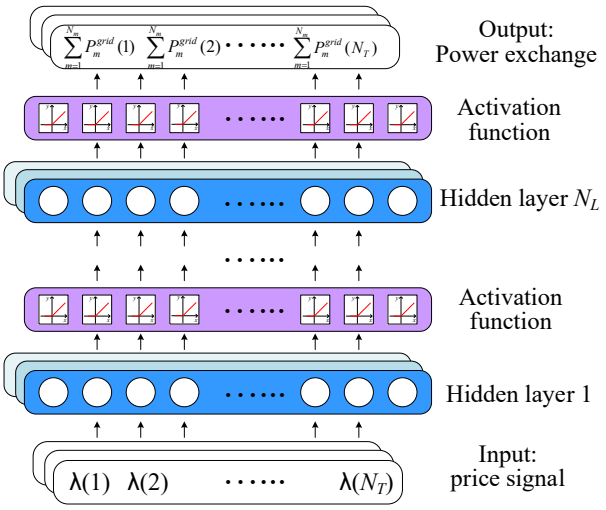


Fig. 2. Multi-layer structure of the Deep Neural Network

The data preprocessing for MMG response raw data includes two steps: firstly, all the sample input data and output data are transformed into the per unit value. By utilizing the per unit value, different features of the sample data become comparable with each other. For the retail price sample, given that they are at the scale of 10$/MWh, 100$/MWh is set as the base value; for the aggregated MMG power exchange, given that they are at the scale of 100 kW, 1000 kW is set as the base value.

Secondly, a *min_max_scaler* transformation is applied for further normalization, as shown below:

$$\lambda_s^{new}(t)=\frac{(\lambda_s(t)-\min_s \lambda(t))}{(\max_s \lambda(t)-\min_s \lambda(t))} \tag{13}$$

In (13), s is the index of training samples, $\max_s\lambda(t)$ and $\min_s\lambda(t)$ are the maximum and minimum values of the retail price at the t^{th} interval among the entire training set. Through the above normalization, the values of the retail price samples will lie within the range of [0,1]. The above data preprocessing helps create a more regular search region for faster algorithm convergence.

In the DNN structure, between the input layer and the output layer are numerous hidden layers. The term "deep" refers to the multiplicity of hidden layers. Each hidden layer is composed of neurons that complete the following affine transformation of the input:

$$y_{sk}^{(l)}=\sum\nolimits_{j=1} x_{sj}^{(l)}\omega_{jk}^{(l)}+b_k^{(l)} \tag{14}$$

The calculation of the output of the l^{th} hidden layer is shown by (14), where s is the index of the sample, j is the index of the features of the sample, and k is the index of neurons. Also, $\omega_{jk}^{(l)}$ is the weight assigned to the j^{th} feature of the input, and $b_k^{(l)}$ is the bias. As can be observed, the output $y_{sk}^{(l)}$ is the weighted aggregation of all the features of the input $x_s^{(l)}$ captured by the k^{th} neuron. The function of the hidden layer is to extract sufficient features from the input data and to construct the mapping between the input and the output.

Notice that (14) is a linear transformation. However, the microgrid ED model (1) is nonlinear, and cannot be handled by a mere linear transformation. An activation function is thus added to the hidden layer to delinearize the model, as shown in Fig. 2. In this study, the rectifier linear units (ReLU) function is used as the activation function [28]. The ReLU function has the following form: $f(x) = \max(x,0)$, which is very close to a linear expression. Hence, the gradient-based methods used in linear optimization can be easily applied to ReLU-based nonlinear models. The ReLU function also preserves strong generalization abilities.

B. DNN Training Algorithm

In the DNN, the network parameters $\omega_{jk}^{(l)}$ and $b_k^{(l)}$ are the unknown variables that need to be calculated. The back-propagation algorithm is applied for this cause. Before the implementation of the algorithm, a loss function is defined as the objective of the DNN training. The loss function implies the accuracy of the output from the DNN. In the MMG energy management problem, mean square error (MSE) is utilized as the loss function:

$$L(\omega,b)=\frac{1}{N_S N_T}\sum_{s=1}^{N_S}\sum_{t=1}^{N_T}(P_{total,s}^{grid*}(t)-P_{total,s}^{grid}(t))^2 \tag{15}$$

$$P_{total,s}^{grid*}(t)=\sum_{m=1}^{N_m}\varepsilon_m P_{m,s}^{grid*}(t),\ \ P_{total,s}^{grid}(t)=\sum_{m=1}^{N_m}\varepsilon_m P_{m,s}^{grid}(t) \tag{16}$$

In (15), N_S is the number of training samples, $P_{total,s}^{grid*}(t)$ is the actual MMG power exchange at the t^{th} time interval of the s^{th} sample, $P_{total,s}^{grid}(t)$ is the estimated MMG power exchange. The loss function tries to minimize the deviation between the ground truth and the estimated value to obtain an accurate enough approximation of the MMG response.

In the studied MMG system, there exist uncertainties, e.g., distributed renewable generation fluctuation, load variations. These uncertainties may cause extremely large or small power exchanges. The existence of such abnormal values in the training set can lead to the issue of overfitting, where the DNN attempts to fit to all the training samples and loses its generalization.

To overcome the overfitting problem, we introduce L_2 regularization to the loss function (15), which is shown as follows [28]:

$$L(\omega,b)=\frac{1}{N_S N_T}\sum_{s=1}^{N_S}\sum_{t=1}^{N_T}(P_{total,s}^{grid*}(t)-P_{total,s}^{grid}(t))^2+\frac{\alpha}{2}\omega^T\omega \tag{17}$$

In (17), a norm-2 penalty for parameters, $\alpha/2\omega^T\omega$, is added to the loss function. α is called the regularization parameter, which is a positive constant. The norm-2 penalty term restricts that the values of weight parameters do not grow excessively large to fit to the abnormal values and noises, hence the generalization of the model can be maintained.

