Energy 238 (2022) 121873

Contents lists available at ScienceDirect

Energy

journal homepage: www.elsevier.com/locate/energy

Real-time optimal energy management of microgrid with uncertainties based on deep reinforcement learning

Chenyu Guo, Xin Wang*, Yihui Zheng, Feng Zhang

Center of Electrical and Electronic Technology, Shanghai Jiao Tong University, Shanghai, 200240, China

ARTICLE INFO

Article history:
Received 18 March 2021
Received in revised form
11 August 2021
Accepted 21 August 2021
Available online 25 August 2021

Keywords:
Microgrid
Optimal energy management
Uncertainties
Deep reinforcement learning

ABSTRACT

Microgrid (MG) is an effective way to integrate renewable energy into power system at the consumer side. In the MG, the energy management system (EMS) is necessary to be deployed to realize efficient utilization and stable operation. To help the EMS make optimal schedule decisions, we proposed a real-time dynamic optimal energy management (OEM) based on deep reinforcement learning (DRL) algorithm. Traditionally, the OEM problem is solved by mathematical programming (MP) or heuristic algorithms, which may lead to low computation accuracy or efficiency. While for the proposed DRL algorithm, the MG-OEM is formulated as a Markov decision process (MDP) considering environment uncertainties, and then solved by the PPO algorithm. The PPO is a novel policy-based DRL algorithm with continuous state and action spaces, which includes two phases: offline training and online operation. In the training process, the PPO can learn from historical data to capture the uncertainty characteristic of renewable energy generation and load consumption. Finally, the case study demonstrates the effectiveness and the computation efficiency of the proposed method.

1. Introduction

To deal with the shortage of fossil fuels and environmental pollution, renewable energy resources (RES) such as wind and solar energy earn widespread attention. However, the intermittent and volatility of renewable energy will affect the security and stability of power system [1]. Microgrid (MG) technology provides an effective way to utilize the distributed renewable energy (DRE). With the energy management system (EMS), the MG can maintain stable and efficient operation [2]. Besides, in grid-connected mode, the MG can trade energy with distribution network.

For MG-EMS, finding a proper strategy to schedule the dispatchable resources (including DGs, ESS, and controllable load) is an optimal problem. Numerous methods have been adopted to solve the MG optimal energy management (OEM) problem based on forecasted load and RES. The methods mainly include the metaheuristic optimization-based method, mathematical programming (MP) method (such as MILP, MINLP), control theory-based method, and machine learning-based method. The description of these methods is listed in Table 1. The metaheuristic methods (such as PSO [3]) are widely used in the no-linear optimal problem. But these methods can't guarantee the global optimum. Besides, metaheuristic methods can't store the optimization knowledge and reusing them for a new task, thus they easily lead to low search efficiency.

The MP [4] methods can be solved using commercial solvers directly once the mathematical model is formulated. If the full information about the system parameters and the exact prediction value of RES are given, MP can calculate the accurate solution of OEM. But that is not realistic in practice because of the uncertainty of renewable energy and load. To deal with this issue, two MP-based methods: robust optimization (RO) [5] and stochastic optimization (SO) [6] are widely applied to the OEM problem in day-ahead optimization. However, there still is a deviation between the day-ahead optimization and actual real-time operation. SP adopts a scenario-based method to model uncertainties, while RO represents them as uncertainty sets. However, the scale of stochastic optimization increases dramatically with the number of scenarios, resulting in a significant computational burden. Furthermore, although RO is less computationally demanding than SP, the obtained solution may be too conservative driven by the nature of RO in hedging against the worst-case realization of the uncertain parameters.

Some control theory-based methods (such as MPC, DP, and PID

* Corresponding author.
E-mail addresses: wangxin26@sjtu.edu.cn, wangxin26@tom.com (X. Wang).

https://doi.org/10.1016/j.energy.2021.121873
0360-5442/

Table 1
The description of methods for OEM problem.

Methods	Ref	Description
Metaheuristic optimization-based methods	[3]	Low search efficiency and local optimum.
Mathematical programming (MP) based methods	[4]	The exact solution can be got based on perfect prediction information. For a high-dimensional complicated problem, the computation time extends and an infeasible solution may be got.
Stochastic optimization (SP)	[5]	Genere a large number of uncertain scenarios based on probability distribution and calculate the expectation. The heavy computational burden for many scenarios to be considered.
Robust optimization (RO)	[6]	The solution is conservative because the worst case of uncertainty is considered.
Model predictive control (MPC)	[7]	Solve the optimization problem in the rolling horizon iteratively. The process is complicated and the optimization quality relies on the forecast accuracy of uncertain variables.
Dynamic Programming (DP)	[8]	Solve time series decisions by dynamic exploration to update state-action pair value at every step. Suffers curse of dimensionality when dealing with a system with a large number of state and action.
Proportional-integral-derivative (PID) control	[9]	Setting PID parameters is complicated work, especially for a system with too many participants.
Deep reinforcement learning (DRL)	[10–23]	Don't rely on a specific probability distribution model of the environment to handle uncertainty. After off-line training using historical data, the mothed can be directly applied to real-time dispatch. Deal with highly-dimensional data with high efficiency.

control) are also used to solve OEM problems. Some literature utilizes the rolling horizon optimization or model predictive control (MPC) [7] to tackle real-time OEM, which repeatedly optimizes the predictive model over a rolling period. But the high-quality optimization result relies heavily on the forecast accuracy which is difficult in the real-time decision. In Ref. [8], the Dynamic programming (DP) is used to OEM of grid-connected reversible solid oxide cell-based renewable MG. The DP method can be used in time series decisions by dynamic exploration to update state-action pair value at every step. However, the DP Needs to know the transition probabilities of the environment over time. Besides, it suffers a "curse of dimensionality" when dealing with a system with a large number of state and action. Ref. [9] presents a closed-loop PID (proportional-integral-derivative) controller-based price control method for autonomous and real-time balancing of energy demand and generation in smart grid electricity markets. But setting PID parameters is complicated work, especially for a system with too many participants. All the aforementioned methods are thought of as model-based methods, which need to formulate an accurate environment model.

Another challenge for these model-based methods is the computation cost. With devices in MG increasing, more and more data need to be processed and computed. As a result, the computational burden will increase accordingly when using model-based methods. And the complex constraints are even difficult to satisfy simultaneously. Besides, for multi-time sequential decision problem, the model-based methods have to resolve the problem at each time slot, which may hinder real-time decision. To deal with the above limitations, deep reinforcement learning (DRL) is applied to solve the MG OEM problem considering uncertainties. Recently model-free DRL methods have achieved great success in optimal decision-making problems, which do not need environment model information [10]. DRL can solve optimization problems effectively with its powerful perception ability of deep learning (DL) and the decision-making ability of reinforcement learning (RL) [11].

In the DRL decision-making process, the EMS smart agent, who is responsible for making decisions, interacts with the environment sequentially to choose the optimal strategy and maximize reward feedback from the environment. The DRL agent employs a deep neural network (DNN) to learn strategy. It can acquire the feature from a massive amount of historical data and overcome the curse of dimensionality shortcomings, suitable for handling large-scale problems. After the DNN trained at offline phase, it can be utilized in the whole optimization period with no need for repetitive training. Besides, the DRL takes into account the future cumulative reward, not only the immediate payback at the current time slot. While the model-based methods only consider immediate payback. If there are time-coupling variables, the model-based methods can't get the globally optimal result.

The DRL has been widely used in power system fields, such as demand response [12], hybrid electric vehicle energy management [13], wind short-term forecasting [14]. The application of DRL in EMS is also a hot field. Ref. [15] applies Q-learning to MG energy management to plan the battery scheduling. Ref. [16] solves the wind farm management with energy storage based on the DQN algorithm considering uncertainties. Ref. [17] proposes a DRL-based DDPG algorithm to realize IES dynamic energy dispatch. Ref. [18] uses DDPG to find an optimal control strategy of battery in MG. However, research in this field is still in the early stage, and further studies are needed. The Q-learning learns policy by updating the state-action pair table (Q-table) [19]. The Deep Q-Network (DQN) [20] can only solve discrete action domains and choose actions from finite discrete values. While the deep deterministic policy gradient (DDPG) [21] is suitable for the problem with high dimension and continuous action spaces. But the effectiveness of learning relies on the selection of hyperparameters. A lot of effort needs to tune the right hyperparameters. Ref. [22] applies SAC to building energy management to manage cooling and domestic hot water storage using an open-source simulation environment. However, the impact of environmental uncertainty is not analyzed in more depth. Ref. [23] solves the off-grid microgrid lifelong control problem with the DRL method. But the used DRL method is based on discrete action space and doesn't consider action constraints.