Once the loss function is calculated, the first partial derivatives of the loss function to the weights and biases can be obtained and used to update the variables:

$$\omega_l^{(i+1)}=\omega_l^{(i)}-\eta\frac{\partial L}{\partial J_{N_L-1}^{(i)}}\cdot\frac{\partial J_{N_L-1}^{(i)}}{\partial J_{N_L-2}^{(i)}}\cdots \tag{18}$$

In (18), i is the index of iteration, l is the index of hidden layers, N_L is the total number of hidden layers, $J_l^{(i)}$ is the output of the l^{th} layer, η is called the learning rate. Since the DNN has multiple hidden layers, the chain rule is applied to calculate the partial derivative of the parameters at each layer. The bias b is updated similarly. As can be observed, the back-propagation

algorithm utilizes the gradient to manipulate the neural network parameter, and to guide the model's evolution toward the global optimum.

IV. Monte Carlo Reinforcement Learning Method for DSO Decision-making

In Section III, the deep neural network is constructed to simulate the multi-microgrid operation under the given price. As such, the DSO can obtain a reliable estimation of the aggregated MMG power exchange without much computation. Next, the DSO will decide the optimal retail price setting with the goal of maximizing the profit of selling power and minimizing the PAR, as shown by (10).

Note that the PAR in (10) is not an explicit expression of the decision variable, which is the retail price $\lambda(t)$, hence (10) is difficult to solve. In previous literature, similar problems are usually solved in a distributed and iterative manner, where the utility provider first releases the retail prices, and each local user sends back their power consumption under the given price. The utility provider then evaluates the current PAR and adjusts the price accordingly. The above process repeats until no power consumption change or price change happens.

The iterative method is not applicable to MMG energy management problem for the following two reasons: 1) In previous studies, the local users are only consumers and are only allowed to shift their load. In this way, the total energy demand becomes constant and the average hourly load can be calculated, which only leaves P_{max}^{grid} unknown, as is the case in [6,9]. However, in the MMG case, since each microgrid is a prosumer, their final energy consumption cannot be predicted, hence both P_{max}^{grid} and P_{avg}^{grid} are unknown terms, and the division leads to a nonconvex problem, the iterative algorithm cannot guarantee to converge in a nonconvex case; 2) The iterative algorithm can be time-consuming and resource-consuming, and not applicable for real-time applications.

Motivated by the above considerations, the reinforcement learning method is applied in this paper to crack the intractability of the DSO pricing problem. The reinforcement learning method is well-known for its applicability to problems with unknown search spaces. For example, in the MMG energy management problem, both maximum power exchange P_{max}^{grid} and average power exchange P_{avg}^{grid} remain unknown to the decision-maker DSO, and they are also not analytically expressed as functions of the retail price. The reinforcement learning method has strong exploration abilities through continuous interactions with the unknown environment and constantly update the agent's experience in order to make the optimal decision. In this section, we will discuss how to implement the reinforcement learning method to optimize the retail pricing strategy of DSO.

A. A Brief Overview of Reinforcement Learning

Reinforcement learning is a type of machine learning approach focusing on how agents take actions within an unknown environment with the goal of maximizing reward [29]. Briefly speaking, in a provided environment, at each state, the agent randomly takes an action, and receives an immediate reward from the environment. Then the agent moves to the next state with a certain probability and repeats the above process, as shown in Fig. 3:

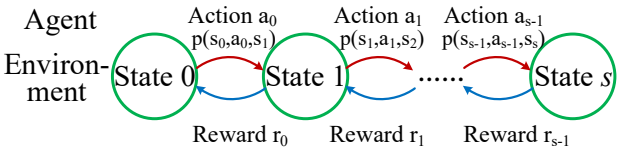


Fig. 3. Illustration of reinforcement learning

In the beginning, the agent has no knowledge of what reward and next state are linked to each action. To maximize the accumulative reward, the agent must learn the above knowledge by continuously interacting with the environment. In most cases, the action taken at the current state not only affects the immediate reward, but also the next state and all the future rewards. Hence, it can be concluded that reinforcement learning is a decision-making process with trial-and-error-search and delayed reward.

B. Mapping Multi-microgrid Energy Management Problem to Reinforcement Learning

The reinforcement learning assumes that the problem under study is a Markovian Decision Process (MDP), which is composed of four fundamental elements: 1) a series of environment states $\boldsymbol{S}$; 2) a set of actions $\boldsymbol{A}$; 3) the transition function $\boldsymbol{P}$ that gives the state probabilities; and 4) a sequence of rewards $\boldsymbol{R}$.

In the MMG energy management problem, the fundamental elements of the reinforcement learning are defined as follows:

- The agent: DSO
- State: hourly total power exchange of MMG, $\sum_{m=1}^{Nm} P_m^{grid}(t)$, for $t = 1\ldots, N_T$
- Action: hourly retail price $\lambda(t)$, for $t = 1\ldots, N_T$
- Reward: Hourly profit of selling power, $\lambda(t) \sum_{m=1}^{Nm} \varepsilon_m P_m^{grid}(t)$

The ultimate objective of DSO is to maximize the total profit of selling power over the entire dispatch cycle, plus weighted PAR, as shown in (10). Since both the accumulative profit and PAR are decided by the power exchange through the entire dispatch cycle instead of a single time step, the DSO has to be farsighted to predict the future MMG power exchange when deciding the retail price for the current time step. This corresponds with the delayed-reward feature of reinforcement learning, and makes reinforcement learning a natural fit for the MMG energy management problem.