The proximal policy optimization (PPO) [24] is a new policy-based DRL algorithm, which is less sensitive to the hyperparameters and can avoid the large policy update with undesirable action selection. In this paper, the PPO-based real-time OEM strategy is proposed. The main contribution of this paper is summarized as follows:

(1) We formulate a real-time optimal energy management model considering uncertainty and map it into the Markov decision process (MDP). Under the MDP, the environmental uncertainties are reflected and the reward function is designed to reflect the objective and constraints.
(2) A novel DRL algorithm: proximal policy optimization (PPO) is adopted to solve the OEM problem. The PPO algorithm utilizes the long-term historical data of energy consumption and renewable energy generation to update the network and learn optimal policy. The clipped surrogate loss function is

adopted to keep the action selection more efficient and steady.

(3) The PPO algorithm performance is compared with other algorithms: DDPG, DQN, and SP. And the computational effectiveness at online operation with environmental variation is verified.

The rest of this paper is organized as follows. In Section 2, the OEM mathematical model and MDP framework is presented. In Section 3, the PPO algorithm is proposed. The numerical experiments are conducted in Section 4. Finally, the conclusion is given in Section 5.

2. The problem formulation

In this section, the MG energy management is described and an optimal mathematical model is built. and then, the mathematical model is formulated as Markov decision process (MDP). The elements of MDP for the OEM problem are defined.

The MG operates in a grid-connected mode, which consists of RESs (including photovoltaics system and wind turbine), controllable distributed resources (i.e., micro-turbine (MT), and energy storage system (ESS) [25]). The structure of MG is shown in Fig. 1. The EMS is deployed in the MG to maintain stable operation by scheduling the controllable devices and trading with the distribution network. The schedule decision is made based on the forecasted WT and PV output, load consumption, and transaction price.

2.1. Optimal energy management mathematical model

The objective of MG-EMS is to minimize the real-time operation cost at each time slot, including the cost of exchanging power, the operation cost of MT and ESS. The MT power output, charging (or discharging) status of ESSs, and the exchanging power are decided, based on the forecasted PV, WT output, electrical load, and electricity exchanging price. And the constraints include power balance, MT operation, and ESS operation constraints.

2.1.1. Objective function

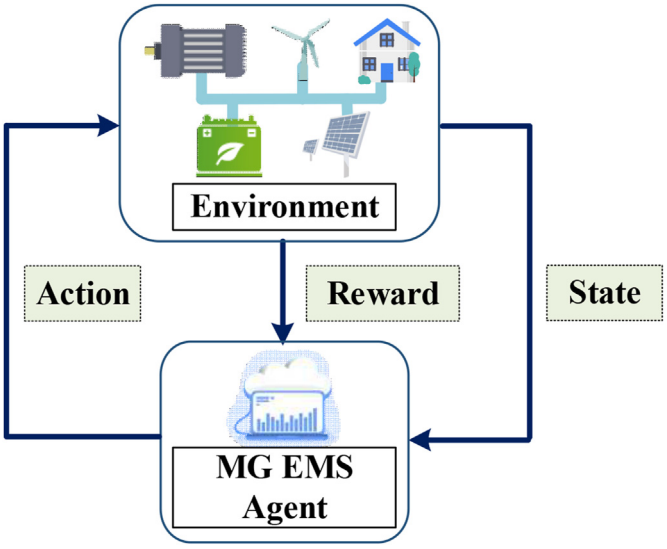

Fig. 1. The framework of MG OEM under the Markov decision process.

$$\min C_t^{MG} = C_t^{ex} + C_t^{MT} + C_t^{ESS} \tag{1}$$

$$C_t^{ex} = c_t^{ex} \cdot P_t^{ex} \cdot \Delta t \tag{2}$$

$$C_t^{MT} = \left(a + b \cdot P_t^{MT}\right) \cdot \Delta t \tag{3}$$

$$C_t^{ESS} = c^{ESS} \cdot \left(P_t^{cha} \eta_{cha} + P_t^{dis} \Big/ \eta_{dis}\right) \cdot \Delta t \tag{4}$$

where, C_t^{cost} is the total operation cost of the MG at t. C_t^{ex} is the cost of exchanging power with the DN, C_t^{MT} is the operation cost of MT, which can be formulated as a linear function [26], $C_t^{ESS}$is the operation cost of ESS [27]. P_t^{ex} is the exchanging power with distribution network (DN), $P_t^{ex} > 0$, if power is sold to MG, otherwise $P_t^{ex} < 0$, c_t^{ex} is the electricity price of exchanging power with DN. P_t^{MT} is the MT power output, a, b are the MT operation coefficients; c^{ESS} is cost coefficients of ESS, P_t^{cha}, P_t^{dis} are charging and discharging power, η^{cha}, η^{dis} are the efficiency of the charging and discharging process respectively.

2.1.2. Constraints

$$P_t^{MT} + P_t^{PV} + P_t^{WT} - P_t^{cha} + P_t^{dis} - P_t^{load} + P_t^{MG} = 0 \tag{5}$$

$$P_{\min}^{MT} \le P_t^{MT} \le P_{\max}^{MT} \tag{6}$$

$$P_{\min}^{ESS} \le P_t^{cha}, P_t^{dis} \le P_{\max}^{ESS} \tag{7}$$

$$E_{t+1}^{ESS} = E_t^{ESS} + \eta_{cha} P_t^{cha} \Delta t - \eta_{dis} P_t^{dis} \Delta t \tag{8}$$

$$SOC_{\min} \le E_t^{ESS} \Big/ E_{\max}^{ES} \le SOC_{\max} \tag{9}$$

$$-P_{\max}^{ex} \le P_t^{ex} \le P_{\max}^{ex} \tag{10}$$

Eq. (5) is the power balance constraint, Eq. (6) is the MT output power constraint. Eq. (7) is the ESS charging and discharging power constraint; Eq. (8) represents the relationship of energy between t+1 and t time slot, Eq. (9) indicates the allowed operation range of the state of charge (SOC). Eq. (10) is the exchange power constraints. $P_{\min}^{MT}$, $P_{\max}^{MT}$ are the minimum and maximum of MT output respectively. E_t^{ESS} is the energy in the ESS at t, $E_{\max}^{ESS}$ is the capacity of ESS. P_{max}^{ex} is the maximum of power exchanged with DN. The power loss of feeder line within the MG is neglected.

2.2. Markov decision process (MDP) formulation

The MG-EMS smart center is taken as an agent, its optimal decision process in an uncertain environment can be formulated as a Markov decision process (MDP). The DRL can be described as a Markov decision process (MDP). MDP is a discrete-time stochastic control process and provides a mathematical framework for modeling decision making in a situation where outcomes are partly random and partly under the control of a decision. Under MDP formulation, the randomness of the environment can be taken into account. The MDP is denoted as a 4-tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R})$ [28]. $\mathscr{S}$ is the set of environmental states observed by the agent. $\mathscr{A}$ is the set of actions the agent can take. $\mathscr{P}: \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to \mathbb{R}$ is the state transition kernel which indicates the mapping between states at

two adjacent time slots. This transition probability reflects the system uncertainties. $\mathscr{R}:\mathscr{S}\times\mathscr{A}\times\mathscr{S}\to\mathbb{R}$ is the immediate reward.

At each time slot within a finite time horizon, the agent observes the environment state $s_t\in\mathscr{S}$, and then selects and executes an action $a_t\in\mathscr{A}$ according to the policy $\pi:\mathscr{S}\to\mathscr{A}$. As a result, the agent gets the immediate reward r_t of the selected action, and the environment state transfers to the next state s_{t+1} from s_t. The aim of the DRL algorithm is to find the optimal policy $\pi^*(a_t|s_t)$ to maximize cumulative long-term rewards.

The immediate reward r_t only reflects the influence of action at current times. The cumulative long-term effect of the policy π at the state s_t is defined as a cumulative reward R_t.

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{k=t}^{T} \gamma^{k-t} r_k \tag{11}$$

where $\gamma\in[0,1]$ indicates the discount factor, which balances the current and future received rewards. The optimal policy π^* that can maximize cumulative long-term reward R_t.

Then, the MG OEM problem is mapped to MDP formulation. The state action and reward of the OEM-MDP are shown as follows.

State:

For the MG agent, the environmental state information includes the WT output, PV output, load consumption, exchanging price, and the SOC of ESS. Among them, the first four are exogenous variables with uncertainties, which can't be controlled by the agent. While the last one is the inner state variable which is affected by the agent's action. Thus, the state of MG is described as follows.

$$s_t = \left\{P_t^{WT}, P_t^{PV}, P_t^{load}, c_t^{ex}, SOC_t\right\} \tag{12}$$

Action:

The OEM decision variable includes $P_t^{MT}, P_t^{dis}, P_t^{cha}, P_t^{ex}$. To simplify the variables, we use one variable $P_{i,t}^{ESS}$ to represent the charging and discharging action at time t. If $P_t^{ESS}\geq 0$, $P_t^{dis} = P_t^{ESS}, P_t^{cha} = 0$; else $P_{,t}^{cha} = -P_t^{ESS}$, $P_t^{dis} = 0$. According to the power balance constraint (5), the exchange power can be directly calculated once P_t^{MT} and P_t^{ESS} is decided. Therefore, the simplified action space is listed as follows.

$$a_t = \left\{P_t^{MT}, P_t^{ESS}\right\} \tag{13}$$

Reward:

The designed reward function $r_t = \mathscr{R}(s_t, a_t, s_{t+1})$ must reflect the agent's optimal objective and lead the agent to learn the action without constraints violation. So the MG real-time operation cost should be included in the reward function. Besides, the practical operation constraints of OEM should be taken into account and the action violating constraints isn't allowed in real-time operation. Thus the penalty function C_t^{pen} is added to the reward function. Based on these requirements, the immediate reward function at each time slot of the MG agent is defined as follows.

$$r_t = -\left(C_t^{cost} + C_t^{pen}\right) + r_t^{ex} \tag{14}$$

where r_t^{ex} is the extra reward if all the constraints are satisfied. For a variable x_k with the boundary $[x_k^{min}, x_k^{max}]$, its penalty function is as follows.

$$penal_k = \ln\frac{\left|x_k - x_k^{\max}\right| + \left|x_k - x_k^{\min}\right|}{2\left(x_k^{\max} - x_k^{\min}\right)} \tag{15}$$

For constraints (9), (10), the penalty functions are defined as $penal^{SOC}$ and $penal^{exch}$ respectively. The total penalty function is as follows.

$$C_t^{pen} = \left(\upsilon_1 penal^{SOC} + \upsilon_2 penal^{exch}\right) \tag{16}$$

where, ν_1, ν_2 are the penalty coefficients.

3. Deep reinforcement learning algorithm

In this section, the deep reinforcement learning (DRL) algorithm: Proximal policy optimization (PPO) is adopted to find the optimal strategy of the OEM problem under MDP framework. The framework of the PPO algorithm training process is shown in Fig. 2.

3.1. Proximal policy optimization (PPO)

In the DRL, the agent is a kind of Deep Neural Network (DNN). The DNN consists of the input layer, output layer, and several hidden layers. All the layers are fully connected neural networks with the parameters θ (matrix of weights and vector of biases). The state information of the environment is fed to the input layer of DNN. And its output is the action or action value. According to whether outputting action directly or not, the DRL can be classified into the policy-based method (such as DDPG) and value-based method (such as deep Q-network). In this paper, we adopt a novel policy-based DRL algorithm: proximal policy optimization (PPO) to solve the OEM problem. The PPO algorithm is a policy-based DRL algorithm with actor-critic architecture. The actor and critic are two deep neural networks (DNN) with parameters θ^μ and θ^Q respectively. The actor network μ is used to estimate the policy function $\pi(a|s,\theta^\mu)$. While the critic network Q is used to estimate the value function $V(s)$ The actor policy $\pi(a|s,\theta^\mu)$is a normal distribution, $\pi(a|s)\sim\mathscr{N}(\mu,\sigma^2)$. The actor outputs the mean μ and standard deviation σ. When executing the policy, the specific action is sampled from the normal distribution. The structure of Actor and Critic network is shown Fig. 3.

Since the parameters between the actor and the critic networks are not shared, they are updated respectively. In the training process, the actor network μ outputs the continuous actions based on the environmental state input. Then, the critic network will map state to a scalar $V(s)$ to measure the quality of the state. The agent explores the environment periodically in the MDP and the parameters of DNN are updated every episode. At the end of each episode with length T, a sequence of states, actions, and rewards within the episode constitutes the trajectories $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_t, a_t, r_t, \ldots, s_T, a_T, r_T\}$, which are considered as the collected past experiences. For the MG-OEM problem, the length of the episode is 24 h.

The aim of training actor network with parameter θ^μ is to maximize the accumulated reward of the agent under policy π, i.e. $\max\mathbb{E}_{\pi_{\theta^\mu}}[R_t]$. The parameters of actor network θ^μ are updated based on the policy gradient method. The policy gradient algorithms rely on sampled decision sequences when interacting with the environment and apply gradient ascent to update the policy parameters. The sampled policy loss function and its gradient are shown as follows [29]. The $\nabla_{\theta^\mu}L(\theta^\mu)$ can improve its strategy in the direction of increasing the probability of action with a greater return.

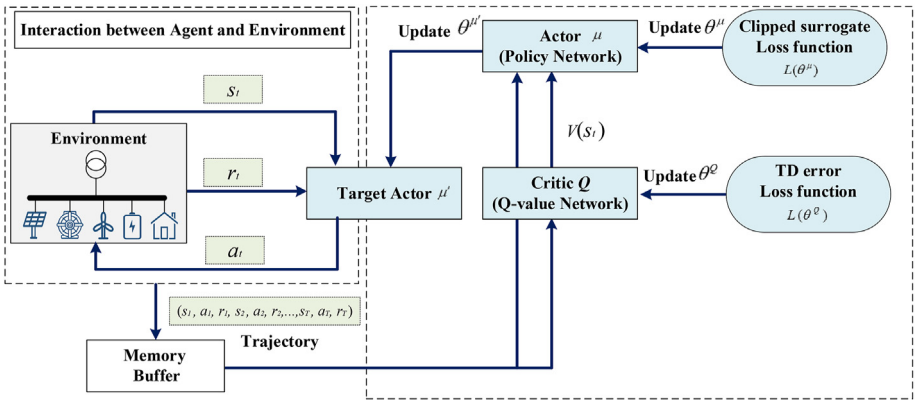

Fig. 2. The framework of the PPO algorithm training process.

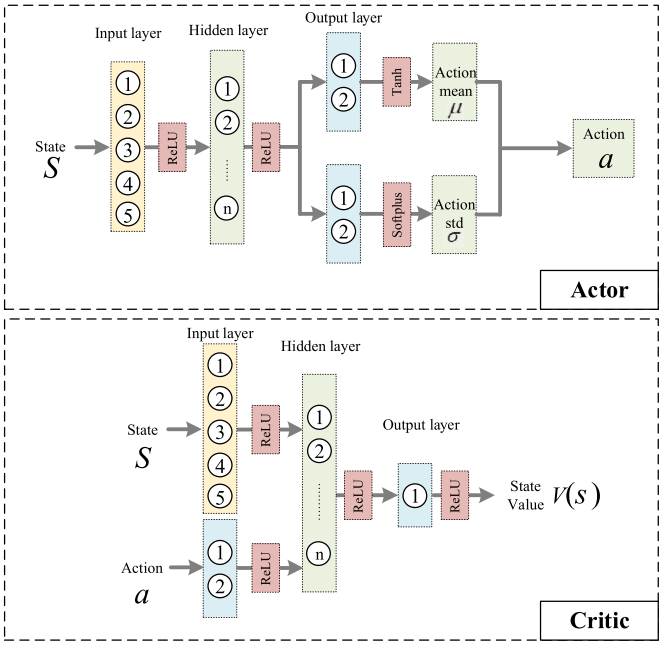

Fig. 3. The Deep Neural Network structure of Actor and Critic.

$$L(\theta^{\mu}) = \mathbb{E}_{\pi_{\theta^{\mu}}}[R_t] \tag{17}$$

$$\nabla_{\theta^{\mu}} L(\theta^{\mu}) = \mathbb{E}_{\pi_{\theta^{\mu}}}[R_t \nabla_{\theta^{\mu}} log\pi_{\theta^{\mu}}(a|s)] \tag{18}$$

Though it might be straightforward to execute multiple steps of optimization on this loss function, there are some challenges from the sample inefficiency, the balance between exploration and exploitation, and the high variance of the learned policy. It may result in large policy updates, which will affect the state and reward distribution in the following steps.