Note that the transition function is not given in the above definitions. This is because the state in this problem is the hourly total power exchange of MMG, which is difficult to predict. The hourly power exchange of microgrids are related to various uncertain factors within the microgrid system, e.g., load variations and distributed renewable energy. In the next subsection, we will introduce a model-free method to overcome the barrier of lacking transition function.

C. Model-free Monte Carlo Method

There are two types of reinforcement learning methods: the model-based method and the model-free method. The former assumes that the problem is a known Markov Decision Process with full knowledge of state transition probabilities. In this way, the problem can be solved analytically via dynamic programming or other iteration methods. However, for some reinforcement learning problems, obtaining the transition probabilities is not a trivial task. In such occasions, the agent has to estimate the transitions and rewards from the interactive

experiences with the environment. This is called the model-free method, since no state transition model can be constructed in advance due to the lack of information.

The Monte Carlo method is a type of model-free method. To obtain the state and the reward information, the Monte Carlo method deploys the simplest possible policy. It utilizes the averaged sample reward for a certain action as its reward value. According to the law of large numbers, when there are enough simulations and enough samples of reward, the averaged value is approximately equal to the actual value, which proves the reasonability of the Monte Carlo method.

As has mentioned before, it is very difficult to obtain the transition probability of the state, which is the hourly total power exchange of MMG, mainly due to the microgrid uncertainties. Therefore, in the MMG energy management problem, we also adopt the model-free Monte Carlo method to optimize the retail pricing strategy of the DSO. The Monte Carlo method is displayed in Algorithm 1 [29]:

Algorithm 1: Monte Carlo Method for DSO decision-making

1: Generate daily retail price sequence samples N_S
2: Input the price samples to the DNN to obtain the MMG power exchange profile
3: **for** t in 1 to N_T **do**
4: Choose retail price $\lambda^{(s)}(t)$ from price samples N_S
5: Initialize the counter $n(s) \rightarrow 0$
6: **for** s' in 1 to N_S **do**
7: **if** $\lambda^{(s')}(t)$ equals $\lambda^{(s)}(t)$
8: **do** $n(s) \rightarrow n(s) + 1$
9: **end if**
10: **end for**
11: Evaluate $\lambda^{(s)}(t)$ based on average weighted reward: $r(\lambda^{(s)}(t)) = 1/n(s) \cdot(\alpha \sum profit(\lambda^{(s)}(t)) - (1-\alpha) \sum \mathrm{PAR}(\lambda^{(s)}(t)))$
12: Select $\lambda(t) = \mathrm{argmax}\, r(\lambda^{(s)}(t))$, for all $s \in N_S$
13: **end for**

Algorithm 1 is explained as follows: to begin with, the DSO randomly generates large quantities of retail price sequence samples. The price samples are then sent into the DNN to obtain the estimated aggregated MMG power exchange. After the generation of all the price samples and the power exchange samples, the DSO selects the optimal hourly retail price based on the procedure as follows:

First, at each time step t, the DSO randomly picks a retail price $\lambda^{(s)}(t)$ from the sample set, then counts the number of price samples that contains $\lambda^{(s)}(t)$ and records it as $n(s)$.

Then, the DSO evaluates $\lambda^{(s)}(t)$ based on its average profit and average PAR. The $profit(\lambda^{(s)}(t))$ is calculated as follows:

$$profit(\lambda^{(s)}(t)) = \sum_{k=t}^{N_T} \gamma^{k-t} \lambda^{(s)}(k) \varepsilon_m P_m^{grid,s}(k) / profit_{base} \quad (19)$$

It can be seen from (19) that the profit of $\lambda^{(s)}(t)$ is the discounted accumulated profit of selling power from t to N_T, where the discount factor γ is between 0 and 1. When γ is zero, it implies that the decision-maker focuses only on the current profit and is totally myopic; when γ is greater than zero, it means that the decision-maker is farsighted by evaluating the current pricing with the consideration of potential future profit. In this study, γ is set to 0.9 to ensure that the DSO has a more robust pricing strategy to avoid future risks.

Next, for a single price $\lambda^{(s)}(t)$, the PAR under the price sequence $[\lambda^{(s)}(1), \ldots, \lambda^{(s)}(t), \ldots, \lambda^{(s)}(N_T)]$ is taken as $\mathrm{PAR}(\lambda^{(s)}(t))$. The PAR also reveals part of the influences of selecting $\lambda^{(s)}(t)$ at the current time interval to the potential future decisions. Each price is then evaluated based on the weighted sum of average $profit(\lambda^{(s)}(t))$ and average $\mathrm{PAR}(\lambda^{(s)}(t))$, as shown in line 11 of Algorithm 1. The weight factor α represents the tradeoff between maximizing profit and minimizing PAR.

Finally, all the prices are compared and the price with the maximum weighted reward is selected as the price for time step t, as shown in line 12. The above process is repeated for all the time steps until the whole optimal retail price sequence is decided.

The above algorithm is a Monte Carlo method because the DSO selects the optimal price sequences from a randomly generated sample set. Note that in the above algorithm, the price for each time step is selected separately, i.e., the price selection process (line 6-line 10) repeats for N_T times to obtain a complete price sequence. A more intuitive way is to directly select the price sequence with the maximum weighted reward from the sample set. However, this intuitive method cannot guarantee to reach global optimization when the possible realizations of the price sequence are huge. For instance, if there are N_T time steps in a dispatch cycle, and for each hour, there are N_p possible prices, then the total number of candidate price sequences will be N_p^(N_T), which can be an enormous figure even for small N_p and N_T, and cannot be completely represented by a limited sample set. By using the average value to evaluate each hourly price and regrouping them, the algorithm can explore beyond the given sample set and discover solutions better than the existing combinations. This judgement will be verified in the simulation part in next section.