The actor-critic architecture is adopted to make a significant impact on reducing the variance of the gradient estimates by reformulating the reward signals in terms of advantage. So we defined the advantage function $A(s,a)$ to expresses how good the selection action a is compared with all the other available actions [30]. If the selected action a is better than average, the advantage function $A(s,a)$ is positive; otherwise, it is negative.

$$A(s,a) = Q(s,a) - V(a) \tag{19}$$

The action-value (Q-value) function $Q(s,a) : \mathscr{A} \times \mathscr{S} \to \mathbb{R}$ is used to evaluate the performance of a policy π:$\mathscr{A} \times \mathscr{S} \to \mathbb{R}$, which is defined as follows.

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a, \right. \\ \left. a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathscr{P}(\cdot|s_t, a_t)\right] \tag{20}$$

Correspondingly, the state-value function $V(s) : \mathscr{S} \to \mathbb{R}$ of a policy, which evaluates the quality of the state, is defined as follows.

$$V(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_t \sim \pi(\cdot|s_t), \right. \\ \left. s_{t+1} \sim \mathscr{P}(\cdot|s_t, a_t)\right] \tag{21}$$

It can be seen from the above definition, the value function $V(s)$ is the expected long term reward at the state s and following the policy π, which estimates the average reward of the state. The $Q(s,a)$ indicates the long-term reward of taking specific action at the state s and following the policy π.

In fact, the actor network can be updated at each time step. However, the policy gradient method needs to resample from the new policy after being optimized at every step, because the old samples are not suitable for the new policy. This will increase training time. To solve this problem, another actor network μ' is added to the PPO algorithm, which is used to interact with the environment to obtain trajectories. The actor network μ' is called the old actor, with parameters $\theta^{\mu'}$ unchanged during one episode. And the actor network μ (also called new actor) can learn policy several times using the trajectories.

So the actor network μ of PPO is updated using a clipped surrogate loss function as follows.

$$L^{CLIP}(\theta^{\mu}) = \mathbb{E}\left[min\left(\rho(\theta)\widehat{A}^{\theta^{\mu'}}(s_t,a_t), clip(\rho(\theta), 1-\varepsilon, 1+\varepsilon)\widehat{A}^{\theta^{\mu'}}(s_t,a_t)\right)\right] \tag{22}$$

$$clip(\rho(\theta), 1-\varepsilon, 1+\varepsilon) = \begin{cases} 1-\varepsilon & \rho(\theta) < 1-\varepsilon \\ 1+\varepsilon & \rho(\theta) > 1+\varepsilon \\ \rho(\theta) & otherwise \end{cases} \tag{23}$$

$$\rho(\theta) = \frac{\pi_{\theta^{\mu}}(a_t|s_t)}{\pi_{\theta^{\mu'}}(a_t|s_t)} \tag{24}$$

$$\theta^{\mu} = \theta^{\mu} + \alpha^{A} \cdot \nabla_{\theta^{\mu}} L^{CLIP}(\theta^{\mu}) \tag{25}$$

where ε $(0 < \varepsilon < 1)$ is the core parameter, which is applied to limit probability difference, $\widehat{A}^{\theta^{\mu'}}$ is the estimate of the advantage function, α^{A} is learning rate of the actor μ, and $\rho(\theta)$ is the probability ratio between μ and μ', which measures the difference between the two policies.

As the actor μ is updated in the training process, the old and new policy will diverge, increasing the variance of the estimation. The old policy is therefore periodically updated to match the new policy. If the $\rho(\theta)$ falls outside the range $[1-\varepsilon, 1+\varepsilon]$, the advantage function will be clipped, which can help to guarantee the stability of this algorithm. The algorithm is neither too greedy in favoring actions with positive advantage, nor too quick to avoid actions with a negative advantage from a small set of samples. The minimum operator ensures that the surrogate objective function is a lower bound of the unclipped objective.

A truncated version of generalized advantage estimation (GAE) is adopted, which is shown as follows.

$$\widehat{A}(s_t, a_t) = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{26}$$

where. $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

The critic network is updated by minimizing loss function based on temporal-different (TD) theory.

$$L\left(\theta^{Q}\right) = \mathbb{E}\left[\left(q_t - V(s_t)\right)^2\right] \tag{27}$$

$$q_t = r_t + \gamma V(s_{t+1}) \tag{28}$$

$$\theta^{Q} = \theta^{Q} + \alpha^{Q} \cdot \nabla_{\theta^{Q}} L\left(\theta^{Q}\right) \tag{29}$$

where $L(\theta^{Q})$ is the loss function. α^{Q} is the learning rate of critic network. In fact, the training of the critic is a process to minimize the difference between q_t and $V(s_t)$.

3.2. Training process

The PPO algorithm includes two phases: the off-line training phase and the online OEM operation phase. At the beginning, the parameters (weights and bias) of DNN are random, the output may not be optimal. Therefore, before being used to real-time online OEM, the DNN needs to be trained at the offline phase using historical environmental data. The parameters of DNN are updated through interacting with the environment repeatedly and accumulating experience. In this way, the DRL agent with DNN architecture can approximate the policy more precisely.

The pseudocode of the off-line training process is shown in algorithm 1. At the beginning of each episode, the environment state is reset. Then the old actor μ' is used to interact with the environment for T time steps and gets the trajectory. Next, the trajectory is used to train the actor and critic network for M times. After offline training, the well-trained parameters of DNN are preserved and applied to calculate the OEM result. The input of the PPO is the real-time environment information of the MG agent. Its output is the optimal strategy to MG OEM at every time step.

Algorithm 1

Off-line training process of PPO

Input: the state of environment
Output: the critic network $Q(s,a|\theta^{Q})$ and actor network$\mu(s|\theta^{\mu})$ with parameters θ^{Q} and θ^{μ}
Initialize parameters θ^{Q} and θ^{μ} randomly
Initialize old actor parameters: $\theta^{\mu'} \leftarrow \theta^{\mu}$
For episode $= 1{:}E$ **do:**
  Reset the initial state randomly
  For $t = 1{:}T$ **do:**
    Observe the state s_t
    Select action a_t using the old actor μ'
    Calculate reward $r_t$and obtain new state s_{t+1}
  End for
  Obtain trajectory τ and store in memory buffer
  For $m = 1{:}\ M$ **do:**
    Train critic network:
      Calculate cumulative reward R_t
        Map s_t to a scalar $V(s_t)$
        Calculate loss function $L(\theta^{Q})$ by (27)
        Update the critic parameters θ^{Q} by (29)
    Train actor network:
        Calculate $L^{CLIP}(\theta^{\mu})$ by (22)
        Update the new actor parameters θ^{μ} by (25)
        Update the old actor parameters by $\theta^{\mu'} \leftarrow \theta^{\mu}$
  End for
End for

4. Case study

4.1. Experimental setup

In the section, a MG test case is adopted to test the performance of the proposed algorithm. MG components include MT, one wind turbine, one photovoltaic unit, and a battery shortage. The parameters of MG components are listed in Table 2. The 100-day hourly data of load, WT, PV and price is used to train the DNN. The training data of load is come from Iowa distribution test systems [31], and the data of wind and solar is obtained from NERL Measurement and Instrumentation Data Center [32]. The wind speed and solar radiation and air temperature data are converted into power data according to Ref. [33]. The wholesale price is obtained from the California ISO [34].

Table 2
The MG components parameters

Parameter	value	Parameter	value
$P^{WT}_{\max}$	200kW	$E^{ESS}_{\max}$	200kWh
$P^{PV}_{\max}$	300kW	P^{ESS}_{max}	100kW
$P^{MT}_{\max}/P^{MT}_{\min}$	0/200kW	SOC_{min}/SOC_{mac}	20%/80%
a/b	0/0.045$/MW	η^{cha}/η^{dis}	95%/95%
P^{ex}_{max}	200kW	c^{ESS}	0.038$/MW

The hyperparameters of the PPO algorithms in this experiment are shown in Table 3. The extra reward is 20. The actor and critic DNN both have one hidden layer with 128 neurons in each layer. The input layer and hidden layer use ReLU activation function, and the output layer uses tahn and Softplus activation function for mean and std respectively. The simulation experiment is implemented using Python 3.6 with the deep learning platform Pytorch 1.4.0. The experiment is carried on the computer with Windows 10 system, AMD Ryzen7 3700X CPU @3.6 GHz, and 16G RAM.

The detailed analysis of setting some hyperparameters (including penalty coefficient, surrogate function clipping rate,

Table 3
The hyperparameters of PPO.

Hyperparameter	Value
Actor learning rate α^A	0.0005
Critic learning rate α^C	0.0005
Discount factorγ	0.9
Surrogate function clipping rate ε	0.2
GAE parameter λ	0.95
Maximum episode E	500
Number of updates new actor and critic M	5

learning rate) is given in Appendix A. The PPO is compared with two common DRL algorithms: DQN and DDPG. The algorithm pseudocode and hyperparameters of DQN and DDPG is shown in Appendix B.

4.2. Off-line training process

The training process of PPO, DDPG, and DQN is shown in Fig. 4. The full line is the moving average value of the cumulative reward. At the beginning, the agent has no knowledge about the environment and chooses action more randomly, so the reward has a large variation. After several interaction episodes, the experiences are accumulated and the DNN parameters is optimized. With the better policy is learned, the reward keeps increasing until the convergence is reached. It can be seen that the PPO algorithm converges after about 150 episodes, while the DDPG and DQN do after about 250 episodes. The PPO learns more efficiently than the other two to explore optimal policy. Besides, the PPO obtains a higher cumulative reward at the convergence. Even though the PPO reward is more fluctuant at the initial stage, it finally gets a higher value in the multi-dimensional continuous action space because of its more precise action choices. On the contrary, the DQN can only explore finite actions.

4.3. Online OEM results

To demonstrate the effectiveness of the proposed algorithm in online decision making, the well-trained DNN is adopted to the real-time OEM. The optimization is made at every time slot with a one-hour interval. A typical day state data is taken to test the performance of the algorithm. The aggregated 24-h power of typical MG load, wind, and PV profiles is presented in Fig. 5, and the trading price is shown in Fig. 6. The above data is considered as the real-time forecasted values. The operation results based on the PPO algorithm are shown in Fig. 7. The MG OEM operation results include exchanging power with DN, MT output, and ESS charging and discharging power under the given environment state. When the trading price is high, MT raises the output to supply the load demand with a relatively low operation cost. ESS units of the MG will charge power at off-peak intervals (0-2h,11-13h) under lower prices and discharge power during peak hours (20-22h) to reduce the cost of purchasing power. But if the load demand exceeds the MG supply capacity, it will purchase power from DSO. It can be concluded from the results that the MG can manage the dispatchable units to respond to the outside environment flexibly.

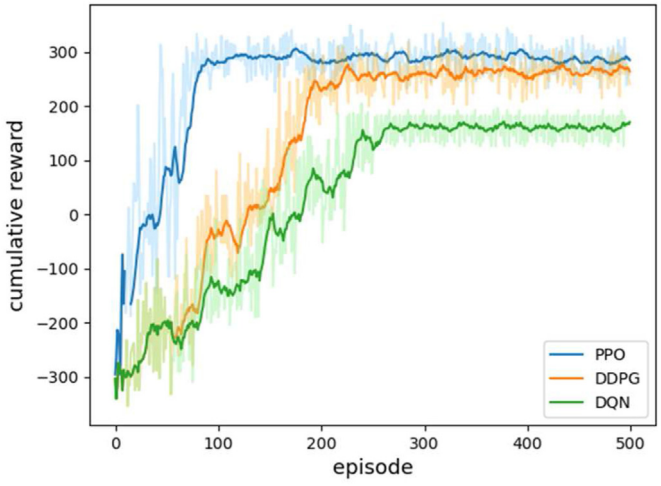

Fig. 4. The cumulative reward curve during the training process PPO, DDPG, and DQN.

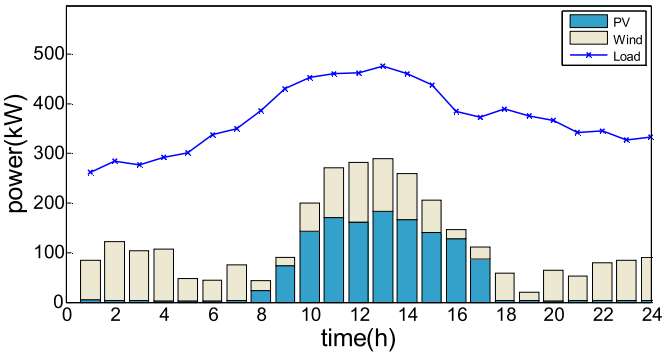

Fig. 5. The daily time series state information of Load, WT, and PV active power output.

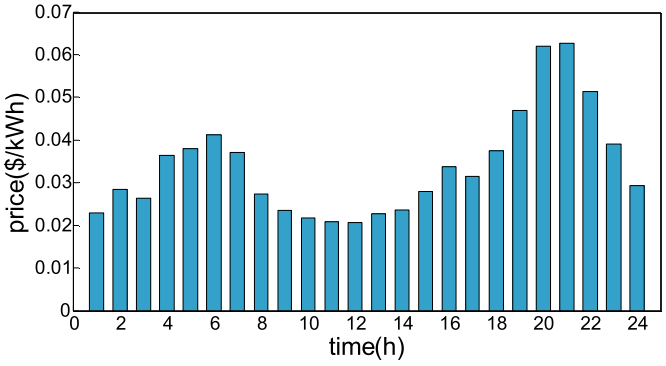

Fig. 6. The daily time series state information of exchanging price.

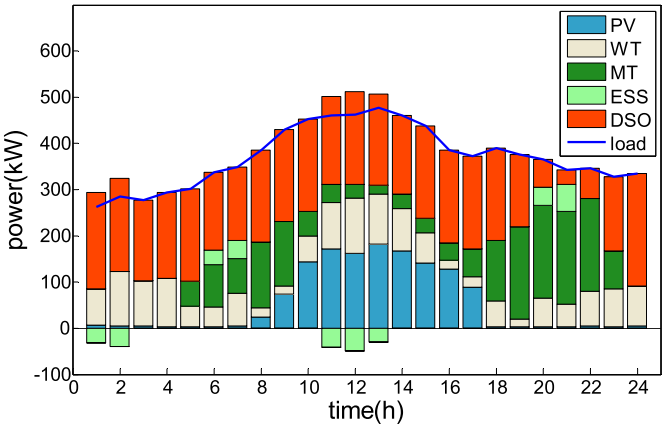

Fig. 7. The real-time OEM results of MGs based on the PPO algorithm.

4.4. Performance comparison

To further evaluate the performance of the PPO, we compare it versus DDPG, DDN, and the traditional MP method: stochastic programming (SP).

The SP is the existing common method to deal with uncertainty which makes an offline decision based on forecasted day-ahead information. In the scenario-based SP method, the WT and PV out are assumed to obey the normal distribution. At every SP scenario, the mathematical model of OEM in section 2.1 can be formulated as a deterministic MILP model and solved using commercial solver CPLEX. We sample 200 scenarios using Monte Carlo simulation based on the assumed probability density function of uncertain elements and reduce them to 50 scenarios. The Weibull distribution and beta distribution are using to describe the uncertainty model of wind speed and solar irradiation. The WT and PV generation of reduced scenarios are presented in Fig. 8 and Fig. 9 respectively.

Fig. 10 shows the comparison of the computation performance among the proposed PPO, DDPG, DDN, and SP on the test one-day case. We calculate the daily cumulative operation cost and record the computation time. It is obvious that these three DRL algorithms have a significant advantage in online operation time after offline training. That's because the SP has to solve the optimal model at every time slot with lots of scenarios, which challenges the memory capacity and computing speed in computers. On the contrary, the DRL-based algorithms can make decision immediately using the well-trained DNN without repetitive computation. So it can adapt the requirements for real-time operation better.

As for the operation cost, the SP method is higher than the DRL-based algorithms because of the difference between the assumed distributions and the actual uncertainty and the limited number of the sampled scenarios. While the model-free DRL-based method doesn't need to explicitly the probability distributions of renewable energy's uncertainties. Instead, the DRL learns strategy from the real-world historical data which explicitly encapsulates the uncertainty feature. And the DRL-based method considers long-term cumulative rewards for the whole period when making decisions, so that more likely getting higher profit.