V. Simulation Analysis

In this section, we first reveal the detailed structural design of the DNN for simulating multi-microgrid operation. Then the testing performance of the DNN is presented. Next, based on the simulated results from the DNN, the model-free Monte Carlo method is applied for the DSO to decide the optimal pricing strategy. The results are evaluated and compared with a conventional model-based method to demonstrate the advantages of the proposed data-driven method.

A. Simulating Multi-microgrid Operation with DNN

1) Multi-microgrid system setup

A test case where 10 microgrids are connected to one DSO is considered here. For simplicity, we assume that microgrids with greater serial number are farther away from PCC, hence suffer from more network losses and receive a higher retail price. The η_m for the 10 microgrids are assumed to be in the range of 1.01-1.1, with an incremental size of 0.01. The setting of η_m is aligned with the results of distributional locational marginal price (DLMP) in [30], in which the DLMP range is around 100% to 110% of the price at PCC. The ε_m is assumed to be the same as η_m. The compositions of each microgrid are summarized in TABLE I.

TABLE I Microgrid Composition

No.	Compositions	No.	Compositions
1	WT, DE, DE, ES, DR	2	WT, DE MT, FC,ES,DR
3	WT, MT,MT,FC, ES,DR	4	WT, MT, FC,ES,DR
5	WT, DE, MT, MT,ES	6	WT, DE, FC, FC,ES,DR
7	WT, DE, DE, FC, ES, DR	8	WT, FC, FC, ES, DR

9	WT, DE, MT, FC, FC, ES, DR	10	WT, MT, MT, MT,ES,DR

WT: wind turbine; DE: diesel generation; MT: micro turbine; FC: fuel cell; ES: energy storage; DR: demand response

TABLE II PARAMETERS OF DISTRIBUTED ENERGY RESOURCES

DG type	$P_k^{DG,min}$ (kW)	$P_k^{DG,max}$ (kW)	Quadratic coefficients a_k^p ($/h)	b_k^p ($/kWh)	c_k^p ($/kW^{2}h)
Micro turbine	0	30	0.4	0.0397	0.00051
Fuel cell	0	30	0.38	0.0267	0.00024
Diesel generator	0	60	1.3	0.0304	0.00104
Energy storage	SOC_{es}^{min} (kWh)	SOC_{es}^{max} (kWh)	$P_{es}^{ch,max}$ (kW)	$P_{es}^{dis,max}$ (kW)	η_{es}	ρ_{es} ($/MW)
	20	50	25	25	0.9	100
DR quantity	33% of total DR		66% of total DR		100% of total DR	
DR unit price ($/kWh)	0.44		0.46		0.48	

2) Design of DNN regression model

We design a DNN with 3 hidden layers for simulating MMG operation under the given retail price. The number of neurons in each layer is 1000. The number of inputs and outputs are both 24, since there are 24 hourly prices with 24 hourly power exchanges (i.e., the dispatch cycle considered here is 1 day). The number of neurons in each hidden layer is decided via repeated trial and error. The selection of the number of neurons is a trade-off between the regression accuracy and computational efficiency. The self-adaptive Adam Optimizer is applied with an initial learning rate of 1e-2 [31]. In addition, the exponential decay of learning rate is applied to stabilize the training. The initial values of the weights and biases of the DNN are obtained from Xavier initialization [32]. Furthermore, to guarantee that the output from each hidden layer is regularized within a certain range, batch normalization is applied to avoid algorithm divergence [33].

In this case study, 12,000 samples of retail price and power exchange are generated for the neural network training. In the first place, the daily retail price is randomly generated as 1 to 1.5 times higher than the wholesale market price, with a step size of 0.1. The wholesale market price can be obtained from historical market data. Then the generated retail price data is sent to model (1)-(9) to calculate the hourly power exchange of each participated microgrid. In addition, there exist uncertain factors with the microgrid, e.g., the output of wind turbine, and the demand variation. To make the DNN regression model more robust against uncertainties, we assume that the forecast error of load and wind generation follows a normal distribution with zero mean and a standard deviation of 0.1 and 0.05, respectively. Because a large number of training samples are generated to cover enough uncertain scenarios, the well-trained deep neural network has high generalization to unseen microgrid uncertainties and can provide regression results with high accuracy.

3) DNN training and test results

The conventional model-based method is used at this stage to solve model (1)-(9). In this study, we use GAMS/CPLEX software package to solve the model. The ratio of training samples to testing samples is 8:2. The total number of iterations is 2,000. The hardware environment is a Nvidia GeForce GTX 1080 Ti Graphic Card with 11 GB memory and 1.582 GHz core clock. The software environment is the online open-source deep learning platform TensorFlow, which is implemented on Python. The whole simulation framework is shown in Fig. 4.

TABLE III shows the detailed settings of DNN training. The training result is summarized as follows: for the 2,400 test samples, the average relative error of estimated power exchange is 0.96%, which indicates the considerable accuracy of DNN regression. The training loss (mean square error plus L_2 weighted penalty) for 9,600 training samples is 0.0034, which is small enough as an indicator of the training convergence. The time for completing 2,000 iterations is 91.05 s, which is acceptable since the training is completed off-line.

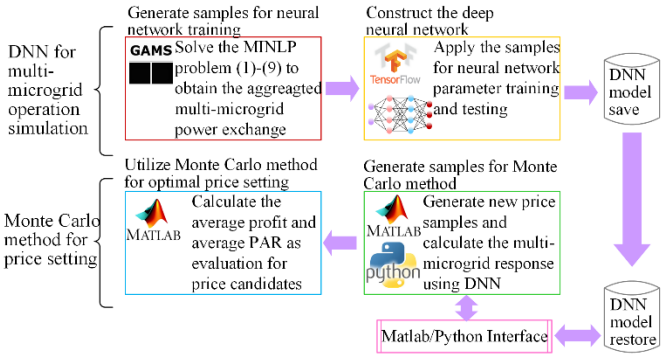


Fig. 4. Simulation framework for multi-microgrid energy management

TABLE III SUMMARY OF DNN TRAINING SETTINGS

Item	Value
No. of hidden layers	3
No. of neurons in each hidden layer	1000
Activation function	ReLU
Loss function	Mean square error plus L_2 regularization
Learning rate	1e-2
Exponential decay rate	0.96
Exponential decay step	50
Optimizer	Adam Optimizer
No. of training samples	9,600
No. of test samples	2,400
Iteration steps	2,000
Data preprocessing	min_max_scaler

4) Sensitivity analysis

To further verify the high generalization of the well-trained DNN to unseen inputs, we conduct the following sensitivity analysis of the DNN regression accuracy.

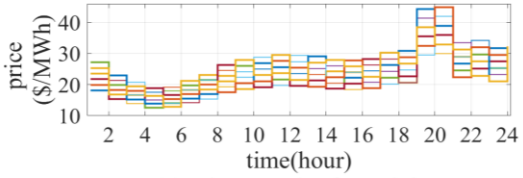
First, the effect of price disturbance is discussed. As previous discussed, for the training set generation, the daily retail price is randomly generated as 1 to 1.5 times higher than the wholesale market price. To include the price disturbance, in the test set generation, we manually create a price peak at hours 10-11. Note that this disturbance is not included in the training set. The comparison of price samples for training and test is shown in Fig. 5.

Note that there appear to be only six lines in Fig. 5, because some price samples are overlapped with others. A test set with the size of 500 based on the above price disturbance is generated and input to the DNN. The average relative error of estimated power exchange is 1.65%. Note that this error is slightly higher than the above 0.96%, given that the price disturbance is not included in the training set. Still, this average relative error is low enough to verify the robustness of DNN regression under price disturbance.

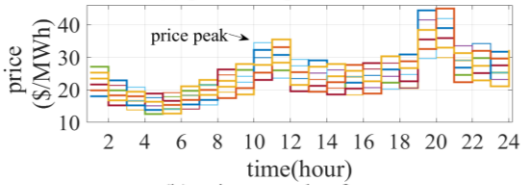
We further explore the effect of microgrid load variation to the DNN regression accuracy. Similar to retail price

disturbance, we manually create a load valley for hours 13-14 on the original microgrid load profile for the test set generation. The comparison of the microgrid load for training and test is shown in Fig. 6. A test size with the size of 500 based on the microgrid load disturbance is generated and input to the DNN. The average relative error of estimated power exchange is 1.60%. This further verifies the robustness of DNN regression under load profile disturbance.

Based on the above observations, it can be safely concluded that the DNN has formulated a considerably accurate regression model between the input, which is the retail price, and the output, which is the MMG power exchange, and is immune to the unseen disturbance in the input data. This is due to the strong automatic feature extraction ability of the large number of neurons embedded within the DNN. As a result, the DNN has tremendous potential in solving problems with unclear or complex mathematical formulations.

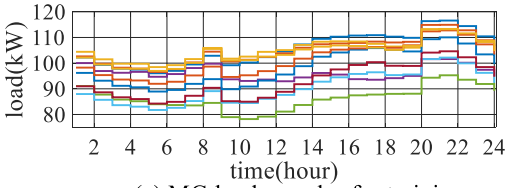


(a) price samples for training

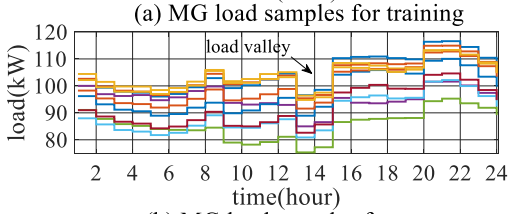


(b) price samples for test

Fig. 5. Disturbance of retail price



(a) MG load samples for training



(b) MG load samples for test

Fig. 6. Disturbance of microgrid load

B. Monte Carlo Method for Optimizing DSO Pricing Strategy

Once the deep neural network is well trained, the fine-tuned parameters can be properly stored for repeated use. The DSO can now apply the Monte Carlo method to search for the optimal retail price for the MMG. Since the Monte Carlo method is based on the law of large numbers, the more samples are generated, the closer the obtained solution is to the actual global optimum. As previously mentioned, each hourly retail price falls within 1 to 1.5 times of the wholesale market price, with a step size of 0.1. Then the total number of all possible price sequences is $5^{24} \approx 5.96 \times 10^{16}$, which is far beyond the hardware's computation capabilities. Instead, we generate 10^4, 2×10^4, 5×10^4, and 8×10^4 price samples respectively, to observe the effect of sample sizes on the performance of the Monte Carlo method.

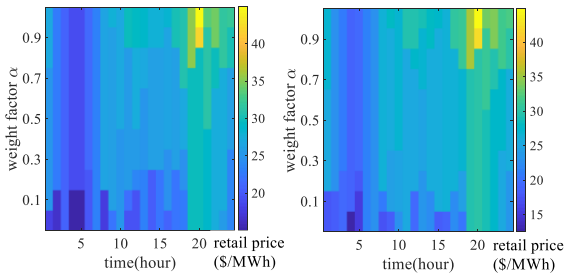
TABLE IV COMPUTATION TIME FOR DNN AND MONTE CARLO METHOD

No. of samples	Calculation time(s)	
	DNN	Monte Carlo
10,000	2.67	31.35
20,000	3.84	35.25
50,000	7.90	41.82
80,000	12.51	51.44
No. of samples	**Calculation time(s) (model-based method)**	**Acceleration ratio of DNN**
10,000	28,301	10,600

In the first place, the computation time for using the DNN to calculate an MMG power exchange, and for the Monte Carlo method to scan all the generated samples for price setting are shown in TABLE IV. The DNN calculation and Monte Carlo method are implemented on Matlab R2017b plus Python, and the hardware environment is a laptop with Intel®Core™ i7-7600U 2.8 GHz CPU, and 16.00 GB RAM. As seen from Table IV, using the well-trained parameters of the DNN to calculate the approximated MMG power exchange is fast enough to generate large numbers of samples for the Monte Carlo method. Also, the proposed Monte Carlo method is able to scan through large quantities of candidate retail price sequences with an acceptable time elapse.