What's more, the PPO can improve the training efficiency obviously in off-line, compared with DDPG and DQN. The DQN can learn the optimal strategy faster than DDPG when the dimension of action space is not quite high, since the DQN only needs to train one

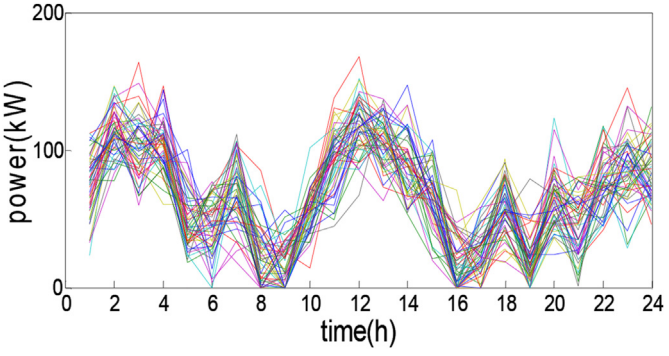

Fig. 8. WT generation for 50 scenarios.

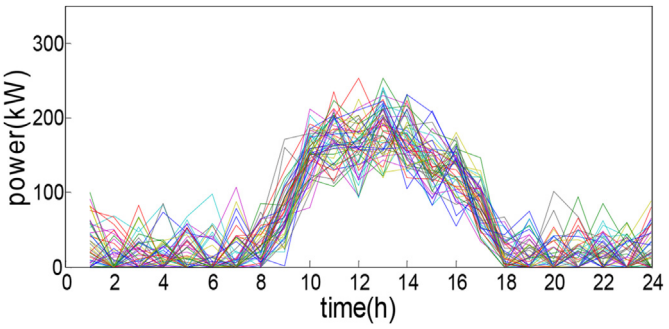

Fig. 9. PV generation for 50 scenarios.

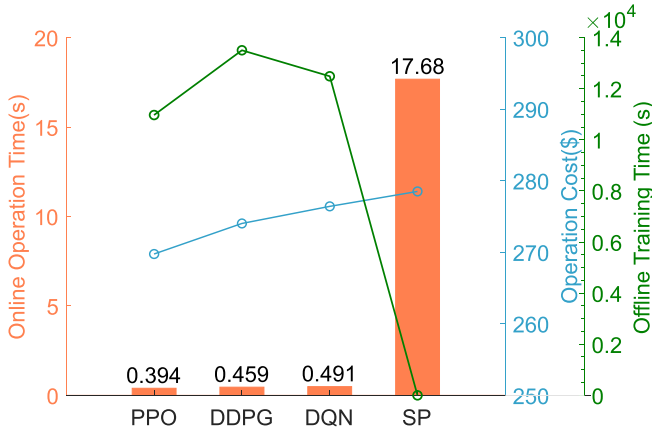

Fig. 10. The computation performance comparison of PPO, DDPG, DQN, and SP method.

network and search action with the finite action space. But it can't obtain precise action like policy-based DRL. The DDPG takes more time at the training process for updating two networks simultaneously and choosing action with larger gradient change. So, the PPO is more suitable for the problem with continuous high-dimension action and high precision requirements.

Besides, we compute the OEM operation cost at 10 test cases using these four methods and compare their error compared with the theoretical result based on the day-ahead MILP method. The error is defined as

$$error = \frac{cost^{rl} - cost^{MILP}}{cost^{MILP}} \times 100\% \tag{30}$$

where $cost^{rl}$ is the daily operation cost of the real-time optimization methods, $cost^{MILP}$is the cost of the day-ahead MILP method. And the boxplot of the optimal operation cost error is presented in Fig. 11. We can see from it that the cost values of the four algorithms are basically close, and the PPO algorithm can get the lowest cost error compared with the other three methods and have better computational stability.

Finally, we test the sensitivity of the proposed approach to environmental variation. In the above calculation, we apply the PPO to real-time OEM operation based on the given forecasted

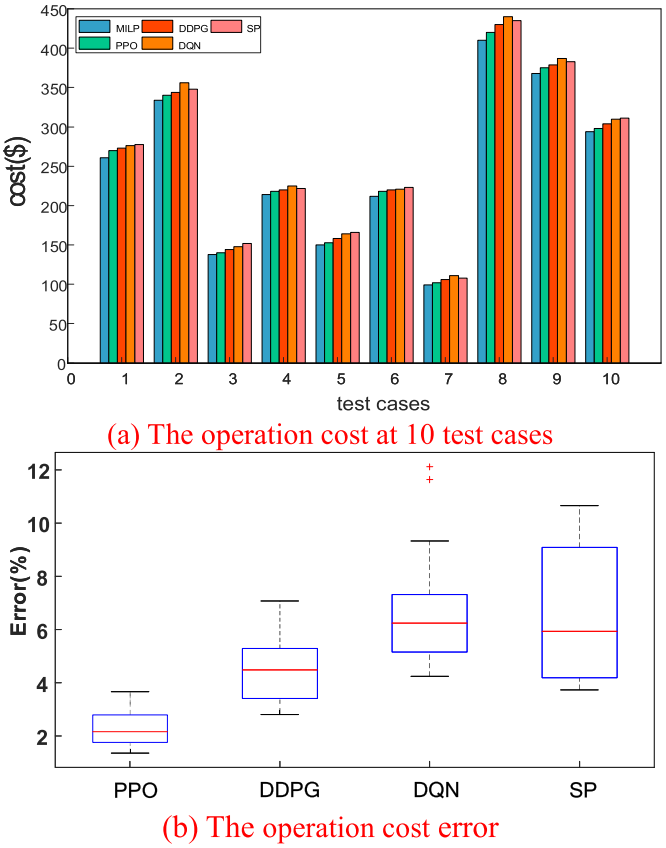

Fig. 11. The comparison of operation cost based on the PPO, DDPG, DQN, and SP method.

environmental data. Since there is a forecast deviation for renewable energy and load even in the real-time phase, it is noteworthy that to what extend the difference between forecasted value and actual value will affect the final OEM result. We take the wind power fluctuation in MG as an example. The Monte-Carlo simulation is used to sample the daily wind power profile with forecast error for 20 times. The forecast error is within the range of $[-\alpha\%,\alpha\%],\alpha\in\{10,15,20\}$. Table 4 shows the MG OEM results with different forecast errors. The rolling scheduling MILP method with MPC frmework is used as a benchmark. The results indicate that with forecast error growing, the variance increases. While comparing with MPC-MLIP, the proposed PPO algorithm can better adapt to the power fluctuation and obtain a more stable solution.

Table 4
The MG OEM results with forecast error.

Error	Operation cost	PPO	MPC
10%	mean	271.27	287.44
	variance	1.032	1.793
15%	mean	271.93	289.96
	variance	1.673	3.642
20%	mean	273.51	293.12
	variance	2.424	4.389

The model-free PPO algorithm has a good generalization to environmental variation since it can cope with uncertainty effectively through learning from the massive historical data of the environment. In the learning process, the DRL agent can capture the data feature and update the network parameters adaptively. As a result, the environmental uncertainty has been taken into consideration in the online operation.

5. Conclusion

MGs are widely implemented for their flexibility when the DG units and ESS devices need to be integrated. The model-free DRL algorithm is proposed to solve the real-time MG OEM problem considering uncertainties. The DRL-based OEM is formulated under an MDP framework and solved by the PPO, a state-of-the-art policy-based actor-critic DRL algorithm. The proposed approach is tested on an MG case and the following conclusions can be drawn.

(1) The MDP formulation for MG OEM is suitable to describe the interaction relationship between the EMS agent and the environment. And the environmental uncertainties can be reflected in the state transition.
(2) The simulation results verify that the proposed PPO algorithm exhibits better training efficiency to obtain high cumulative reward than the DDPG and DQN. The PPO learns optimal policies by exploring the continuous action space in a more stable update rule. After offline training based on historical environmental data, the DRL can be directly applied to the online real-time decision, reducing the computational time obviously compared with the SP which needs to calculate repeatedly at every time slot.
(3) The performance comparison demonstrates that the PPO can tackle uncertainties effectively. With the property of learning from the environment historical data automatically, the PPO will consider the future uncertainties, not relying on the exact probability distribution. Besides, the PPO exhibits good generalization capabilities for environment variation and obtain a stable operation cost.

Future research can focus on further improving the performance of DRL using advanced algorithms and the new mechanism of coordinated operation of multi-MGs.

Author statement

Chenyu Guo: Software, Writing – original draft, Visualization, Xin Wang: Supervision, Methodology, Validation, Yihui Zheng: Conceptualization, Investigation, Funding acquisition, Feng Zhang: Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science

Foundation of China [Grant Nos. 61533012 and 61673268].