In addition, we also test the computing time for solving (1)-(9) using a conventional model-based method. The software solver is GAMS/CPLEX, and the hardware environment is the same as previously mentioned. The computational efficiency is shown in the last row in TABLE IV. The acceleration ratio is the ratio between the computation time of model-based method and the DNN regression. The latter is thousands of times faster than the former, thus the high computational efficiency of the data-driven DNN is verified.

Note that in Algorithm 1, each price is evaluated by a weighted reward. The value of the weight factor α will affect the eventual price selection. Fig. 7 demonstrates the optimal price setting obtained by the Monte Carlo method with different weight factors. Fig. 8 and Fig. 9 compare the total profit and PAR under different weight factors. More detailed explanations of Fig. 7-Fig. 9 are shown as follows.



(a) 10,000 samples (b) 20,000 samples

(c) 50,000 samples (d) 80,000 samples

Fig. 7. Optimal price setting under different weight factors

In Fig. 7, it can be observed that with a larger weight factor, the DSO intends to increase the hourly retail price. For example, in all the subfigures at hour 20, as α increases from 0 to 1, the hourly retail price goes from green, which stands for a lower price value, to bright yellow, which stands for a higher price value. This is because an increasing weight factor implies that the DSO weighs the profit of selling power more than the PAR, as shown in (10). The DSO intends to raise the price to achieve a higher profit.

Fig. 8 and Fig. 9 demonstrate the DSO's profit of selling power and the PAR under the specific weight factor. The profit and PAR shown in the figures are obtained by sending the selected price sequence to the individual microgrid model (1)-(9), and to calculate their aggregated power exchange. The DNN is not used here because we only need to test the selected price sequence, and the conventional model will provide an accurate result. As seen in the figures, a growing weight parameter leads to higher profit and higher PAR. For example, when α is 0.1, the optimal profit of selling power obtained based on 10,000, 20,000, 50,000, and 80,000 samples are $609, $626, $663, and $651, respectively, and the optimal PAR are all 1.0686,1.0744,1.0744,1.0744; when α is 0.7, the optimal profit of selling power obtained based on 10,000, 20,000, 50,000, and 80,000 samples are $681,$679,$682, and $682, respectively, and the optimal PAR are 1.082,1.0777,1.0840, and 1.0840, respectively. This is because with an increasing weight factor, the DSO values the total profit more than PAR, and tends to increase the hourly retail price, which has already been discussed in Fig. 7. An increasing price level drives microgrids to shift more of their load to hours with relatively lower prices, which exacerbates the peak to valley distance, and increases PAR. Therefore, the DSO needs to make a trade-off between gaining more profit and maintaining a smooth load profile.

It can also be observed from Fig. 7-Fig. 9 that the results based on larger sizes of samples (i.e., 5×10^4 and 8×10^4) don't show much difference. Hence, we can assume that such sample sizes are large enough for the Monte Carlo method to find the optimal solution.

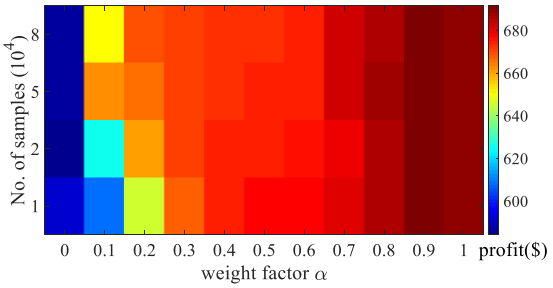


Fig. 8. Total profit ($) under different weight factors

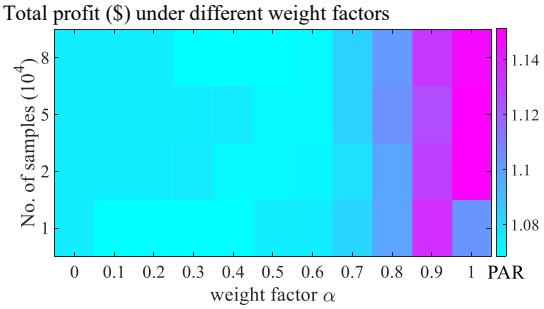
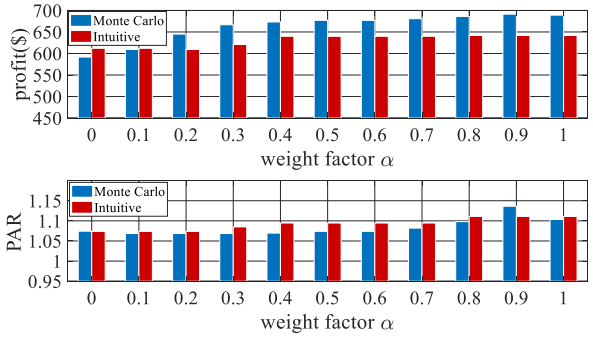
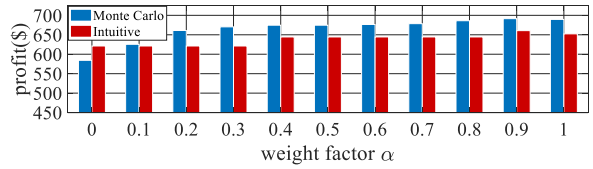