Appendix A

Setting appropriate hyperparameters is important for the DRL algorithm to get an ideal result. So we take further analysis of some hyperparameters in this section.

A.1 Penalty coefficient

Firstly, the influence of the penalty coefficient in the reward function is analyzed. The cumulative reward during the training process with different penalty coefficient values is shown in Fig. A1. It can be seen, if the penalty coefficient value is too large (e.g. $v1$ and $v2$ are set to be 2), it will take a longer time to converge. While if the penalty coefficient value is too small (e.g. $v1$ and $v2$ are set to be 0.1), the penalty of constraint violation can't reflect in reward. And thus, the DNN can't be trained well to avoid constraint violation. When the $v1$ and $v2$ are set to be 0.6–1, the training process of DNN converges to an acceptable range. Furthermore, the well-trained DNN with different penalty coefficient values is applied to the real-time operation of five cases. The daily operation cost is listed in Table A1. In four of the five cases, the DNN with a penalty coefficient be 0.9 can get the lowest cost. So, we choose 0.9 as the value as the penalty coefficient.

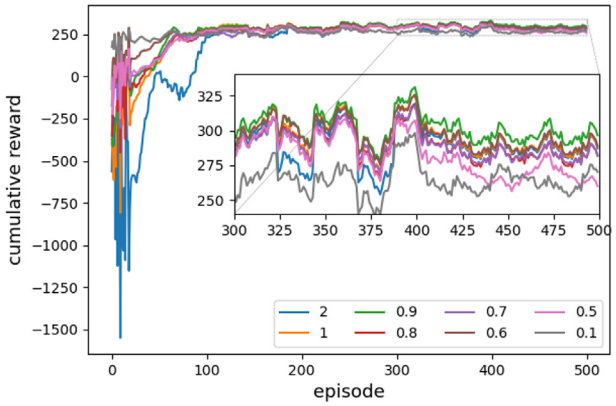

Fig. A1. The average cumulative reward with different penalty coefficient.

Tab.A1
The operation cost with different penalty coefficient

Value of v1/v2	Case 1	Case 2	Case 3	Case 4	Case 5
2/2	277.23	339.20	106.13	205.15	44.26
1/1	269.95	332.86	105.44	200.43	41.94
0.9/0.9	**268.65**	**330.11**	**103.09**	198.21	**40.57**
0.8/0.8	270.67	334.61	105.31	199.52	41.45
0.7/0.7	273.85	335.72	104.84	199.31	42.37
0.6/0.6	271.37	335.28	104.13	**198.13**	41.98
0.5/0.5	278.76	345.43	110.53	221.27	44.21
0.1/0.1	291.18	361.19	115.07	225.47	47.19

A.2 Extra reward r_t^{ex}

The proper extra reward added to the reward function can help to accelerate the training process. We add an extra reward to let the total reward value be positive. On the other hand, we need to avoid that the rise of reward is hidden by a too large extra reward. Fig. A2 presents the cumulative reward during the training process with different extra reward values. 50 is a too large value that leads to an unstable policy. While 20 is an acceptable value that let the total reward value positive.

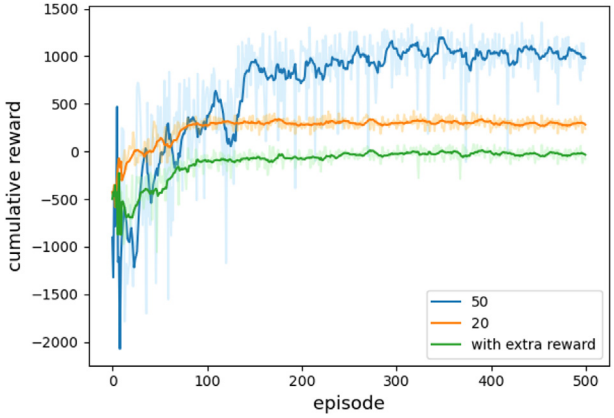

Fig. A2. The average cumulative reward with different extra reward value.

A.3 Surrogate function clipping rate ε

The clipped function helps the algorithm to converge faster. It can limit the new policy in a reasonable range, and thus reduce the influence of the uncertainty of environment state on the approximation of the value function.

A proper value should be set to avoid strategies that are too greedy or too conservative. The average cumulative reward with different clipping rate ε and without the Surrogate function is shown in Fig. A3. It can be seen that if there is no surrogate function, the training process is unstable. As a result, the DNN may choose a bad strategy. While if the ε is too small (e.g. $\varepsilon = 0.01$), the difference between new and old policy will be limited to a very small extent, and the training process will be slow down. When $0.1 < \varepsilon < 0.3$, the training process converges to close result. Referring to the original paper [24], the clipping rate ε is set to be 0.2.

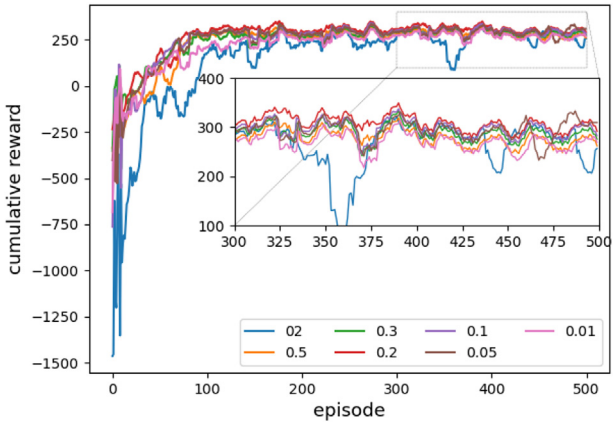

Fig. A3. The average cumulative reward with different clipping rate

A.4 Learning rate

The average cumulative reward with different learning rate is shown in Fig. A4. A small learning rate may learn slowly in the training process, while a large learning rate may fail to find optimal strategy owing to too long gradient descent step. The learning rate is set to be 0.0005.

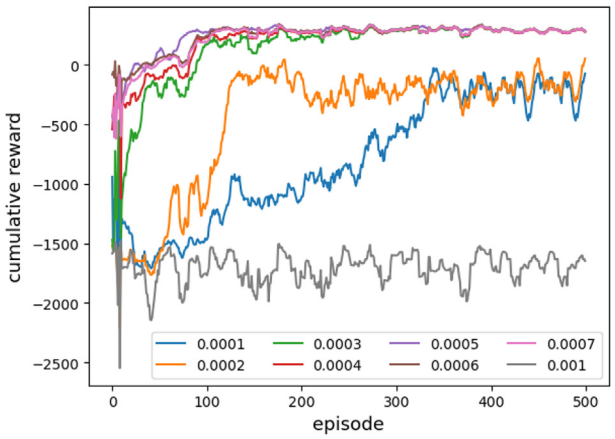

Fig. A4. The average cumulative reward with different learning rate.

Appendix B

The Hyperparameters of DQN and DDPG are listed in Table B1. The replay memory capacity N is 2000, and the minibatch sampling size F is 64 for each step. The maximum off-line training episode E is set to be 500.

For the DQN, the learning rate factor α is 0.001 and the discount factor γ is 0.9. The episode C to update target parameters is 50. The ϵ-greedy parameter ε is 0.1. The continuous actions are discretized. $P_t^{MT} = \{0,20,40,60,80, \ldots,200\}$ and $P_t^{ESS} = \{-80,-40,0,40,80\}$. So, the dimension of output action-value is 55. The DNN both have one hidden layer with 128 neurons. The input layer and hidden layer use ReLU activation function.

For the DDPG, the actor learning rate α^A is 0.0005. The critic learning rate α^C is 0.001 and the critic discount factor γ is 0.9. The soft target update τ is 0.001. The actor and critic DNN both have one hidden layer with 128 neurons in each layer. The output layer uses tahn activation function.

Table B1
the parameters of three DRL-based algorithms: DQN and DDPG

Hyperparameter	DQN	DDPG
Actor learning rate α^A	–	0.0005
Critic learning rate α^C	0.001	0.001
Discount factorγ	0.9	0.9
Exploration decay ε	0.1	–
Target Q update step C	50	–
Soft update τ	–	0.001
Memory size N	2000	2000
Minibatch sampling size F	64	64
Maximum episode number E	500	500

Algorithm B1

Training process of DQN

Input : the state of environment
Output: the Q network parameters θ^{DQN}
Initialize the Q network parameters θ^{DQN}randomly
Initialize the target Q network parameters, let $\overline{\theta}^{DQN} \leftarrow \theta^{DQN}$
For episode = 1:E **do:**
  Reset the initial state randomly
  For t = 1:T **do:**
    Observe the state s_t
    Select action a_t with ε-greedy
    Calculate reward $r_t$and obtain new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in memory D
    If current memory size > minibatch size:
      Sample minibatch of transitions from memory D
      If episode terminates at step t:
        $q_t = r_t$
      Else:
        $q_t = r_t + \gamma max\widehat{Q}(s_{t+1}, a_{t+1}; \overline{\theta}^{DQN})$
      Calculate the loss function

$$L\left(\theta_t^{DQN}\right) = \sqrt{\frac{1}{F}\sum_{t=1}^{F}\left[q_t - Q\left(s_t, a_t; \theta_t^{DQN}\right)\right]^2}$$

      Update parameters $\theta^{DQN} = \theta^{DQN} - \alpha^{DQN}\nabla_\theta L(\theta^{DQN})$
      Every C steps update the target Q network
      $\overline{\theta}^{DQN} \leftarrow \theta^{DQN}$
  End for
End for

Algorithm B2

Training process of DDPG

Input : the state of environment
Output: the critic network $Q(s,a|\theta^Q)$ and actor network$\mu(s|\theta^\mu)$ with parameters θ^Q and θ^μ
Initialize the network parameters θ^Q and θ^μrandomly
Initialize the target network parameters, let $\overline{\theta} \leftarrow \theta$
For episode = 1:E **do:**
  Reset the initial state randomly
  For t = 1:T **do:**
    Observe the state s_t
    Select action a_t
    Calculate reward $r_t$and obtain new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in memory D
    If current memory size > minibatch size:
      Sample minibatch of transitions from memory D
      If episode terminates at step t:
        $q_t = r_t$
      Else:
        $q_t = r_t + \gamma max\widehat{Q}(s_{t+1}, a_{t+1}; \overline{\theta}^{Q})$
      Update the critic by minimizing the loss function:

$$L(\theta_t^Q) = \sqrt{\frac{1}{F}\sum_{t=1}^{F}\left[q_t - Q(s_t, a_t; \theta_t^Q)\right]^2}$$

      Update the actor policy using sampled policy grandient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{F}\sum_{t}^{F}\nabla_{a_t} Q(s_t, a_t; \theta_t^Q)|_{a_t=\mu(s_t)}\nabla_{\theta^\mu}\mu(s_t)$$

      Update the actor and critic parameters:

$$\theta^Q = \theta^Q + \alpha^Q \cdot \nabla_{\theta^Q} L(\theta^Q)$$
$$\theta^\mu = \theta^\mu + \alpha^A \cdot \nabla_{\theta^\mu} J$$

      Update the target networks:

$$\overline{\theta}^Q \leftarrow \tau\theta^Q + (1-\tau)\,\overline{\theta}^Q$$
$$\overline{\theta}^\mu \leftarrow \tau\theta^\mu + (1-\tau)\,\overline{\theta}^\mu$$

  End for
End for

References

[1] Fan S, He G, Zhou X, et al. Online Optimization for Networked Distributed Energy Resources With Time-Coupling Constraints. IEEE Transactions on Smart Grid 2021;12(1):251–67. https://doi.org/10.1109/TSG.2020.3010866. 1.

[2] Qiu H, Zhao B, Gu W, et al. Bi-level two-stage robust optimal scheduling for AC/DC hybrid multi-microgrids. IEEE Trans Smart Grid 2018;9(5):5455–66.

[3] Stoppato, A, Cavazzini, G, Ardizzon, G, et al. A PSO (particle swarm optimization)-based model for the optimal management of a small PV (photovoltaic)-pump hydro.

[4] Bischi A, Taccari L, Martelli E, et al. A detailed MILP optimization model for combined cooling, heat and power system operation planning. Energy 2014;74:12–26. energy storage in a rural dry area. Energy 2014; 76:168–174.

[5] Yang J, Su C. Robust optimization of microgrid based on renewable distributed power generation and load demand uncertainty. Energy 2021;223:120043.

[6] Gomes I, Melicio R, Mendes V. A novel microgrid support management system based on stochastic mixed-integer linear programming. Energy 2021;223: 120030.

[7] Zhang Y, Meng F, Wang R, Kazemtabrizi B, Shi J. Uncertainty-resistant stochastic MPC approach for optimal operation of CHP microgrid. Energy 2019;179:1265–78.

[8] Vitale F, Rispoli N, Sorrentino M, et al. On the use of dynamic programming for optimal energy management of grid-connected reversible solid oxide cell-based renewable microgrids. Energy 2021;2:120304.

[9] Alagoz BB, Kaygusuz A, Akcin M, et al. A closed-loop energy price controlling method for real-time energy balancing in a smart grid energy market. Energy 2013;59:95–104.

[10] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature 2015;518(7540):529–33.

[11] Zhang B, Hu W, Li J, et al. Dynamic energy conversion and management strategy for an integrated electricity and natural gas system with renewable energy: deep reinforcement learning approach. Energy Convers Manag 2020;220:113063.

[12] Wen L, Zhou K, Li J, et al. Modified deep learning and reinforcement learning for an incentive-based demand response model. Energy 2020;205:118019.

[13] Du G, Zou Y, Zhang X, et al. Deep reinforcement learning based energy management for a hybrid electric vehicle. Energy 2020;201:117591.

[14] Liu H, Yu C, Wu H, et al. A new hybrid ensemble deep reinforcement learning model for wind speed short term forecasting. Energy 2020;202:117794.

[15] Kuznetsova E, Li YF, Ruiz C, et al. Reinforcement learning for microgrid energy management. Energy 2013;59:133–46.

[16] Yang JJ, Yang M, Wang MX, et al. A deep reinforcement learning for managing wind farm uncertainty through energy storage system control and external reserve purchasing. Int J Electr Power Energy Syst 2020;119:1–11.

[17] Yang T, Zhao L, Li W, et al. Dynamic energy dispatch strategy for integrated energy system based on improved deep reinforcement learning. Energy 2021;235:121377.

[18] Chen P, Liu M, Chen C, et al. A battery management strategy in microgrid for personalized customer requirements. Energy 2019;189:116245.

[19] Wen Z, O'Neill D, Maei H. Optimal demand response using device-based reinforcement learning. IEEE Trans Smart Grid 2015;6(5):2312–24.

[20] Mocanu E, et al. On-line building energy optimization using deep reinforcement learning. IEEE Trans Smart Grid 2019;10(4):3698–708.

[21] Lillicrap TP, et al. Continuous control with deep reinforcement learning. 2015. arXiv:1509.02971vol. 5.

[22] Pinto G, Piscitelli MS, Vázquez-Canteli JR, et al. Coordinated energy management for a cluster of buildings through deep reinforcement learning. Energy 2021;229:120725.

[23] Totaro S, Boukas I, Jonsson A, et al. Lifelong control of off-grid microgrid with model-based reinforcement learning. Energy 2020;232:121035.

[24] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. arXiv:1707.06347. 2017.

[25] Wang D, Qiu J, Reedman L, et al. Two-stage energy management for networked microgrids with high renewable penetration. Appl Energy 2018;226: 39–48.

[26] Xiang Y, Liu J, Liu Y. Robust energy management of microgrid with uncertain renewable generation and load. IEEE Trans Smart Grid 2016;7(2):1034–43.

[27] Liu Y, Guo L, Wang C. A robust operation-based scheduling optimization for smart distribution networks with multi-microgrids. Appl Energy 2018;228: 130–40.

[28] Lu R, Hong SH. Incentive-based demand response for smart grid with reinforcement learning and deep neural network. Appl Energy 2019;236:937–49.

[29] Sutton RS, Barto AG. Reinforcement learning: an introduction. MIT Press; 2018.

[30] Schulman J, Moritz P, Levine S, et al. High-dimensional continuous control using generalized advantage estimation. 2015. arXiv: 1506.02438.

[31] Iowa distribution test systems, http://wzy.ece.iastate.edu/Testsystem.html; [accessed 15 December 2019].

[32] Measurement and Instrumentation Data Center (MIDC), https://midcdmz.nrel.gov; [accessed 15 December 2019].

[33] Nikmehr N, Ravadanegh SN. Optimal power dispatch of multi-microgrids at future smart distribution grids. IEEE Trans Smart Grid 2015;6(4):1648–57.

[34] California independent system operator, http://oasis.caiso.com/mrioasis/logon.do; [accessed 15 December 2019].