Fig. 9. Final PAR under different weight factors

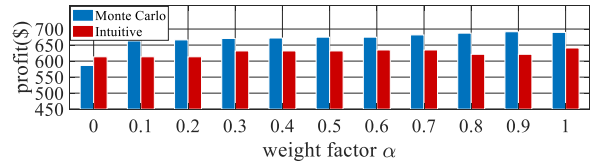
A final conclusion that can be drawn from Fig. 8 and Fig. 9 concerns the optimal value of the weight factor. It can be observed from Fig. 9 that as α increases from 0 to 0.7, the PAR increases considerably slow, while the profit of selling power keeps growing. When α is greater than 0.7, the PAR shows obvious increase. Hence, the DSO is recommended to set the weight factor to 0.7 to maximize the profit of selling power, while maintaining a considerably low PAR.



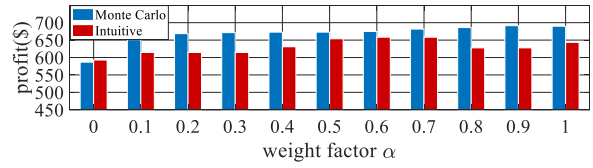
(a) 10,000 samples



(b) 20,000 samples



(c) 50,000 samples



(d) 80,000 samples

Fig. 10. Comparison of Monte Carlo method and intuitive method

As stated in Section IV-C, the Monte Carlo method regroups the prices from different price sequence samples instead of intuitively choosing the price sequence with the largest

weighted reward. To verify the merit of the Monte Carlo method, a comparison with the intuitive method is shown in Fig. 10. As can be observed in the figure, with the change of weight factor, the Monte Carlo method is able to achieve higher profit of selling power and lower PAR than the intuitive method. For example, in subfigure (d), when α is 0.6, the profits of selling power obtained from the Monte Carlo method and the intuitive method are $675 and $658, respectively; and the PARs are 1.0705 and 1.1003, respectively. This is because the Monte Carlo method has a strong exploration ability to discover new price sequences by regrouping the existing price samples, which can lead to better solutions; while the intuitive method only relies on the existing samples, which can be stuck to local optimum.

VI. Conclusions

In this paper, a novel data-driven method is proposed for the multi-microgrid (MMG) energy management problem. First, a deep neural network is constructed to simulate MMG operation under dynamic retail price signals, with no requirement of local generation or consumption information, which protects customer privacy. Second, the DSO applies a model-free Monte Carlo reinforcement learning method to optimize its pricing strategy, with the aim of maximizing profit of selling power and minimizing PAR. Simulation results demonstrate that the DNN regression model has considerable accuracy due to its automatic feature extraction ability. It also outshines the conventional model-based method in computational efficiency with its high generalization. Compared with an intuitive selection method, the Monte Carlo method proves to have strong exploration ability in problems with no explicit mathematical formulations or with high computation complexity. The combination of the proposed data-driven deep neural network and the Monte Carlo method can be a promising tool for studying power system problems with hidden information or vast search spaces in future researches.

References

[1] Y. Du, F. Li, J. Li, and T. Zheng, "Achieving 100x Acceleration for N-1 Contingency Screening with Uncertain Scenarios using Deep Convolutional Neural Network," *IEEE Trans. Power Syst.*, vol. 34, pp. 3303-3305, Jul. 2019.

[2] X. Fang, Q. Hu, F. Li, B. Wang, and Y. Li, "Coupon-based demand response considering wind power uncertainty: A strategic bidding model for load serving entities,". *IEEE Trans. Power Syst.*, vol. 31, pp. 1025-1037, Mar. 2015.

[3] Y. Du, and F. Li, "A Hierarchical Real-time Balancing Market Considering Multi-microgrids with Distributed Sustainable Resources," *IEEE Trans. Sustainable Energy*, 2018, early access.

[4] J. P. Catalão, P. Siano, F. Li, M. A. Masoum, and J. Aghaei, "Guest Editorial Special Section on Industrial and Commercial Demand Response," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 5017-5019, Nov. 2018.

[5] H. Shin, and R. Baldick, "Plug-in electric vehicle to home (V2H) operation under a grid outage," *IEEE Trans. Smart Grid*, vol. 8, pp. 2032-2041, Jul. 2017.

[6] A.-H. Mohsenian-Rad, V. W. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Trans. Smart Grid*, vol. 1, pp. 320-331, Nov. 2010.

[7] R. Deng, Z. Yang, J. Chen, N. R. Asr, and M.-Y. Chow, "Residential Energy Consumption Scheduling: A Coupled-Constraint Game Approach," *IEEE Trans. Smart Grid*, vol. 5, pp. 1340-1350, May 2014.

[8] H. M. Soliman, and A. Leon-Garcia, "Game-Theoretic Demand-Side Management With Storage Devices for the Future Smart Grid," *IEEE Trans. Smart Grid*, vol. 5, pp. 1475-1485, May 2014.

[9] P. Samadi, H. Mohsenian-Rad, V. W. S. Wong, and R. Schober, "Real-Time Pricing for Demand Response Based on Stochastic Approximation," *IEEE Trans. Smart Grid*, vol. 5, pp. 789-798, Mar. 2014.

[10] C. Li, X. Yu, W. Yu, G. Chen, and J. Wang, "Efficient Computation for Sparse Load Shifting in Demand Side Management," *IEEE Trans. Smart Grid*, vol. 8, pp. 250-261, Jan. 2017.

[11] S. Bahrami, V. W. S. Wong, and J. Huang, "An Online Learning Algorithm for Demand Response in Smart Grid," *IEEE Trans. Smart Grid*, vol. 9, pp. 4712-4725, Sep. 2018.

[12] L. Wang, Z. Zhang, and J. Chen. "Short-term electricity price forecasting with stacked denoising autoencoders," *IEEE Trans. Power Syst.*, vol. 32, pp. 2673-2681, Jul. 2017.

[13] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, "Short-Term Residential Load Forecasting based on Resident Behaviour Learning," *IEEE Trans. Power Syst*., vol. 3, pp. 1087-1088, Jan. 2018.

[14] J. Yan, H. Zhang, Y. Liu, S. Han, L. Li, and Z. Lu, "Forecasting the High Penetration of Wind Power on Multiple Scales Using Multi-to-Multi Mapping," *IEEE Trans. Power Syst.*, vol. 33, pp. 3276-3284, May 2018.

[15] Y. Wang, Q. Chen, D. Gan, J. Yang, D. S. Kirschen, and C. Kang, "Deep Learning-Based Socio-demographic Information Identification from Smart Meter Data," *IEEE Trans. Smart Grid*, early access.

[16] Y. Chen, Y. Wang, D. Kirschen, and B. Zhang, "Model-free renewable scenario generation using generative adversarial networks," *IEEE Trans. Power Syst.*, vol. 33, pp. 3265-3275, May. 2018.

[17] P. Zeng, H. Li, H. He, and S. Li, "Dynamic Energy Management of a Microgrid using Approximate Dynamic Programming and Deep Recurrent Neural Network Learning," *IEEE Trans. Smart Grid*, early access.

[18] D. O'Neill, M. Levorato, A. Goldsmith, and U. Mitra, "Residential demand response using reinforcement learning," in *2010 First IEEE International Conference on Smart Grid Communications*, 2010, pp. 409-414.

[19] Z. Wen, D. O'Neill, and H. Maei, "Optimal Demand Response Using Device-Based Reinforcement Learning," *IEEE Trans. Smart Grid*, vol. 6, pp. 2312-2324, Sep. 2015.

[20] A. Sheikhi, M. Rayati, and A. M. Ranjbar, "Dynamic load management for a residential customer; reinforcement learning approach," *Sustainable Cities and Society*, vol. 24, pp. 42-51, Jul. 2016.

[21] B. J. Claessens, P. Vrancx, and F. Ruelens, "Convolutional Neural Networks for Automatic State-Time Feature Extraction in Reinforcement Learning Applied to Residential Load Control," *IEEE Trans. Smart Grid*, vol. 9, pp. 3259-3269, Jul. 2018.

[22] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, and J. G. Slootweg, "On-line Building Energy Optimization using Deep Reinforcement Learning," *IEEE Trans. Smart Grid*, early access.

[23] Z. Wan, H. Li, H. He, and D. Prokhorov, "Model-Free Real-Time EV Charging Scheduling Based on Deep Reinforcement Learning," *IEEE Trans. Smart Grid*, early access.

[24] Z. Yan, and Y. Xu, "Data-driven Load Frequency Control for Stochastic Power Systems: A Deep Reinforcement Learning Method with Continuous Action Search," *IEEE Trans. Power Syst.*, early access.

[25] F. Li and Y. Du, "From AlphaGo to Power System AI: What Engineers Can Learn from Solving the Most Complex Board Game," IEEE Power & Energy Magazine, vol. 16, pp. 76-84, Mar. 2018.

[26] M. Parvania, and M. Fotuhi-Firuzabad, "Demand response scheduling by stochastic SCUC," *IEEE Trans. Smart Grid*, vol. 1, pp. 89-98, Jun. 2010.

[27] W. Liu, P. Zhuang, H. Liang, J. Peng, Z. Huang, "Distributed Economic Dispatch in Microgrids Based on Cooperative Reinforcement Learning," *IEEE Trans Neur Net Learn Syst.*, vol. 29, pp. 2192-2203, Jun. 2018.

[28] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," Cambridge: MIT press, 2016, pp. 152 - 231.

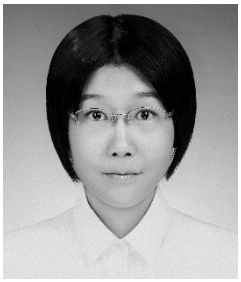
[29] R. S. Sutton, and A. G. Barto, "Reinforcement Learning: An Introduction," Cambridge: MIT press, 2018.

[30] H. Yuan, F. Li, Y. Wei, and J. Zhu, "Novel Linearized Power Flow and Linearized OPF Models for Active Distribution Networks with Application in Distribution LMP," *IEEE Trans. Smart Grid*, vol. 9, pp. 438-448, Jan. 2018.

[31] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980, 2014.

[32] X. Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

[33] S. Ioffe, C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” arXiv preprint, arXiv:1502.03167, 2015.



Yan Du (S’16) received the B.S. degree from Tianjin University, Tianjin, China, and the M.S. degree from Institute of Electrical Engineering, Chinese Academy of Sciences, Beijing, China, in 2013, and 2016, respectively. She is currently working toward the Ph.D. degree at the University of Tennessee, Knoxville, TN, USA. Her research interests include distribution system operation, and deep learning in power systems.

Fangxing Li (S’98–M’01–SM’05-F’17) is also known as Fran Li. He received the B.S.E.E. and M.S.E.E. degrees from Southeast University, Nanjing, China, in 1994 and 1997, respectively, and the Ph.D. degree from Virginia Tech, Blacksburg, VA, USA, in 2001.

Currently, he is a James McConnell Professor with the University of Tennessee, Knoxville, TN, USA. His research interests include renewable energy integration, demand response, power markets, power system control, and power system computing.